

Diseño y Desarrollo de Sistemas de Información

Preparación del entorno para la realización de las prácticas

Software y herramientas

- Apache NetBeans IDE (versión 26)
- DBeaver Community (*Free Universal Database Tool*).
Usaremos esta herramienta libre y de código abierto para gestionar la base de datos
- Java Development Kit (JDK) (versión 24)



Apache
NetBeans IDE



DBeaver



Java™

Sistema de Gestión de Base de Datos

■ MariaDB

Información de
conexión desde
DBeaver

Configuración de la conexión "DDSI_001 en Maestro Yoda"

Ajustes de conexión

MariaDB ajustes de conexión

▼ Ajustes de conexión

- Inicialización
- Comandos de shell
- Identificación de cliente
- Transacciones
- General
- Metadatos
- Errores y timeouts
- Data Transfer
- > Data Editor
- > Editor SQL

General Driver properties SSH SSL + Network configurations...

Server

Connect by: ☒ Host ☐ URL

URL: jdbc:mariadb://172.18.1.241:3306/DDSI_001

Server Host: 172.18.1.241 Port: 3306

Database: DDSI_001

Authentication (Database Native)

Nombre de usuario: DDSI_001

Contraseña: ☒ Save password

Advanced

Server Time Zone: Auto-detect ▼

Local Client: ▼

[Connection variables information](#)

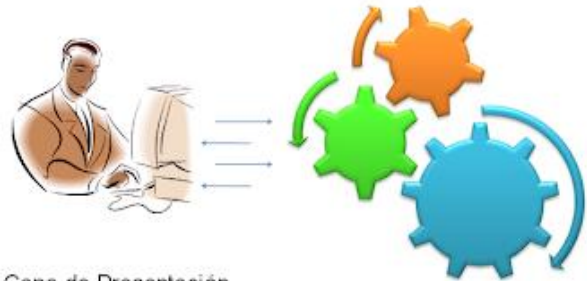
Driver name: MariaDB Driver Settings Licencia del driver

Probar conexión ... Cancelar Aceptar

Arquitectura utilizada para el desarrollo del proyecto

Modelo - Vista - Controlador

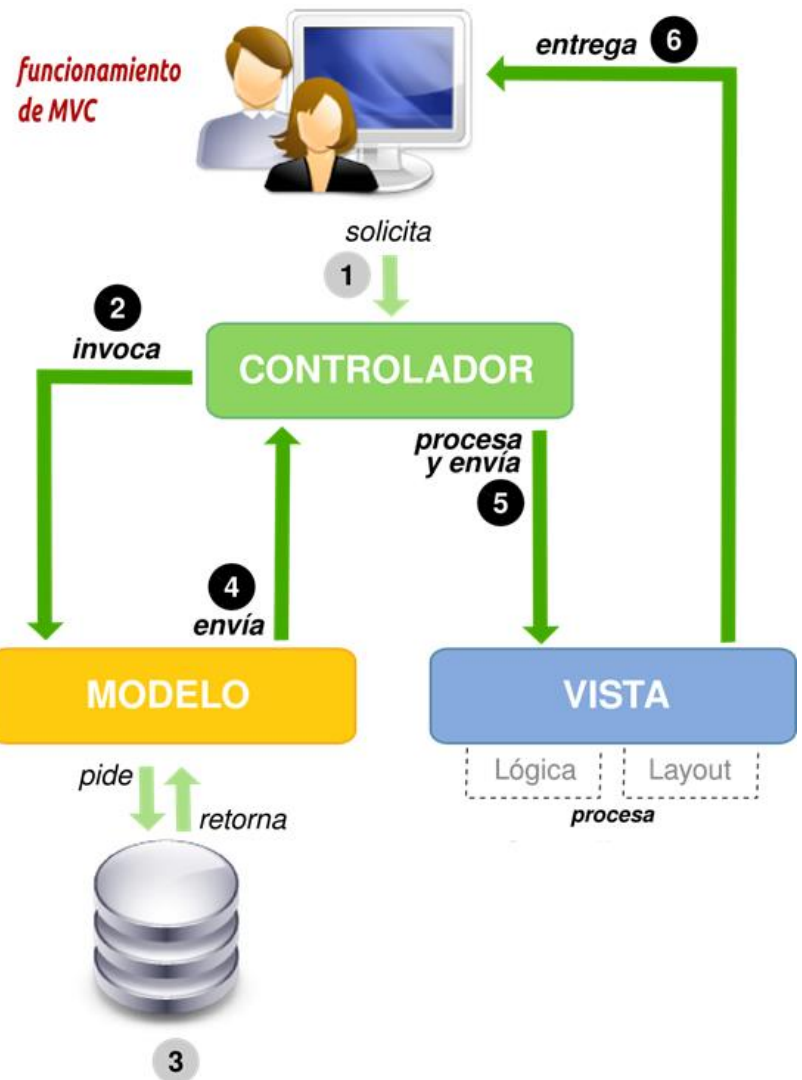
- Modelo-Vista-Controlador (**MVC**) es un patrón de arquitectura de software que organiza el código según sus responsabilidades, dividiéndolo en capas con tareas específicas
- Se utiliza principalmente en aplicaciones con interfaces de usuario, aunque puede aplicarse a cualquier tipo de aplicación
- Surge de la necesidad de crear software robusto, potenciando la facilidad de mantenimiento, reutilización del código y separación de conceptos



Capa de Presentación
(Vista al usuario)

Capa de Negocios
Controlador
(donde la mayor parte del proceso ocurre)

Capa de Datos



funcionamiento
de MVC

■ Modelo

- Capa encargada de la gestión de los datos de la aplicación
- Incluye la lógica para acceder, consultar y actualizar la información (generalmente en una base de datos)

■ Vista

- Capa responsable de la presentación de la información al usuario
- Define la interfaz gráfica y cómo se muestran los datos del modelo

■ Controlador

- Capa que actúa como enlace entre la Vista y el Modelo
- Gestiona las acciones del usuario, coordina la lógica y decide qué datos del modelo mostrar en la vista

Crear un proyecto **Maven** en
Apache NetBeans
"de un vistazo"

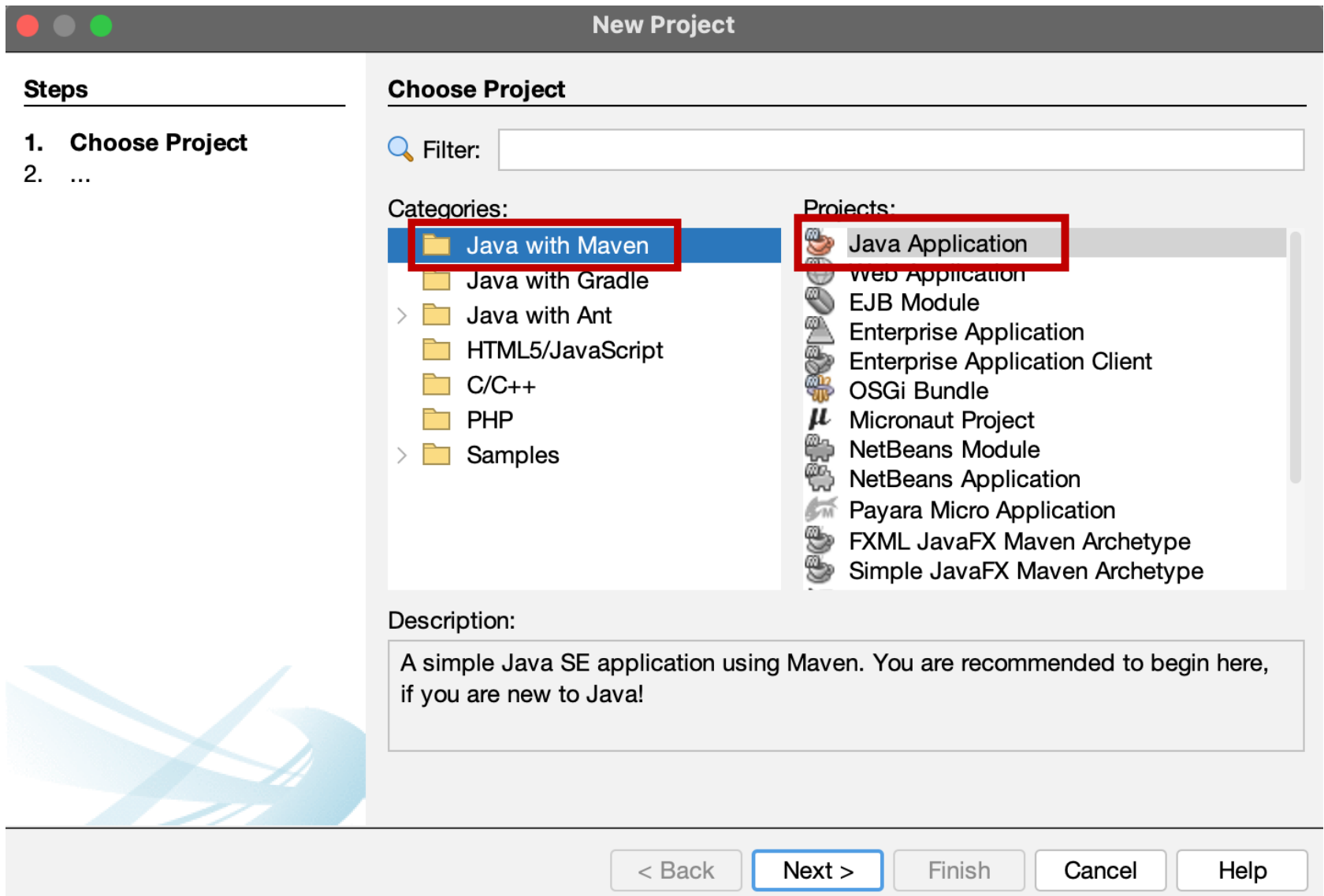
■ ¿Qué es *Maven*?

- Herramienta de gestión y construcción de proyectos Java
- *Open source* y de uso gratuito
- Facilita la compilación, empaquetado, gestión de dependencias y despliegue de aplicaciones

■ Para conocer más sobre el proyecto *Maven*

- Página oficial: <https://maven.apache.org/>
- Maven Central Repository: <https://central.sonatype.com/>





New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

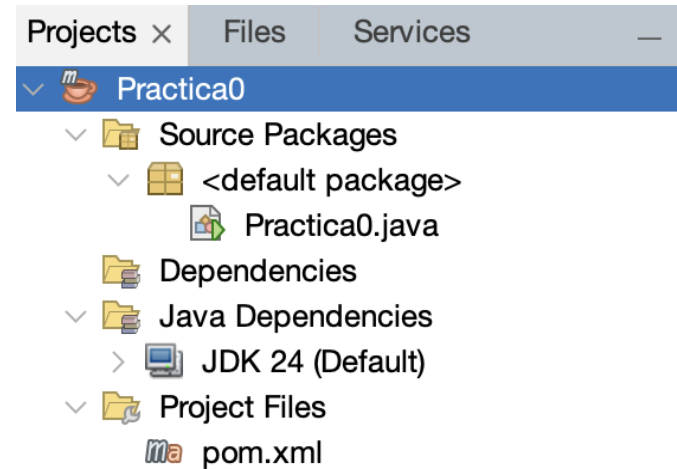
Artifact Id:

Group Id:

Version:

Package: (Optional)

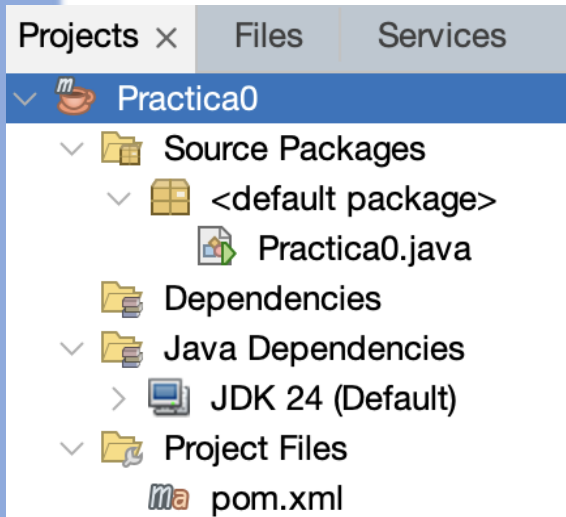
Estructura original de un proyecto



- POM (*Project Object Model*)

- Archivo principal de configuración de Maven ([pom.xml](#))
 - Contiene la información y los parámetros que **Maven** utiliza para construir y gestionar el proyecto
 - Elementos clave:
 - **<artifactId>** → Nombre único del proyecto dentro de un grupo
 - **<groupId>** → Identifica al grupo o conjunto de proyectos relacionados
 - **<version>** → Estado del proyecto (ej. SNAPSHOT para desarrollo, sin SNAPSHOT para producción)
 - **<maven.compiler.release>** → Indica la versión de Java para la cual se compilará el código.
- En este curso lo usaremos, principalmente, para añadir las dependencias (por ejemplo, los conectores para acceder al SGBD).

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>DDSI</groupId>
  <artifactId>Practica0</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.release>24</maven.compiler.release>
    <exec.mainClass>Practica0</exec.mainClass>
  </properties>
</project>
```

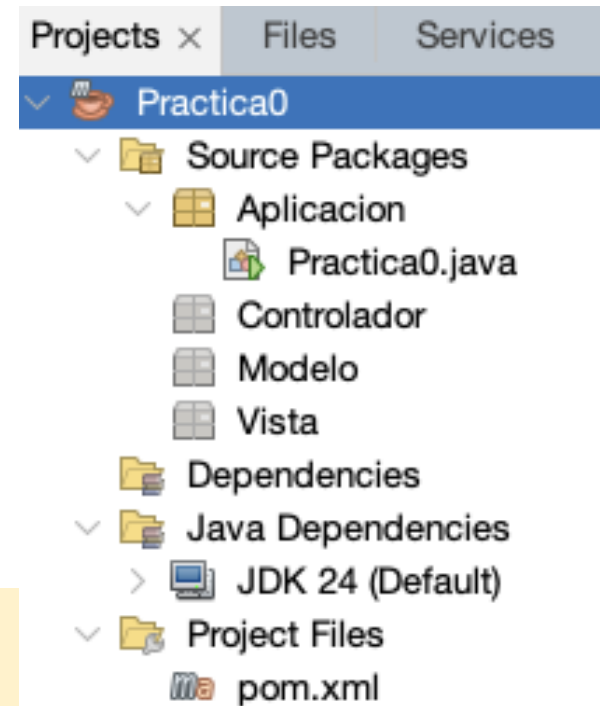


En la carpeta "Source Packages" crearemos la estructura de paquetes (capas) que tendrá el proyecto

Con el botón derecho en "Source Package" seleccionamos "New" y "Java Package" y creamos los paquetes Modelo, Vista, Controlador y Aplicacion (**sin tilde**)

El fichero .java que contiene la clase principal **main()**, y que se genera de forma automática, se trasladará al paquete "Aplicacion"

Finalmente, la estructura general del proyecto se muestra en esta figura



En las propiedades del proyecto (botón derecho) hay que especificar cuál es el fichero que contiene el método **main()**. Dentro de **"Run"**, en la propiedad **"Main Class"**

- Para incluir las distintas dependencias que vaya necesitando nuestro proyecto, editamos el fichero pom.xml y añadimos la sección `<dependencies>` `</dependencies>`

```
<dependencies>
```

```
  <dependency>
```

```
    <groupId>org.hibernate.orm</groupId>
```

```
    <artifactId>hibernate-core</artifactId>
```

```
    <version>7.1.0.Final</version>
```

```
  </dependency>
```

Dependencia para
trabajar con Hibernate

```
  <dependency>
```

```
    <groupId>jakarta.persistence</groupId>
```

```
    <artifactId>jakarta.persistence-api</artifactId>
```

```
    <version>3.2.0</version>
```

```
  </dependency>
```

Dependencia para
trabajar con la
persistencia de Java

```
  <dependency>
```

```
    <groupId>org.mariadb.jdbc</groupId>
```

```
    <artifactId>mariadb-java-client</artifactId>
```






















```
    <version>3.5.5</version>
```

```
  </dependency>
```

```
</dependencies>
```

Dependencia para la
conexión con el servidor
MariaDB

- Al grabar el fichero pom.xml, en la carpeta **"Dependencies"** aparecerán las librerías **".jar"** necesarias para desarrollar el proyecto

- ✓  Dependencies
 - >  hibernate-core-7.1.0.Final.jar
 - >  jakarta.persistence-api-3.2.0.jar
 - >  mariadb-java-client-3.5.5.jar
 - >  jakarta.transaction-api-2.0.1.jar
- ✓  Runtime Dependencies
 - >  angus-activation-2.0.2.jar
 - >  antlr4-runtime-4.13.2.jar
 - >  byte-buddy-1.17.6.jar
 - >  classmate-1.7.0.jar
 - >  hibernate-models-1.0.1.jar
 - >  istack-commons-runtime-4.1.2.jar
 - >  jakarta.activation-api-2.1.3.jar
 - >  jakarta.inject-api-2.0.1.jar
 - >  jakarta.xml.bind-api-4.0.2.jar
 - >  jaxb-core-4.0.5.jar
 - >  jaxb-runtime-4.0.5.jar
 - >  jboss-logging-3.6.1.Final.jar
 - >  txw2-4.0.5.jar
- ✓  Java Dependencies
 - >  JDK 24 (Default)

Con el botón derecho en el proyecto, seleccionamos **"Build with Dependencies"** para descargarlas y enlazarlas con el programa