

# PERSISTENCIA

Objeto que puede almacenar su estado para ser reusado.

# Introducción

## Persistencia

- Base de Datos
- Tablas
- Claves primarias
- Índices
- Relaciones entre tablas
- Formas normales

## OO

- Objetos: encapsula comportamiento y estado
- Herencia
- Colecciones
- Clases: abstractas, concretas

## Forma de persistencia

- JDBC (Java database **Connectivity**): API estándar para acceder a BD relacional.
- ORM (Object-Relational **Mapping**): delega el acceso a la BD a una herramienta externa. Hibernate, TopLink Java Data Object.
- JPA (Java Persistence **API**): integra el modelo relacional y orientado-objetos. Incluida en Java EE 6.

## JPA especificaciones

- ORM: Mecanismo que mapea objetos a datos almacenados en una BD.
- API para manejar entidades: para realizar operaciones sobre la BD (crear, leer, etc.)
- Lenguaje de consulta (**JPQL**): recuperar datos con un lenguaje query OO.
- Mecanismo de transacciones y bloqueo que controla accesos concurrentes a los datos (**JTA API**).

## //Conceptos previos

- a) Entidad
- b) Mapeo relacional-objetos
  - Anotaciones
  - Descriptores XML
- c) Unidad de persistencia

## a) Entidad

Objetos



Son instancias que viven en memoria.

Entidad



Son objetos que viven en memoria y **persisten** en una bd.  
Especificadas por JPA.

## b) Mapeo relacional- objetos

- Anotaciones: Proveen información sobre clases, métodos, variables o cualquier otro elemento del programa.
  - Ej. `@Override`

El mapeo esta definido en  
**`jakarta.persistence`**

- Descriptores XML: El mapeo esta definido en un archivo externo xml y sera desplegado con la entidad.

## c) Unidad de persistencia

- Se genera un archivo [persistence.xml](#)
- Contiene toda la información necesaria para conectarse a la bd y para determinar el modo de generar la bd.
- El proveedor, en nuestro caso, **eclipseLink**



## Recursos:

1. NetBeans -> **Actual**
2. JDK -> **Versión 11**
3. Servidor -> **GlassFish Server 7.0.15**

## Pasos del proceso

1. Crear la base de datos (tablas, índices, relaciones, etc).
2. Crear la proyecto web.
3. Generar la clase entidad.
4. Crear la unidad de persistencia.
5. Generar la clase java que manejará la entidad.

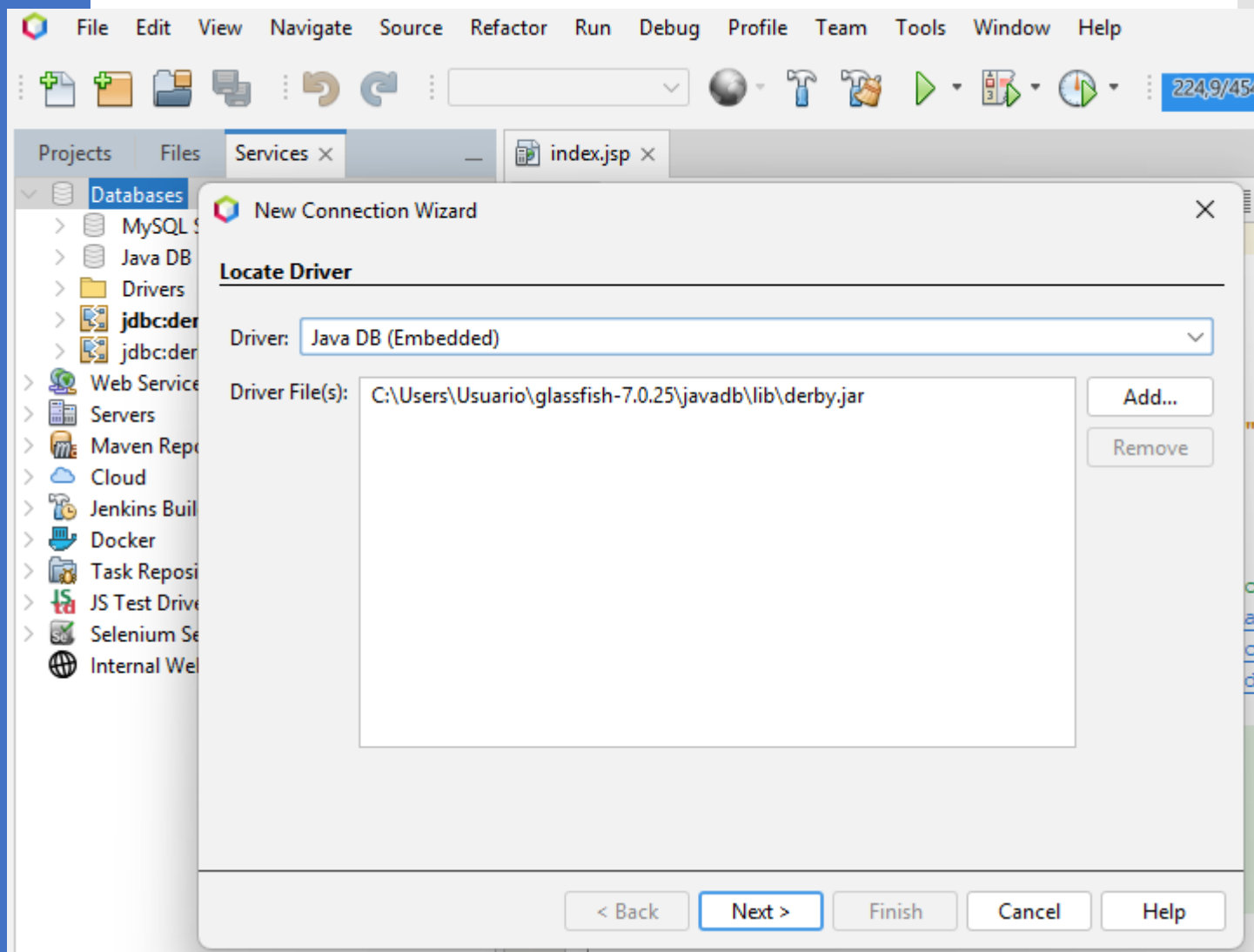
# 1. Crear la base de datos

## Derby

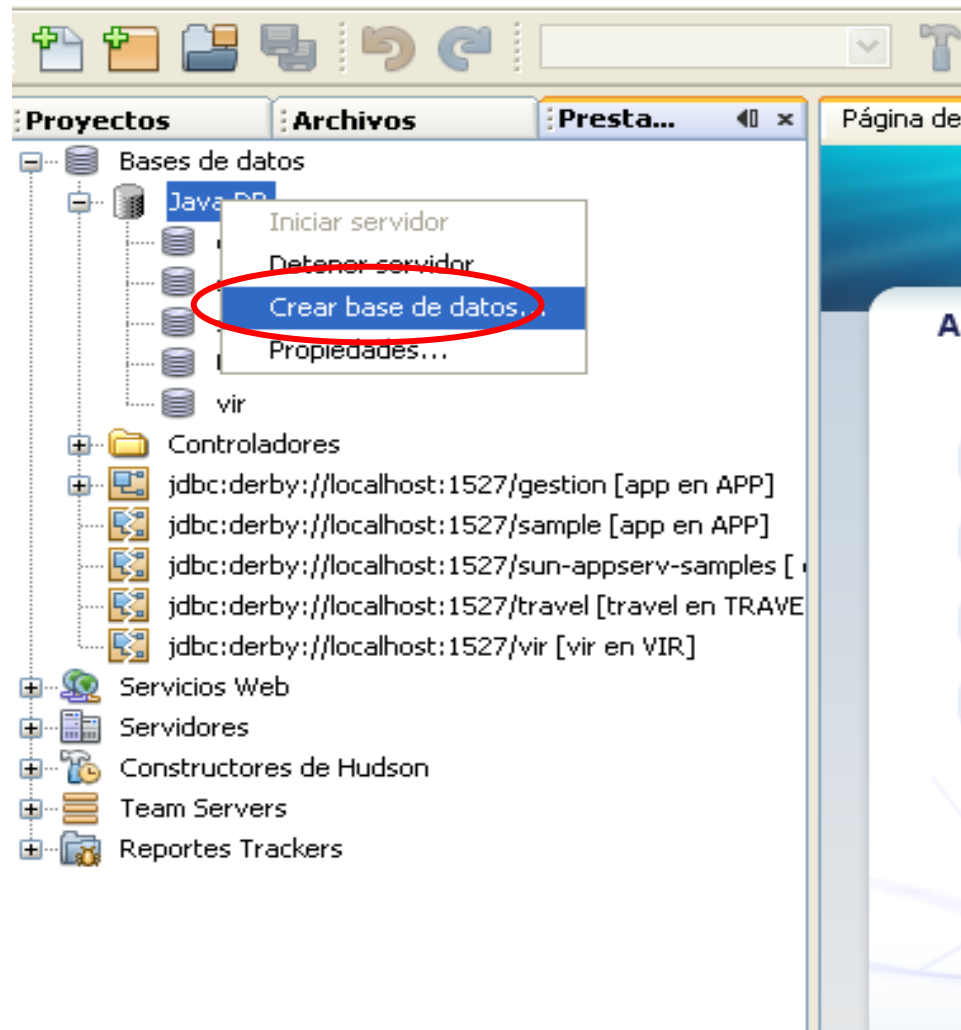
- Desarrollada en java por IBM.
- Transferida a la Fundación Apache.
- Convertida en open source.
- Sun Microsystems lanzo su propia versión como **JavaDB**
- **Estaba** incluida en JDK.

<https://db.apache.org/derby/>

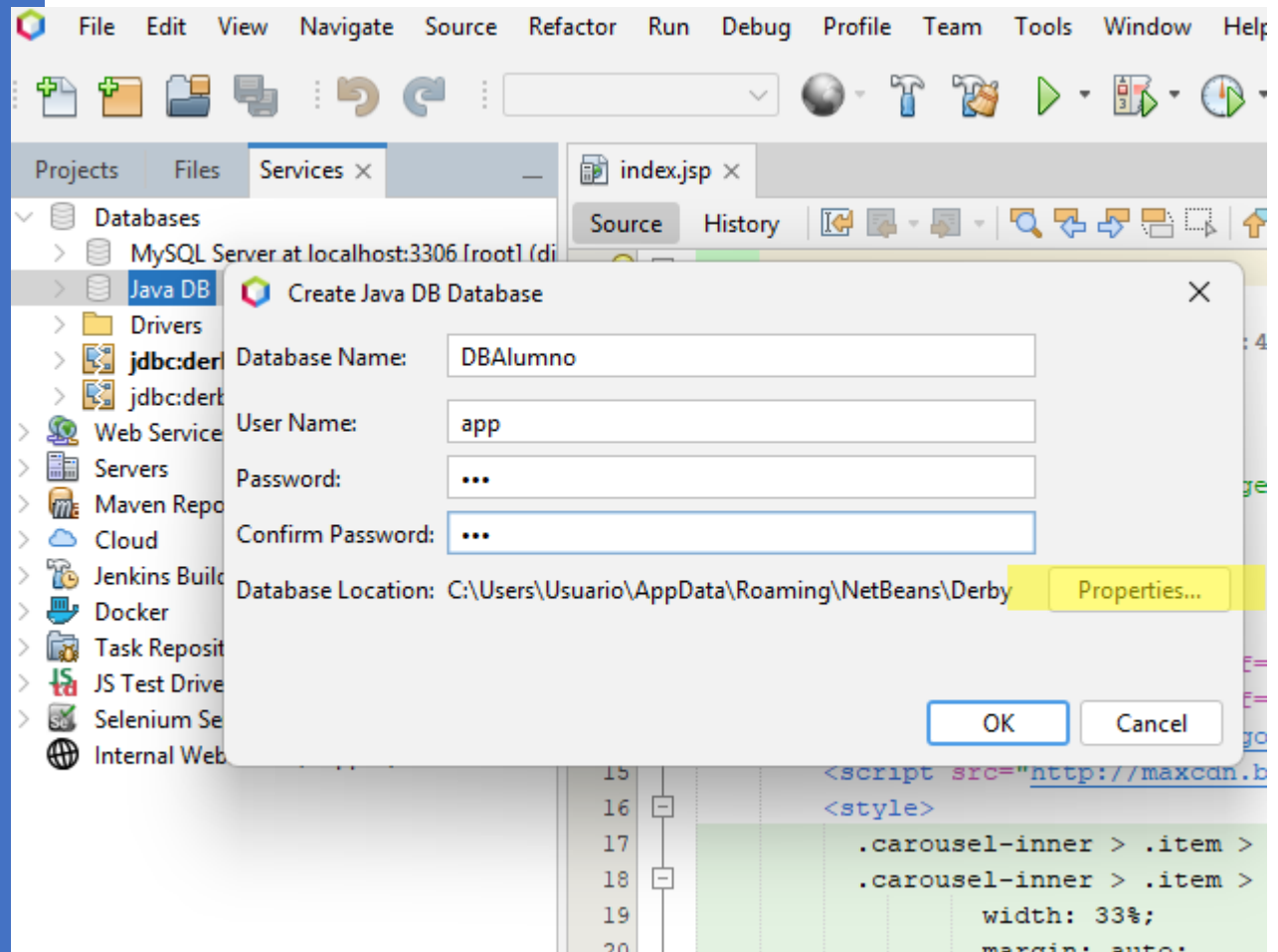
Ayuda



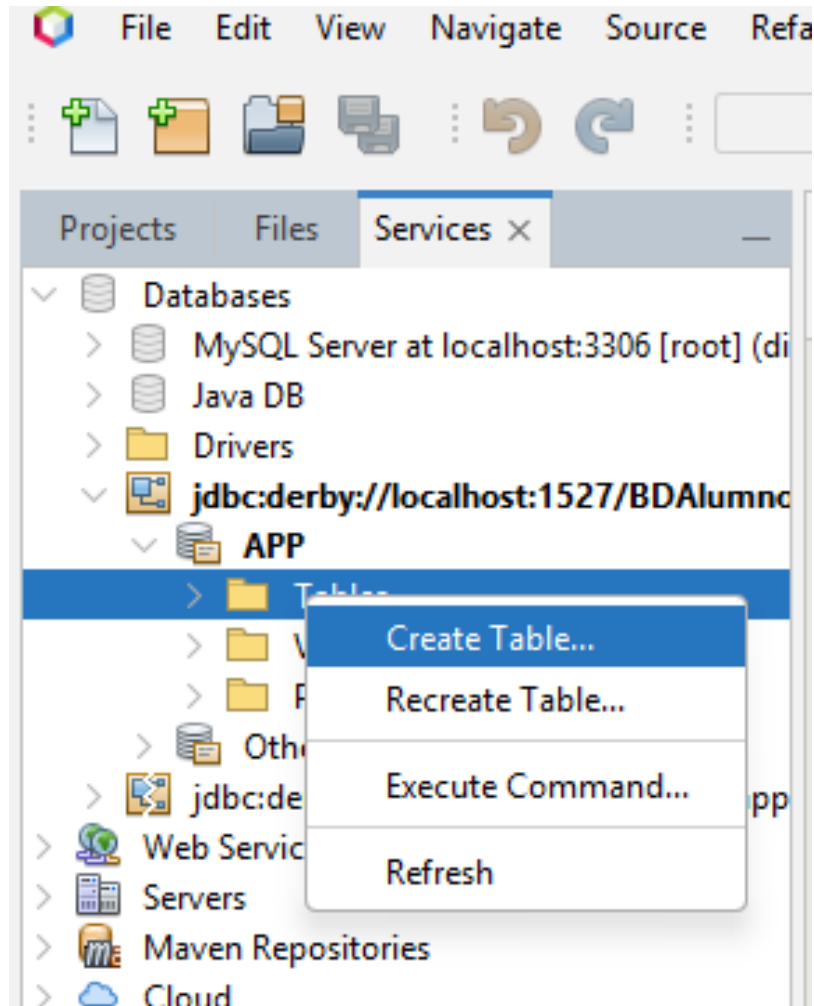
# 1. Crear bd



# 1. Crear bd



# 1. Crear tabla



Create Table

Table name: alumno

Key	Index	Null	Unique	Column name	Data type	Size
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	nombre	VARCHAR	20
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	carrera	VARCHAR	5
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	registro	VARCHAR	10

Add column

Edit

Remove

Move Up

Move Down

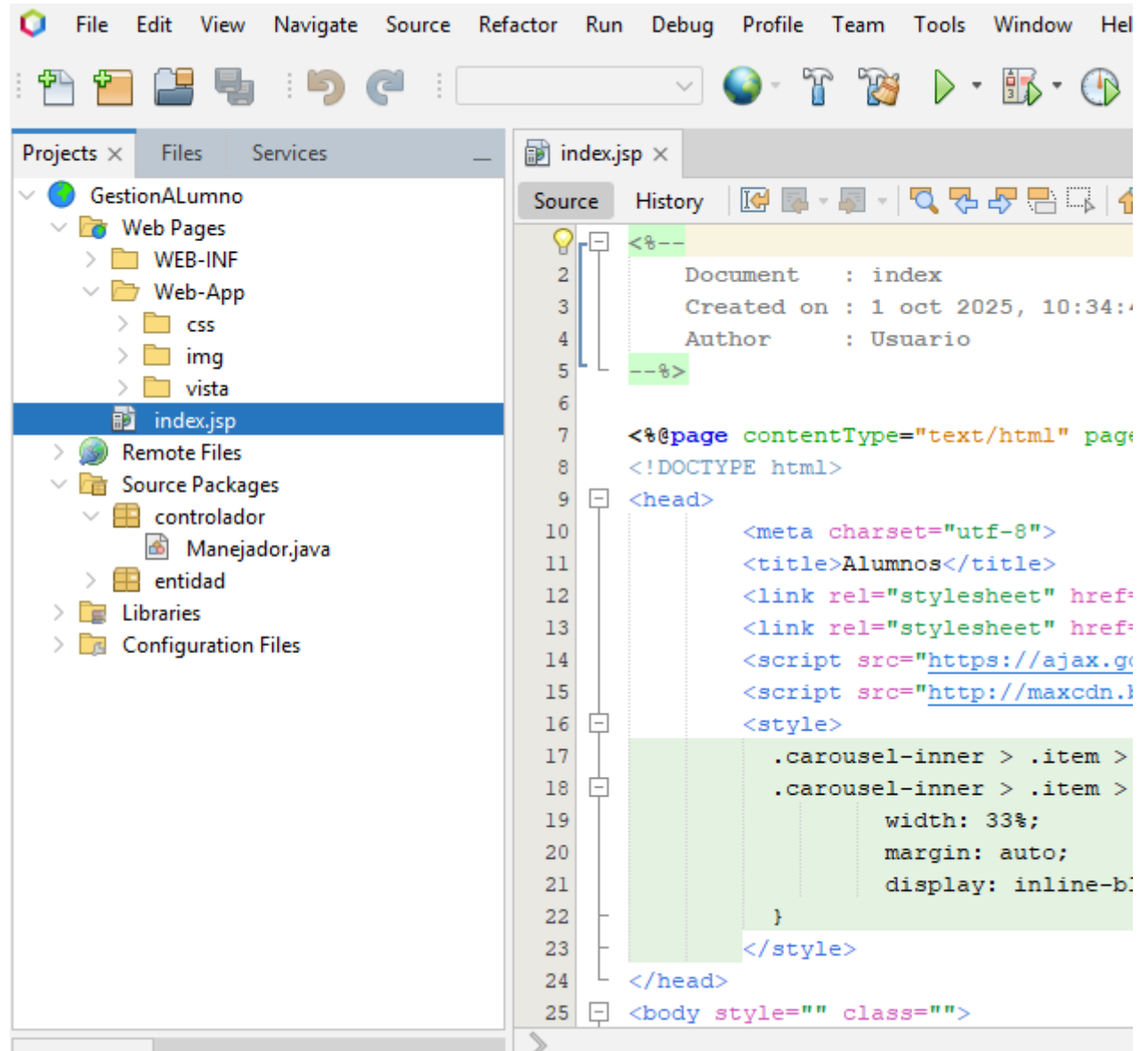
OK

Cancel

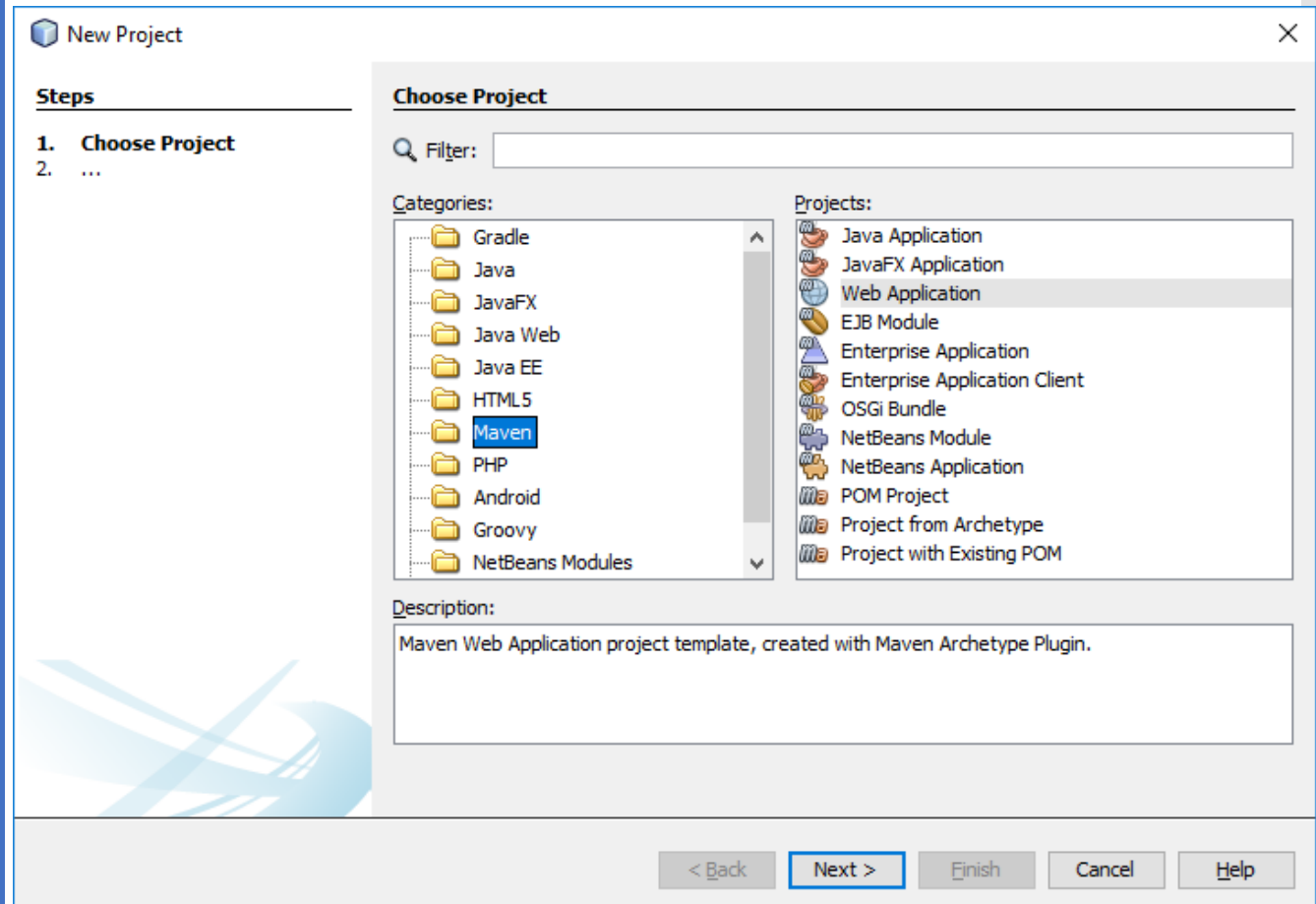
Help



## 2. Crear proyecto web (siguiendo el ej.)



## 2. Crear proyecto con maven (propuesta)



<http://maven.apache.org/>

## New Web Application

### Steps

1. Choose Project
- 2. Name and Location**
3. Settings

### Name and Location

Project Name:

Project Location:

Project Folder:

Artifact Id:

Group Id:

Version:

Package:

(Optional)

IDE interface showing the Maven POM file for the 'Alumnos' project.

**Projects:** Alumnos

- Web Pages
  - index.html
- Source Packages
  - com.taw.alumnos
- Dependencies
  - javaee-web-api-7.0.jar
- Java Dependencies
  - JDK 1.8 (Default)
- Project Files
  - pom.xml
  - nb-configuration.xml

**Navigator:** POM model

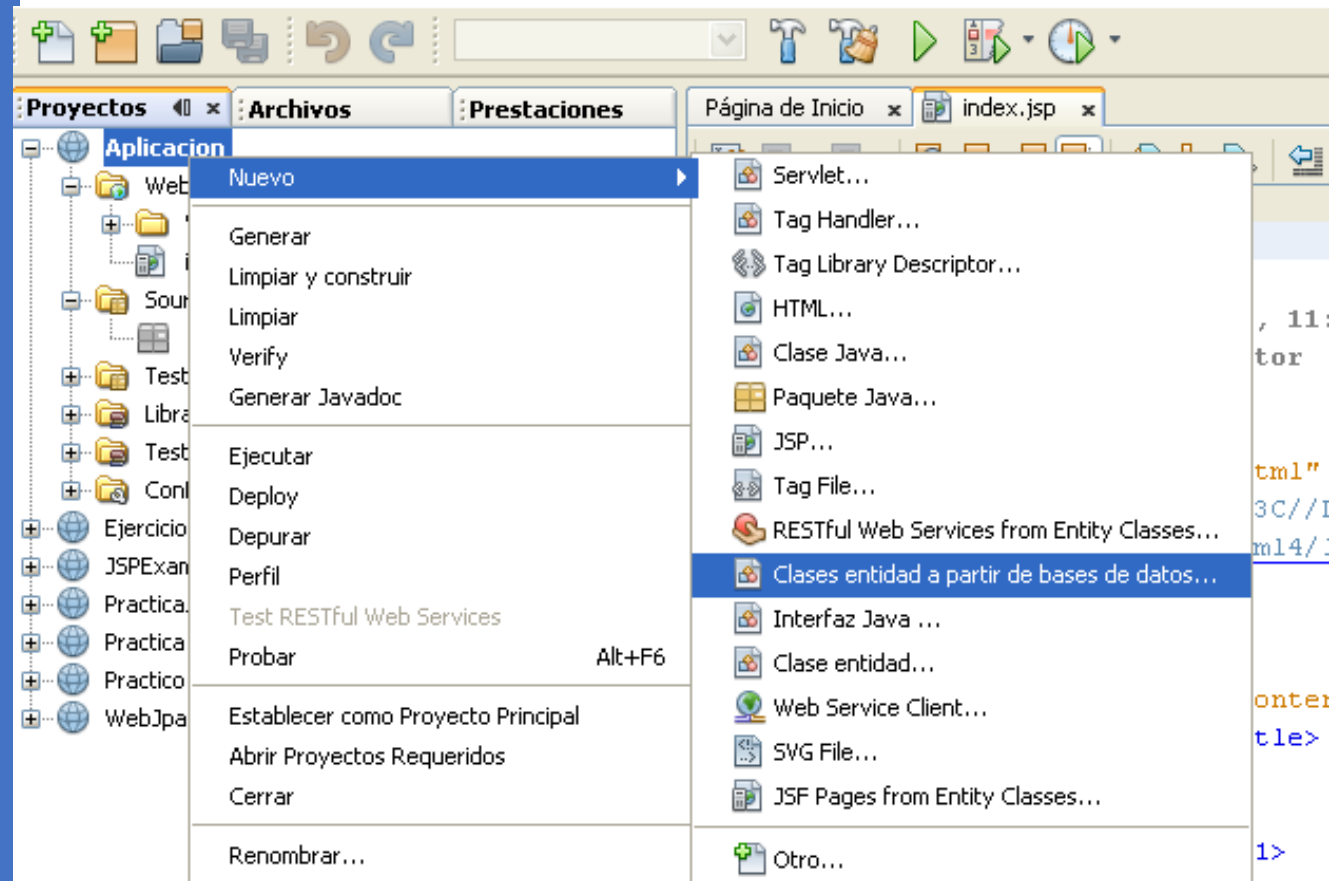
- Model Version : 4.0.0

**Source Code (pom.xml):**

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
3     <modelVersion>4.0.0</modelVersion>
4
5     <groupId>com.taw</groupId>
6     <artifactId>Alumnos</artifactId>
7     <version>1.0-SNAPSHOT</version>
8     <packaging>war</packaging>
9
10    <name>Alumnos</name>
11
12    <properties>
13        <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
14        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15    </properties>
16
17    <dependencies>
18        <dependency>
19            <groupId>javax</groupId>
20            <artifactId>javaee-web-api</artifactId>
21            <version>7.0</version>
22            <scope>provided</scope>
23        </dependency>
24    </dependencies>
25
26    <build>
27        <plugins>
28            <plugin>
29                <groupId>org.apache.maven.plugins</groupId>
30                <artifactId>maven-compiler-plugin</artifactId>
31                <version>3.1</version>
```

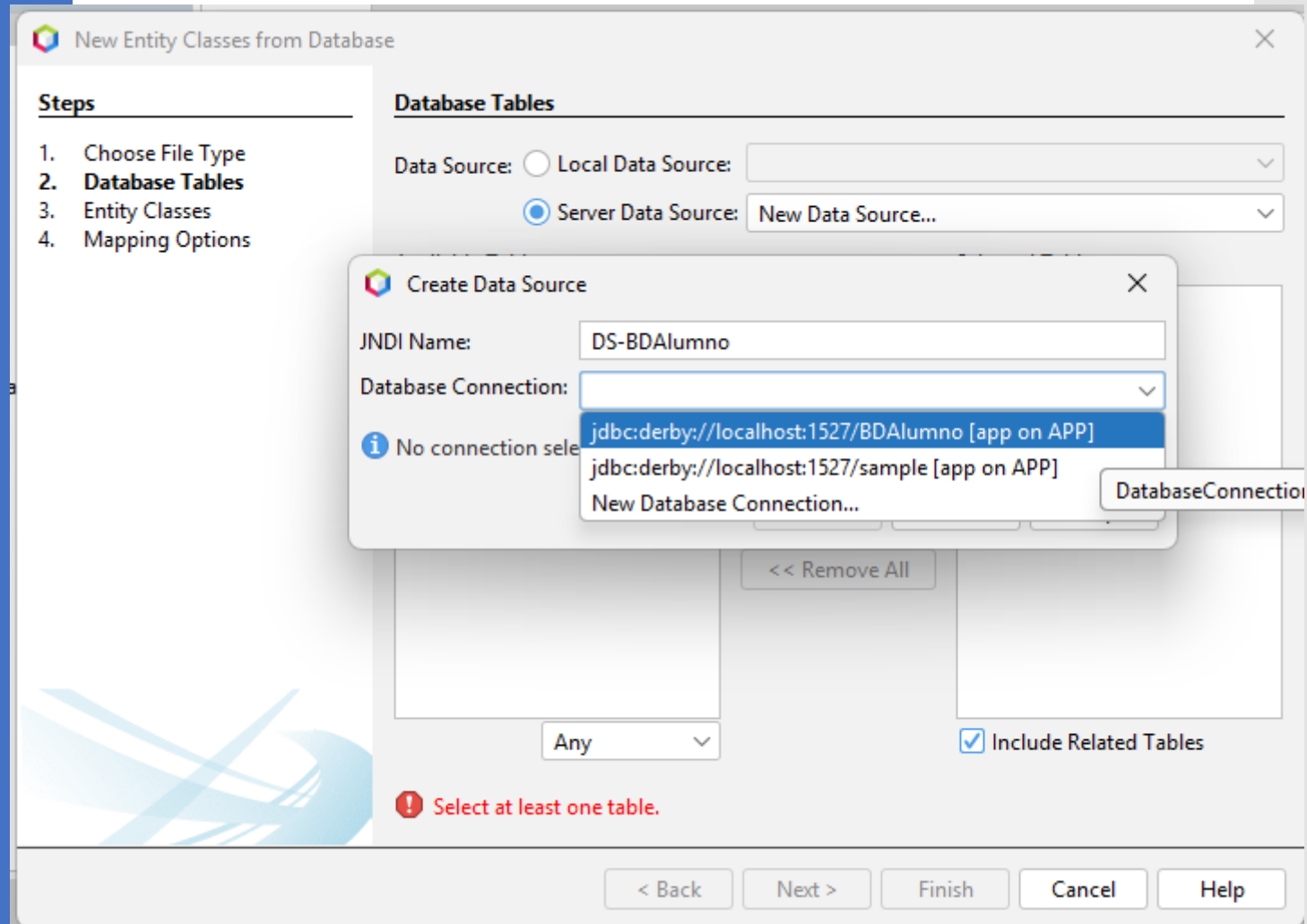
Una vez creado un nuevo proyecto web, con servidor GlassFish:

### 3. Generar la entidad



Nota: El servidor debe estar iniciado !!!

### 3. Generar la entidad



...crear la  
unidad de  
persistencia

New Entity Classes from Database

**Steps**

1. Choose File Type
2. Database Tables
3. **Entity Classes**
4. Mapping Options

**Entity Classes**

Specify the names and the location of the entity classes.

Class Names:

Database Table	Class Name	Generation Type
ALUMNO	Alumno	Update

...

Project: GestionAlumno

Location: Source Packages

Package: entidad

☒ Generate Named Query Annotations for Persistent Fields

☒ Generate JAXB Annotations

☐ Generate MappedSuperclasses instead of Entities

< Back Next > Finish Cancel Help

## New Entity Classes from Database

### Steps

1. Choose File Type
2. Database Tables
3. Entity Classes
4. **Mapping Options**

### Mapping Options

Specify the default mapping options.

Association Fetch: default

Collection Type: java.util.Collection

- ☒ Fully Qualified Database Table Names
- ☒ Attributes for Regenerating Tables
- ☒ Use Column Names in Relationships
- ☐ Use Defaults if Possible
- ☐ Generate Fields for Unresolved Relationships

< Back

Next >

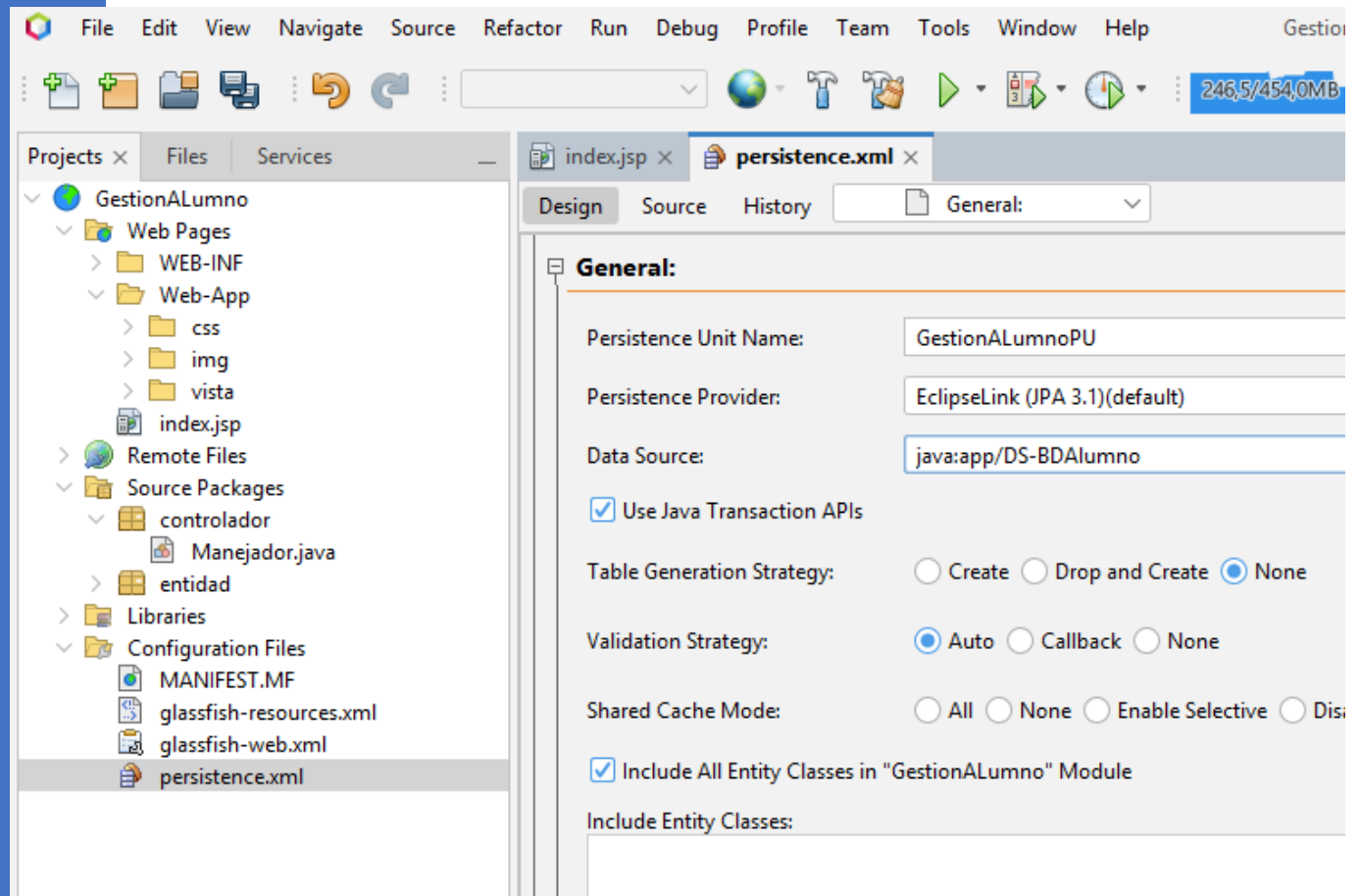
Finish

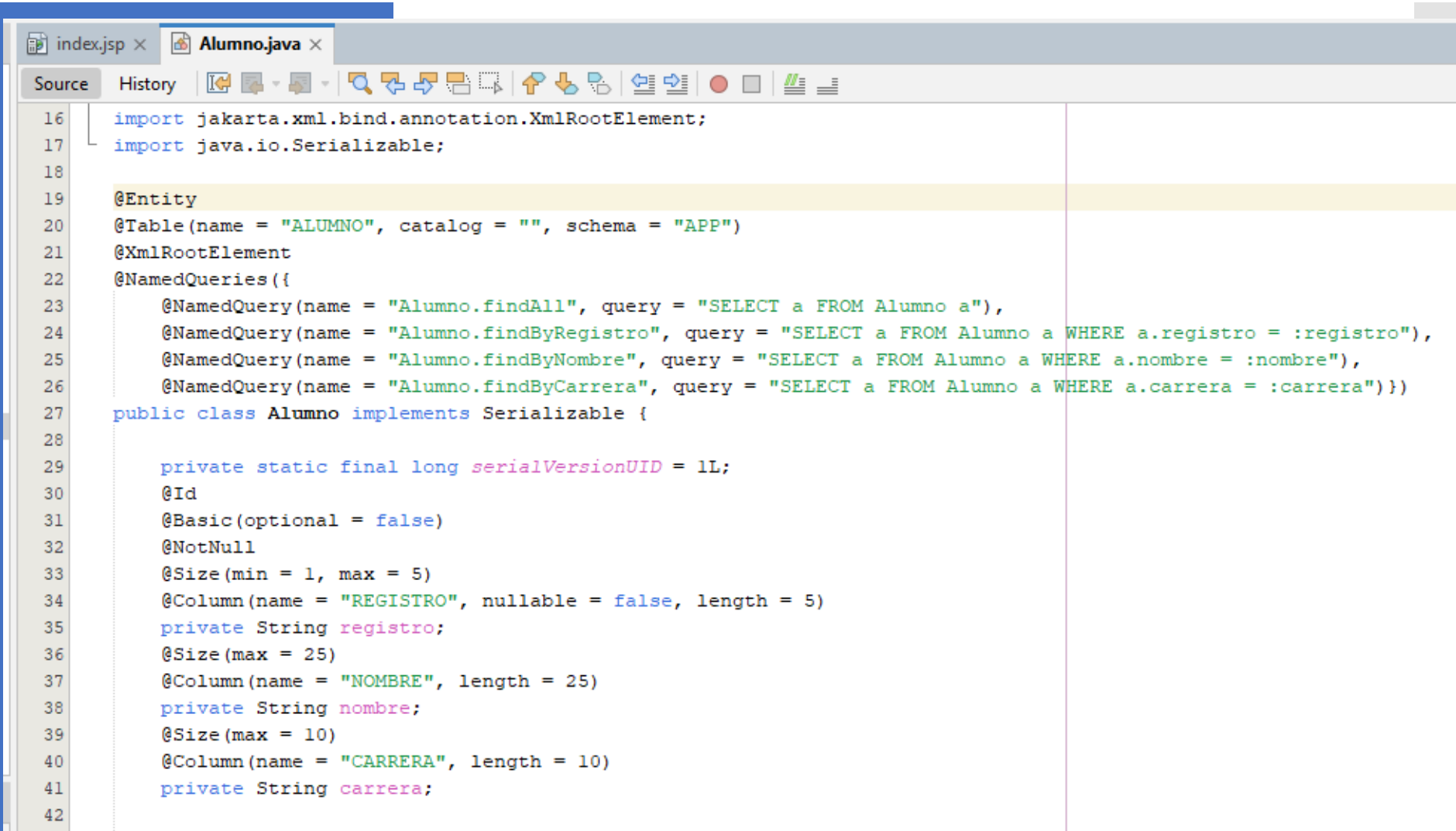
Cancel

Help



# //Código Generado - Unidad de persistencia





```
16 import jakarta.xml.bind.annotation.XmlRootElement;
17 import java.io.Serializable;
18
19 @Entity
20 @Table(name = "ALUMNO", catalog = "", schema = "APP")
21 @XmlRootElement
22 @NamedQueries({
23     @NamedQuery(name = "Alumno.findAll", query = "SELECT a FROM Alumno a"),
24     @NamedQuery(name = "Alumno.findByRegistro", query = "SELECT a FROM Alumno a WHERE a.registro = :registro"),
25     @NamedQuery(name = "Alumno.findByNombre", query = "SELECT a FROM Alumno a WHERE a.nombre = :nombre"),
26     @NamedQuery(name = "Alumno.findByCarrera", query = "SELECT a FROM Alumno a WHERE a.carrera = :carrera"))
27 public class Alumno implements Serializable {
28
29     private static final long serialVersionUID = 1L;
30     @Id
31     @Basic(optional = false)
32     @NotNull
33     @Size(min = 1, max = 5)
34     @Column(name = "REGISTRO", nullable = false, length = 5)
35     private String registro;
36     @Size(max = 25)
37     @Column(name = "NOMBRE", length = 25)
38     private String nombre;
39     @Size(max = 10)
40     @Column(name = "CARRERA", length = 10)
41     private String carrera;
42 }
```

## 5. Generar la clase Manejador

```
@WebServlet(name = "Manejador",
            loadOnStartup = 1, //Para que el sevlet se instancia e inicia cdo se despliega
            urlPatterns = {"/SolicitarDatos",
                           "/AgregarAlumno",
                           "/Listar"})

public class Manejador extends HttpServlet {
    @PersistenceUnit
    private EntityManagerFactory emf;
    @Resource
    private UserTransaction utx;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        try {
            //response.setContentType("text/html;charset=UTF-8");
            String pathUsuario = request.getServletPath();
            System.out.println("path = " + pathUsuario);
            String url = null;
            EntityManager em = null;
            switch (pathUsuario) {
                case "/SolicitarDatos":
                    url = "/WEB-INF/vista/" + pathUsuario + ".jsp";
                    break;
                case "/AgregarAlumno":
```

```
case "/AgregarAlumno":
```

```
String registro= (String) request.getParameter("registro");  
String nombre  = (String) request.getParameter("nombre");  
String carrera = (String) request.getParameter("carrera");
```

```
Alumno a = new Alumno();  
a.setRegistro(Integer.parseInt(registro));  
a.setNombre(nombre);  
a.setCarrera(carrera);
```

```
utx.begin();  
    em = emf.createEntityManager();  
    em.persist(a);  
utx.commit();
```

```
url = "index.jsp";  
break;
```

```
case "/Listar":
```

```
        case "/Listar":
            em = emf.createEntityManager();
            Query q = em.createNamedQuery("Alumno.findAll");
            List todos = q.getResultList();
            request.setAttribute("lista", todos);
            url = "/WEB-INF/vista/" + pathUsuario + ".jsp";
            break;
    }

    // usa RequestDispatcher para reTransmitir el requerimiento
    try {
        request.getRequestDispatcher(url).forward(request, response);
    } catch (ServletException | IOException ex) {
    }
}
```

# Ejercicio

- Crear la aplicación y los componentes necesarios para que funcione el **Sistema Alumnos**.

# Recursos

- [Que es Apache Derby?](#)
- [Anotaciones](#)
- [Tutorial Java EE5](#)
- [Tutorial Java EE6](#)
- [API documents](#)
- [Especificaciones Derby](#)