



Tema 2

Java EE – Primeros pasos

Persistencia y Servlets

Plataformas de Software Empresariales
Grado en Ingeniería Informática de Servicios y Aplicaciones
Curso 2020/2021

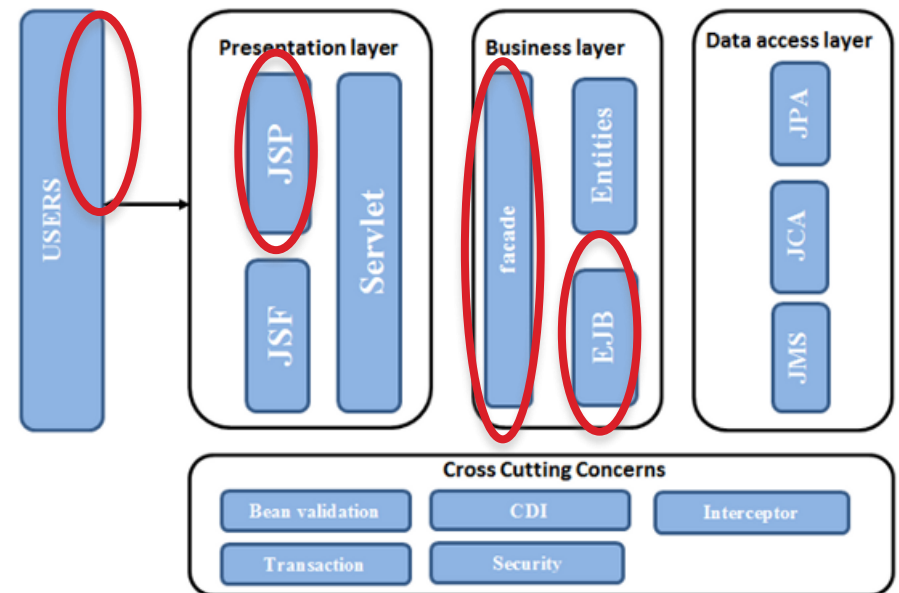
Introducción

► Objetivos

- Ver el funcionamiento de la persistencia en Java EE
- Instalar el servidor de bases de datos
- Configurar el servidor de bases de datos en Netbeans
- Crear una base de datos para la aplicación factorial
- Implementar la persistencia utilizando EclipseLink
- Crear un servlet que se llame desde el html y que guarde información en la base de datos.

Primera aplicación Java EE

- ▶ Ahora queremos almacenar la información que se genere en una base de datos.
- ▶ Vamos a utilizar PostgreSQL
- ▶ Por ejemplo, supongamos que queremos guardar el nombre de la persona que ha utilizado la aplicación y la base para la que ha consultado su factorial.



¿Qué necesitamos?

1. PostgreSQL (www.postgresql.org)
2. Herramienta de administración PostgreSQL
 1. Crear una base de datos de prueba
3. Configuración de PostgreSQL en NetBeans
 1. Crear una instancia de la base de datos en NetBeans
 2. Crear las tablas para la aplicación Factorial

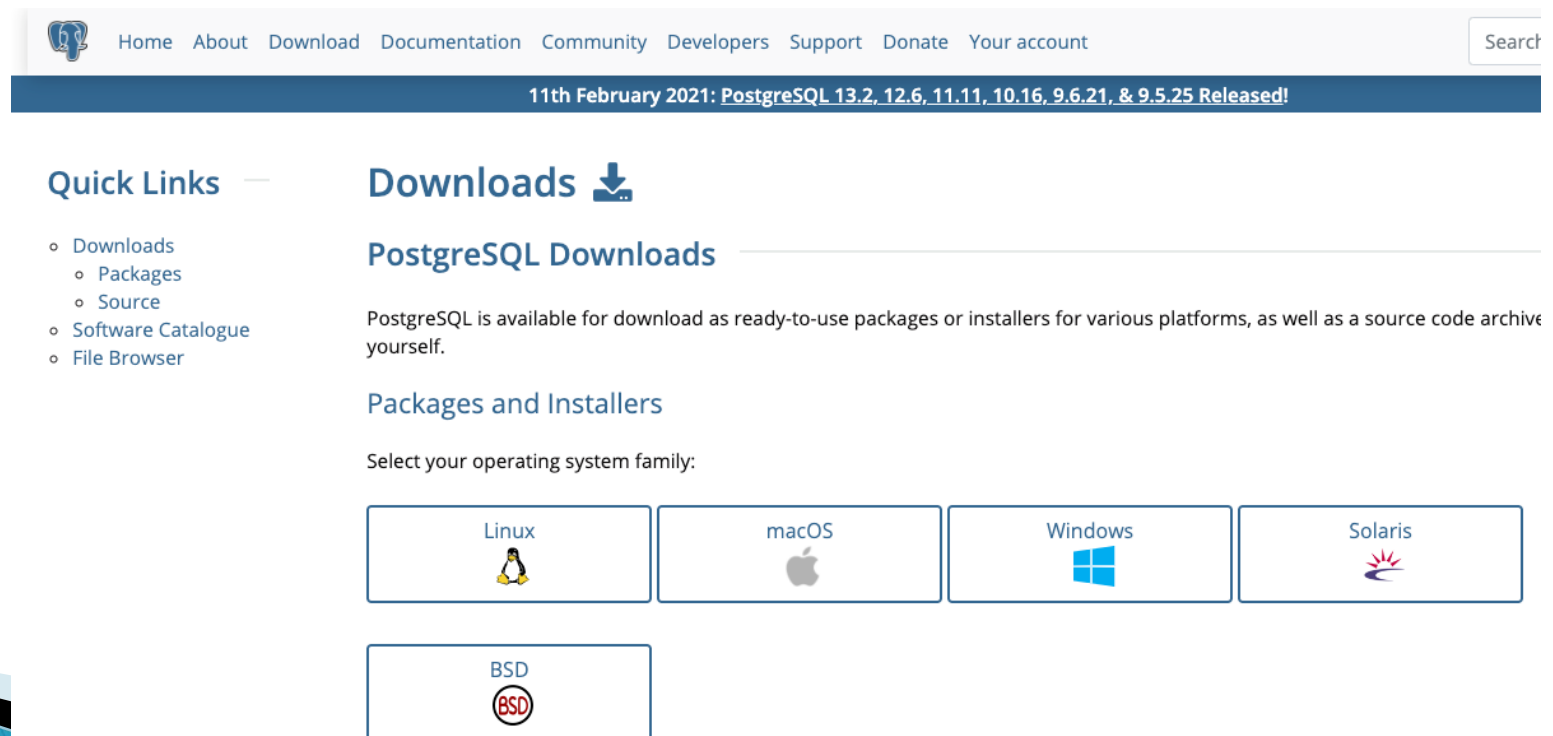


PostgreSQL

- ▶ Obtened la última versión PostgreSQL:

<https://www.postgresql.org/download/>

1. Seleccionad la versión adecuada para vuestro sistema operativo
2. Descargadla
3. Instaladla



The screenshot shows the PostgreSQL website's download page. At the top, there is a navigation bar with links: Home, About, Download, Documentation, Community, Developers, Support, Donate, and Your account. A search bar is located on the right. Below the navigation bar, a blue banner announces: "11th February 2021: PostgreSQL 13.2, 12.6, 11.11, 10.16, 9.6.21, & 9.5.25 Released!". The main content area is divided into two columns. The left column, titled "Quick Links", contains a list of links: Downloads, Packages, Source, Software Catalogue, and File Browser. The right column, titled "Downloads" with a download icon, contains the section "PostgreSQL Downloads". Below this, it states: "PostgreSQL is available for download as ready-to-use packages or installers for various platforms, as well as a source code archive yourself." The section "Packages and Installers" follows, with the instruction "Select your operating system family:". Below this instruction are five buttons for different operating systems: Linux (with a penguin icon), macOS (with an Apple icon), Windows (with a Windows logo icon), Solaris (with a Solaris logo icon), and BSD (with a BSD logo icon).

PostgreSQL

PostgreSQL Database Download

Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
13.2	N/A	N/A	Download	Download	N/A
12.6	N/A	N/A	Download	Download	N/A
11.11	N/A	N/A	Download	Download	N/A
10.16	Download	Download	Download	Download	Download
9.6.21	Download	Download	Download	Download	Download
9.5.25	Download	Download	Download	Download	Download
9.4.26 (Not Supported)	Download	Download	Download	Download	Download
9.3.25 (Not Supported)	Download	Download	Download	Download	Download

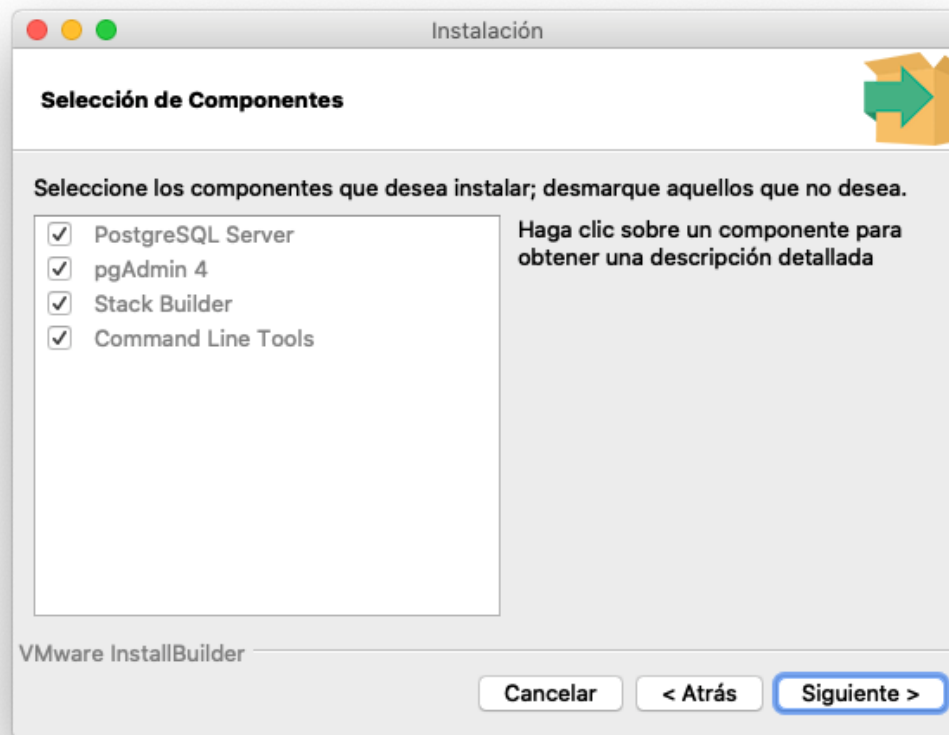
PostgreSQL

- ▶ Lo instalamos...



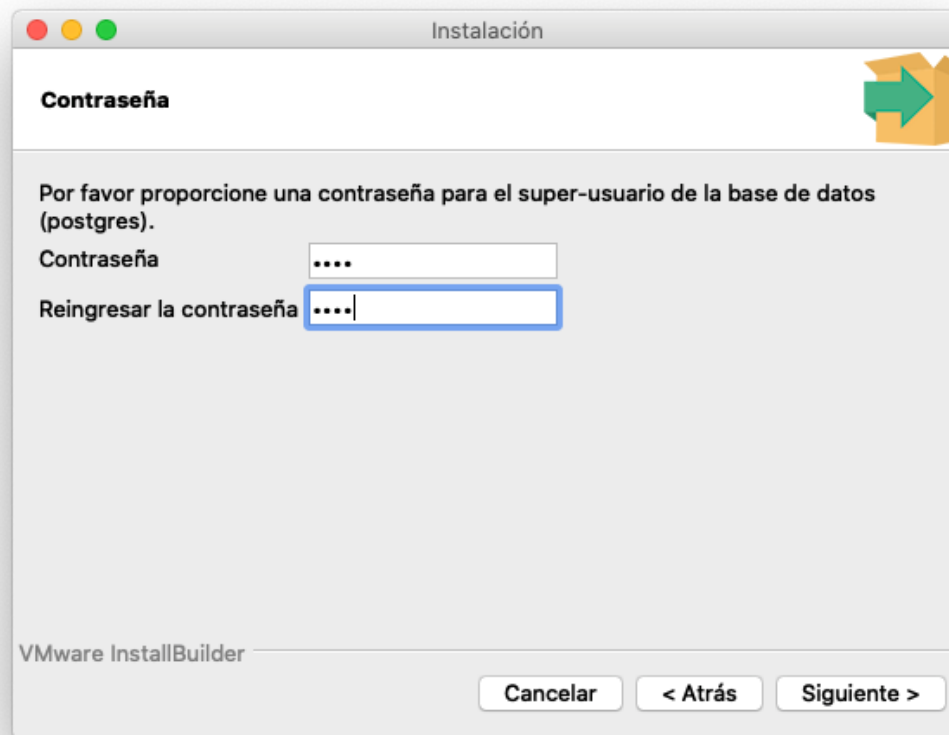
PostgreSQL

- Lo instalamos...



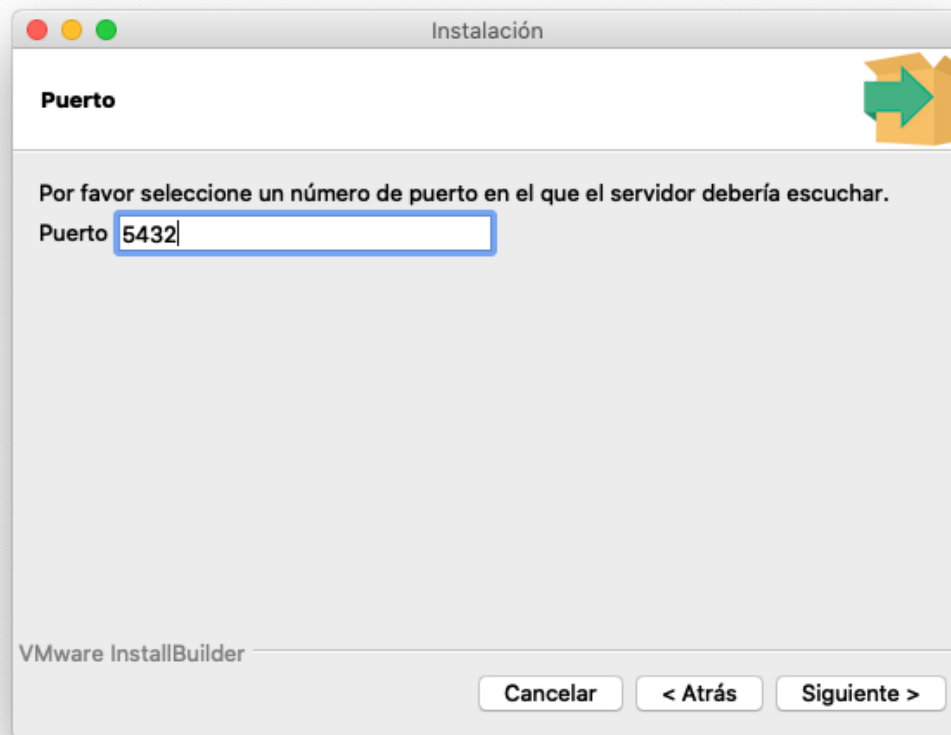
PostgreSQL

- ▶ Los que lo instaléis desde cero poned “**root**” como password para no tener que cambiarlo al revisar las prácticas



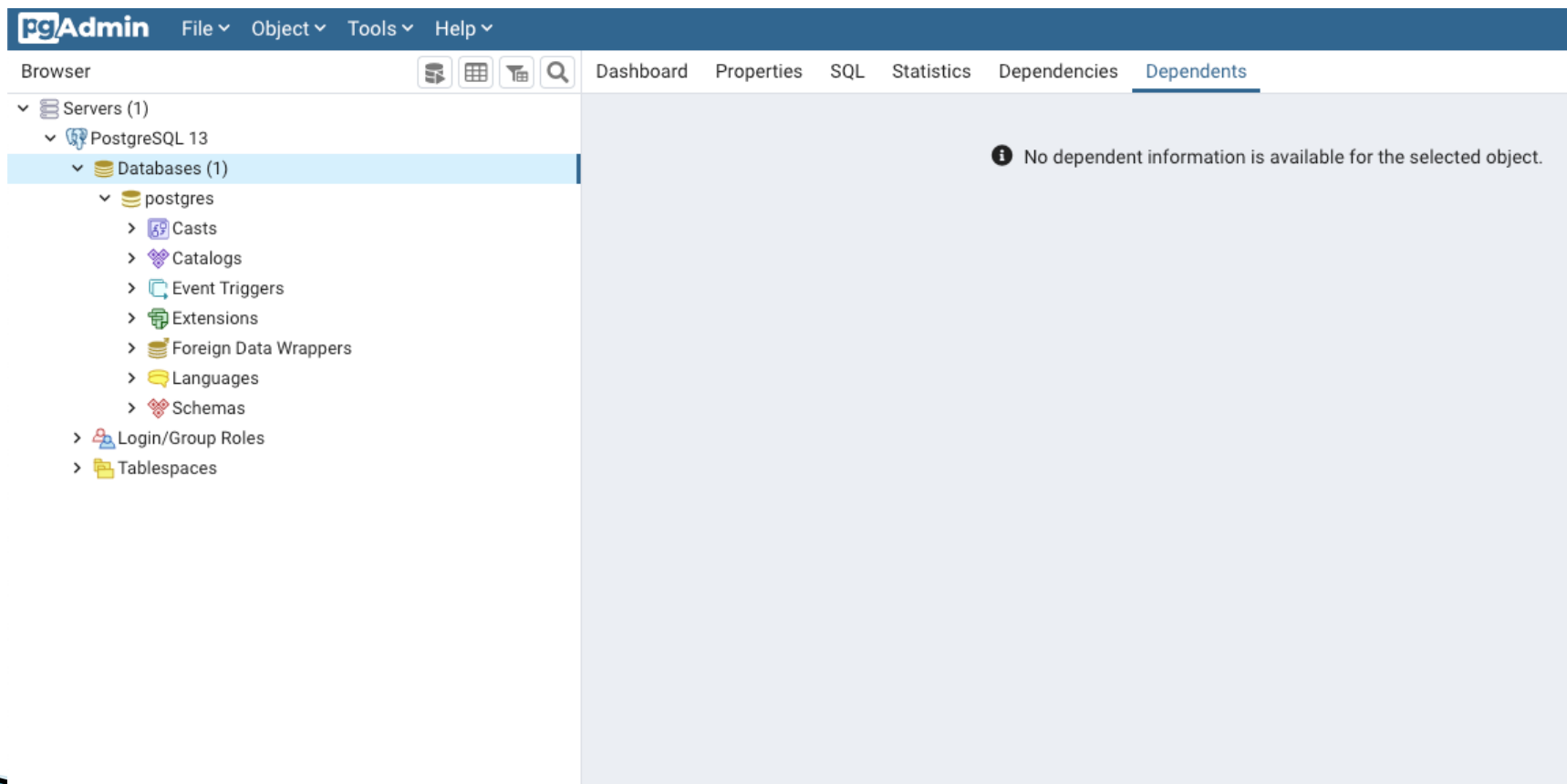
PostgreSQL

- ▶ Dejamos el puerto por defecto



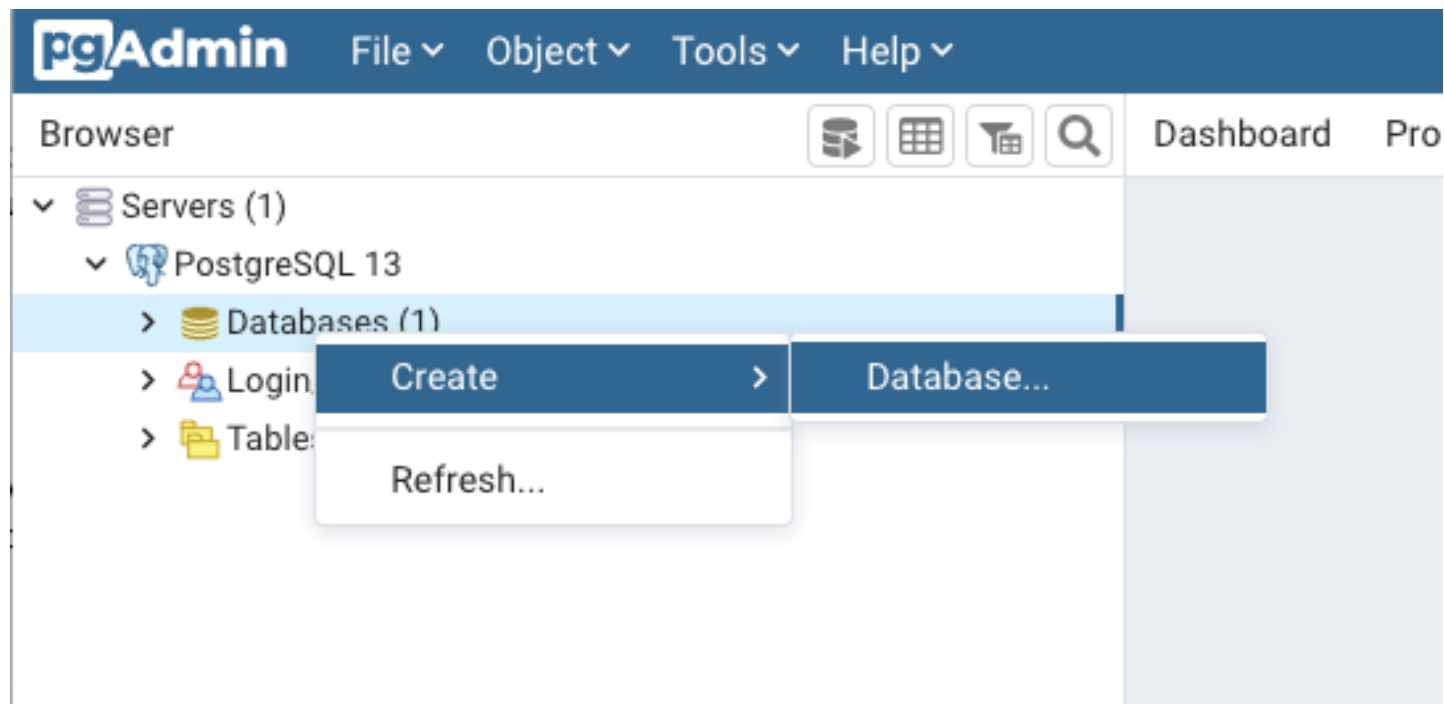
Herramienta de administración PostgreSQL

- ▶ PostgreSQL ya viene con la herramienta de administración pgAdmin4... la lanzamos...



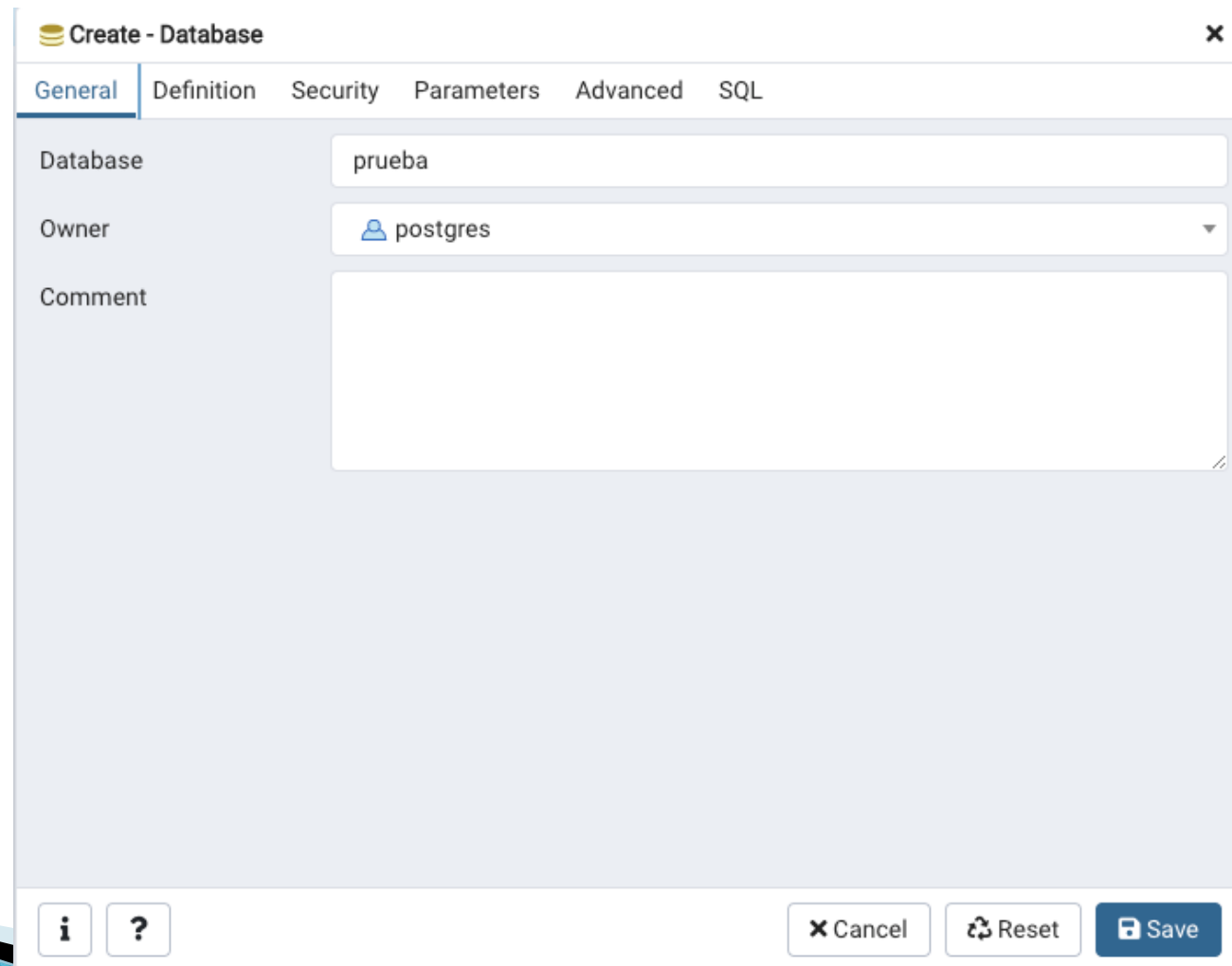
Herramienta de administración PostgreSQL

- ▶ Vamos a crear la base de datos "prueba" (click con el botón derecho sobre Databases – Create – Database)



Herramienta de administración PostgreSQL

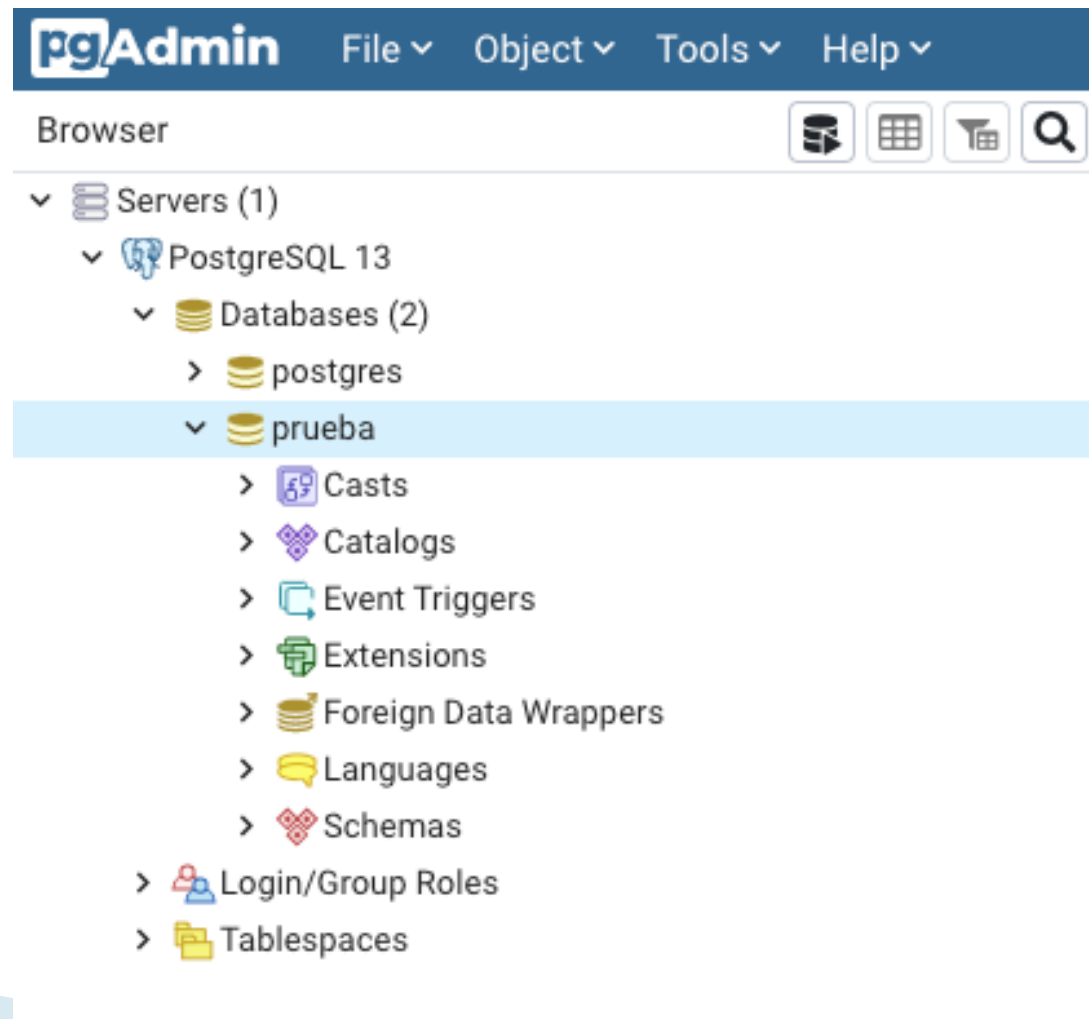
- ▶ Vamos a crear la base de datos "prueba"



The screenshot shows a 'Create - Database' dialog box with a close button (X) in the top right corner. The dialog has five tabs: 'General' (selected), 'Definition', 'Security', 'Parameters', 'Advanced', and 'SQL'. The 'General' tab contains three fields: 'Database' with the text 'prueba', 'Owner' with a dropdown menu showing 'postgres' and a user icon, and a large empty 'Comment' text area. At the bottom left are two small buttons with an 'i' and a '?'. At the bottom right are three buttons: 'Cancel' (with an X icon), 'Reset' (with a circular arrow icon), and 'Save' (in a dark blue box with a save icon).

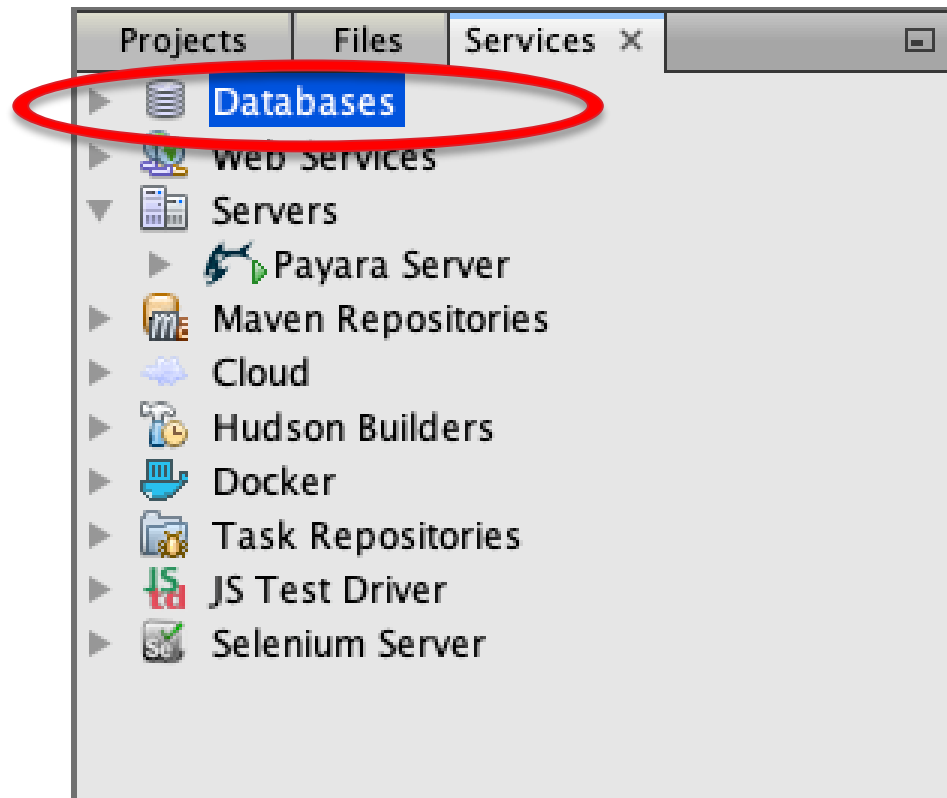
Herramienta de administración PostgreSQL

- ▶ Vamos a crear la base de datos "prueba"



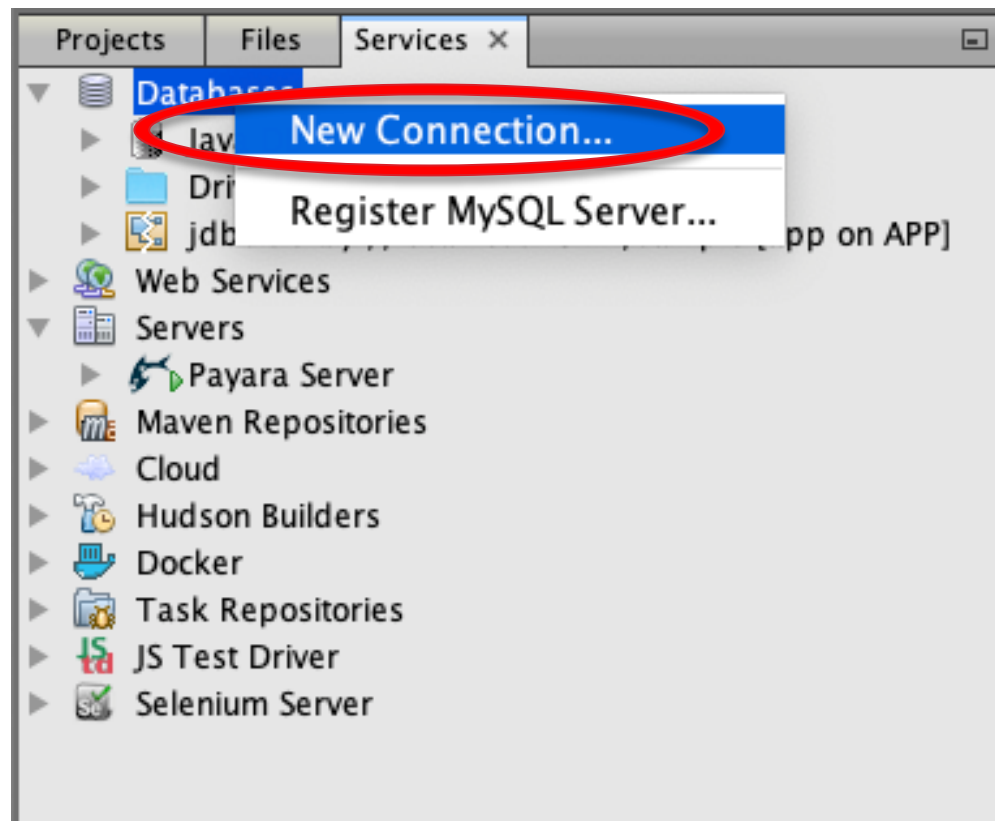
Configuración de PostgreSQL en NetBeans

- ▶ Nos vamos a “Databases” dentro de la pestaña “Services”



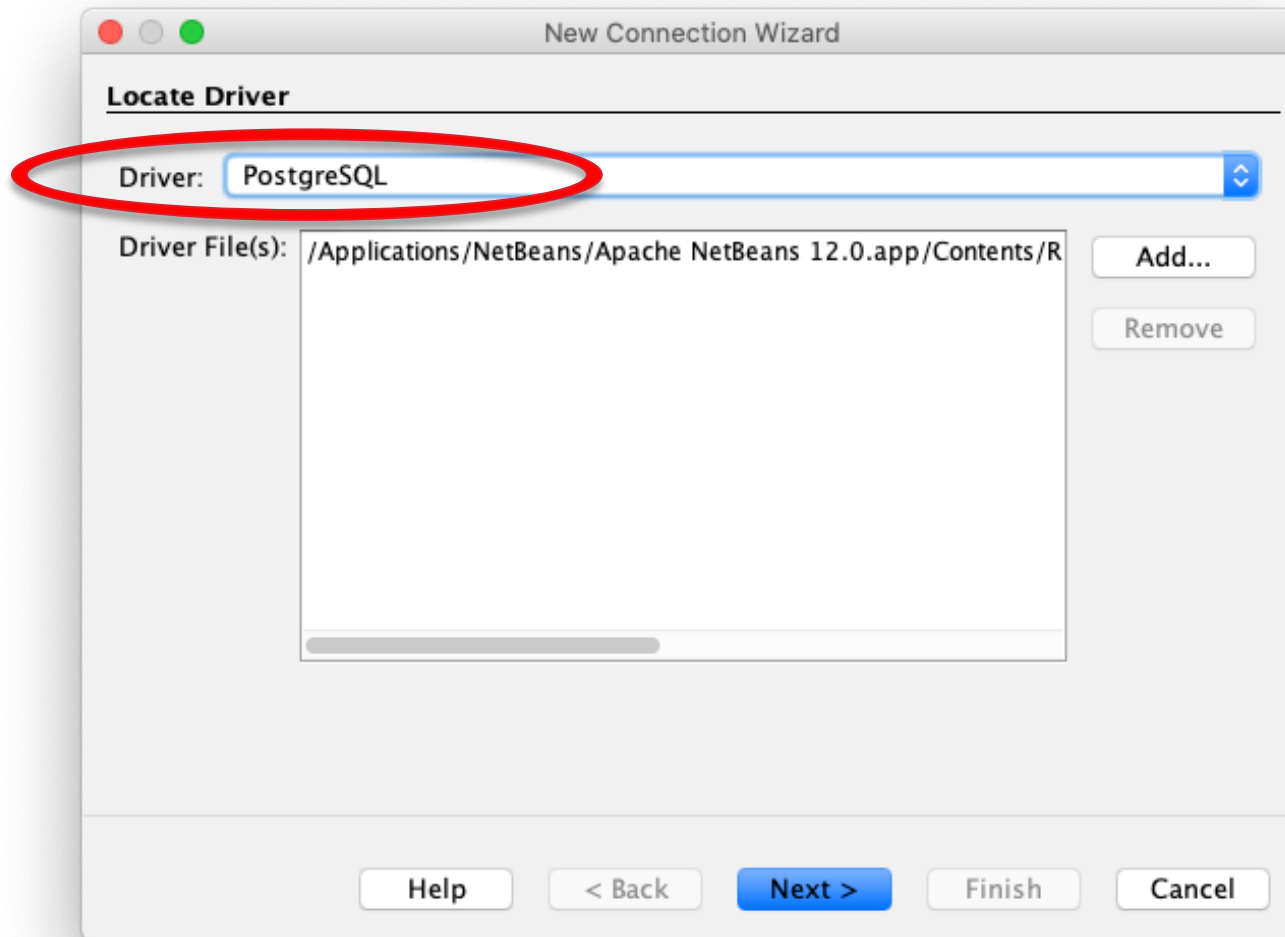
Configuración de PostgreSQL en NetBeans

- ▶ Click con el botón derecho en “Databases” – “New Connection”



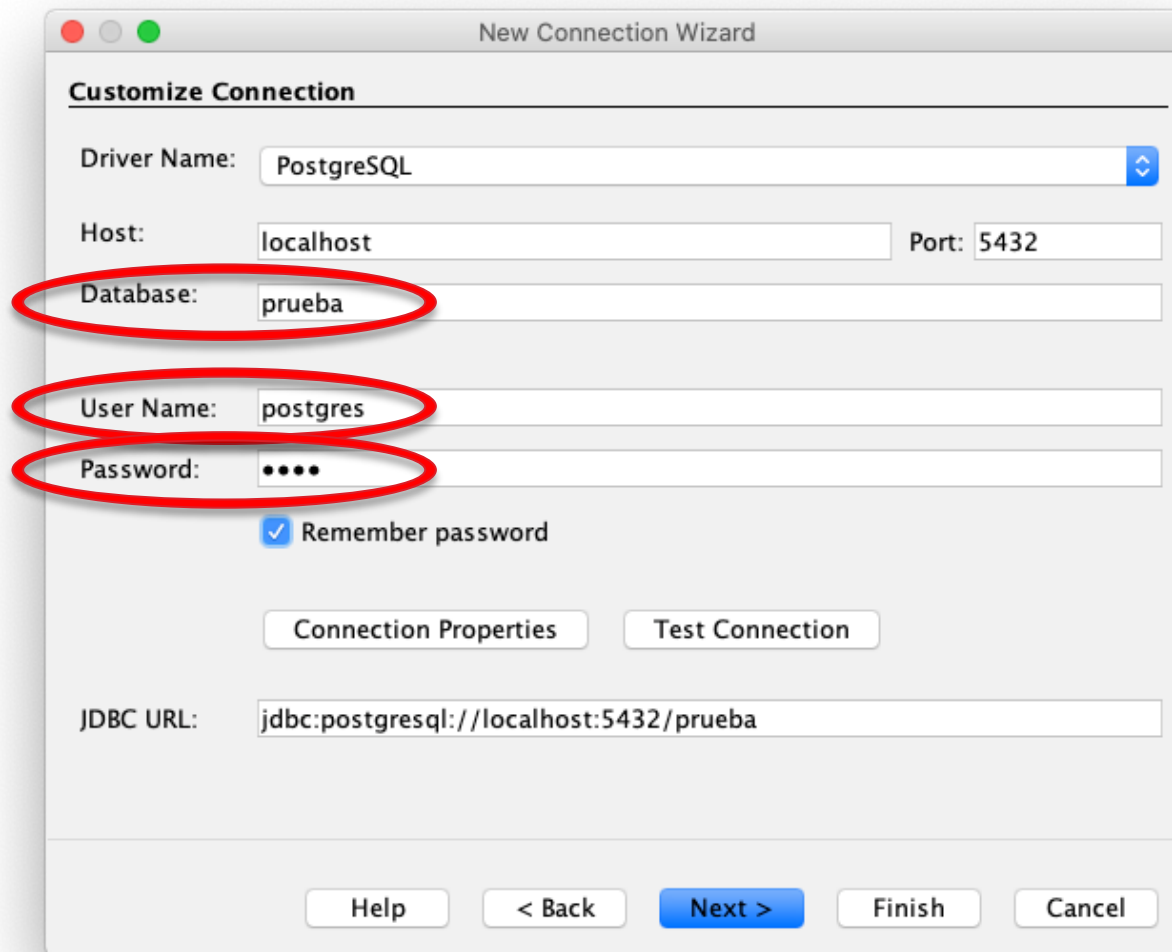
Configuración de PostgreSQL en NetBeans

- ▶ Seleccionamos el driver de PostgreSQL



Configuración de PostgreSQL en NetBeans

- ▶ Introducimos los datos de conexión a la base de datos



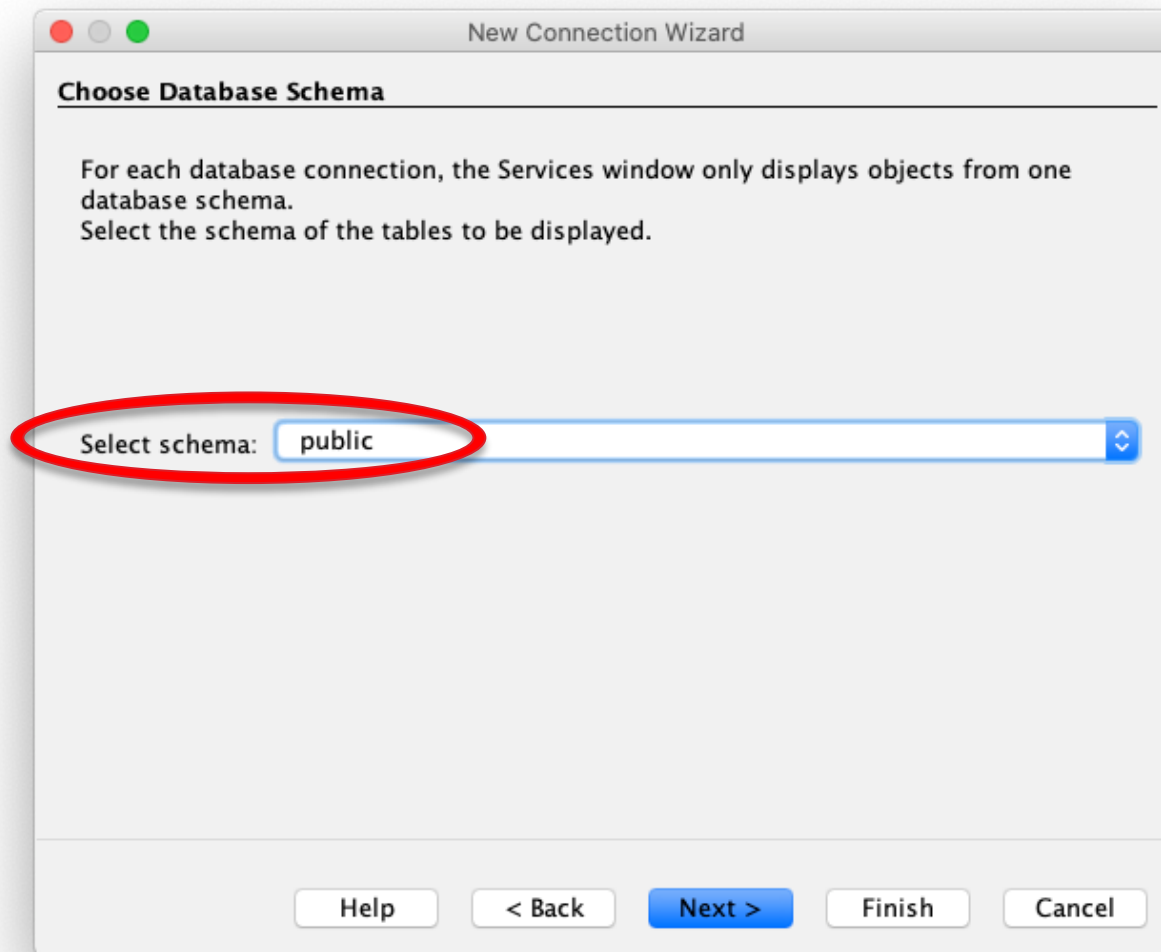
The screenshot shows the 'New Connection Wizard' dialog box in NetBeans, specifically the 'Customize Connection' step. The following fields are highlighted with red circles:

- Driver Name: PostgreSQL
- Host: localhost
- Port: 5432
- Database: prueba
- User Name: postgres
- Password:

Below these fields, the 'Remember password' checkbox is checked. There are two buttons: 'Connection Properties' and 'Test Connection'. At the bottom, the JDBC URL is displayed as 'jdbc:postgresql://localhost:5432/prueba'. The bottom navigation bar includes 'Help', '< Back', 'Next >', 'Finish', and 'Cancel' buttons.

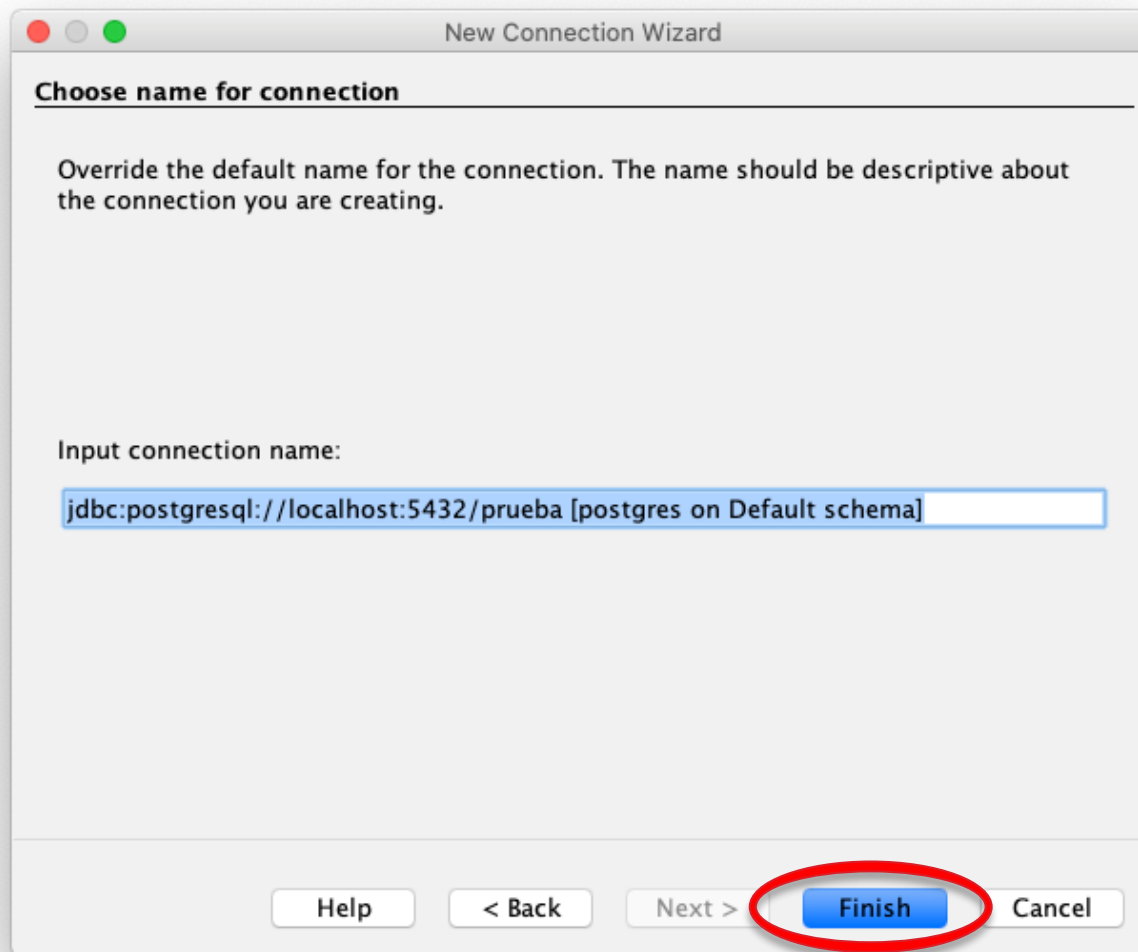
Configuración de PostgreSQL en NetBeans

- ▶ Esquema “public”



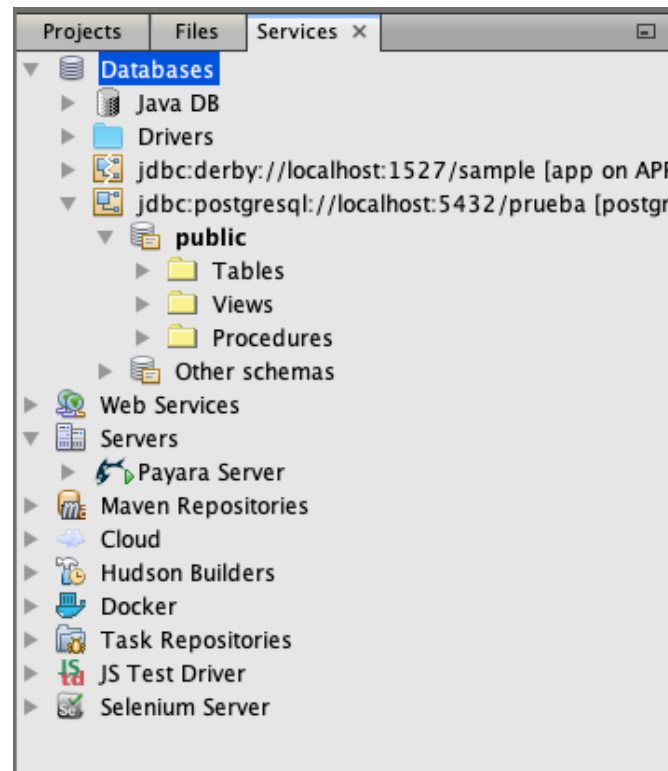
Configuración de PostgreSQL en NetBeans

- Y terminamos...

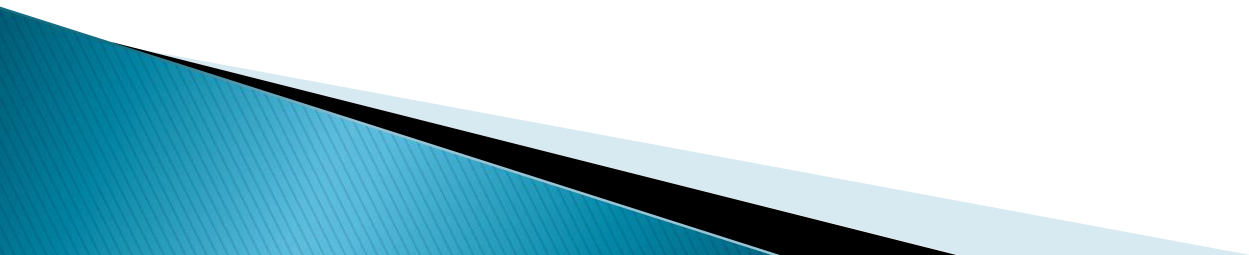


Crear una instancia de BD y conectarnos con ella

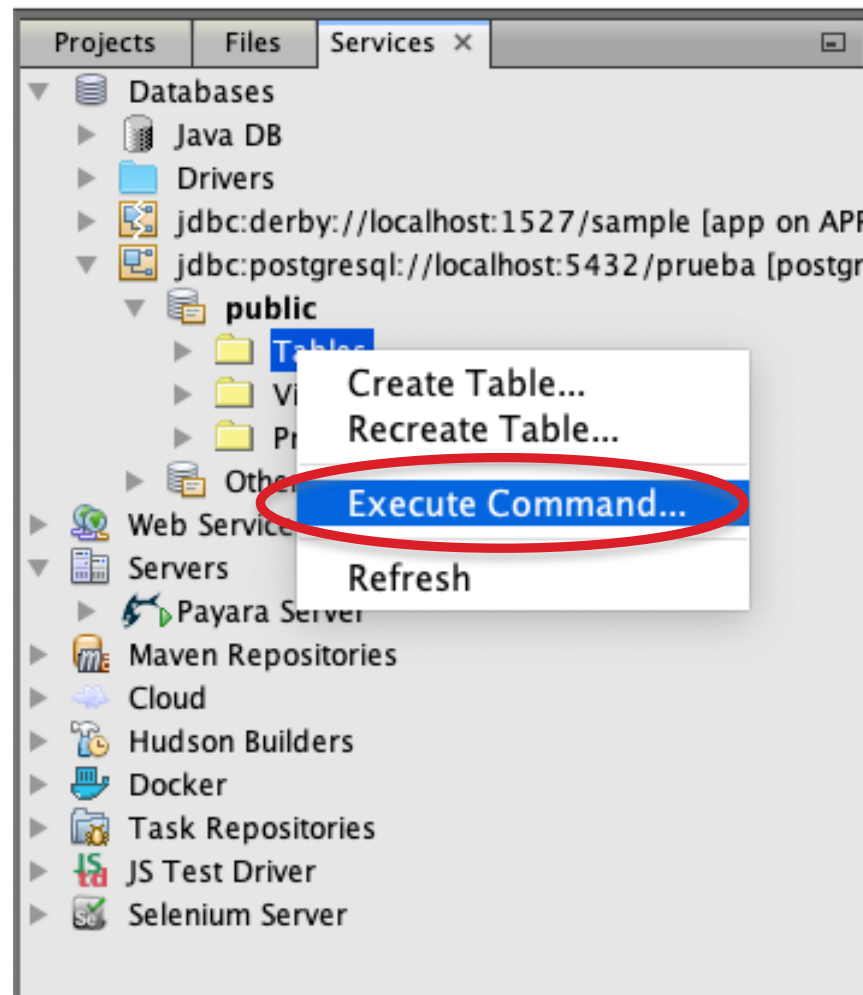
- ▶ Si todo ha ido bien, deberíais poder conectaros y ver vuestra base de datos "prueba"



Crear las tablas para la aplicación Factorial

- ▶ Podemos hacerlo directamente usando comandos SQL desde Netbeans (botón derecho sobre “Tables” – “Execute Command”)
 - ▶ Podemos hacerlo usando el editor gráfico (también en Netbeans)
 - ▶ Podemos hacerlo directamente con la herramienta de administración (en este caso tendremos que pulsar en “Refresh” para ver los cambios)
- 

Crear las tablas para la aplicación Factorial




Crear las tablas para la aplicación Factorial

- ▶ Creamos una tabla, con 3 campos:

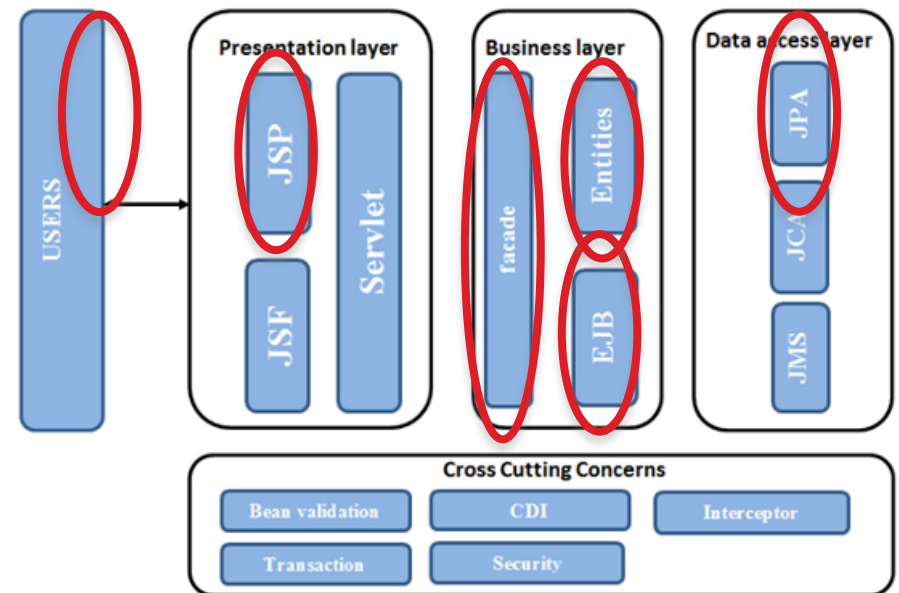
- Id: identificador
- Base: base introducida por el usuario
- Usuario: nombre del usuario

```
CREATE TABLE factorial(  
  id integer GENERATED ALWAYS AS IDENTITY PRIMARY KEY,  
  base TEXT NULL,  
  usuario TEXT NULL);
```

- ▶ Escribimos la sentencia SQL de creación y la ejecutamos (pulsando en )
- ▶ Debería ejecutarse sin errores y habiendo creado la tabla factorial en la base de datos

Primera aplicación Java EE

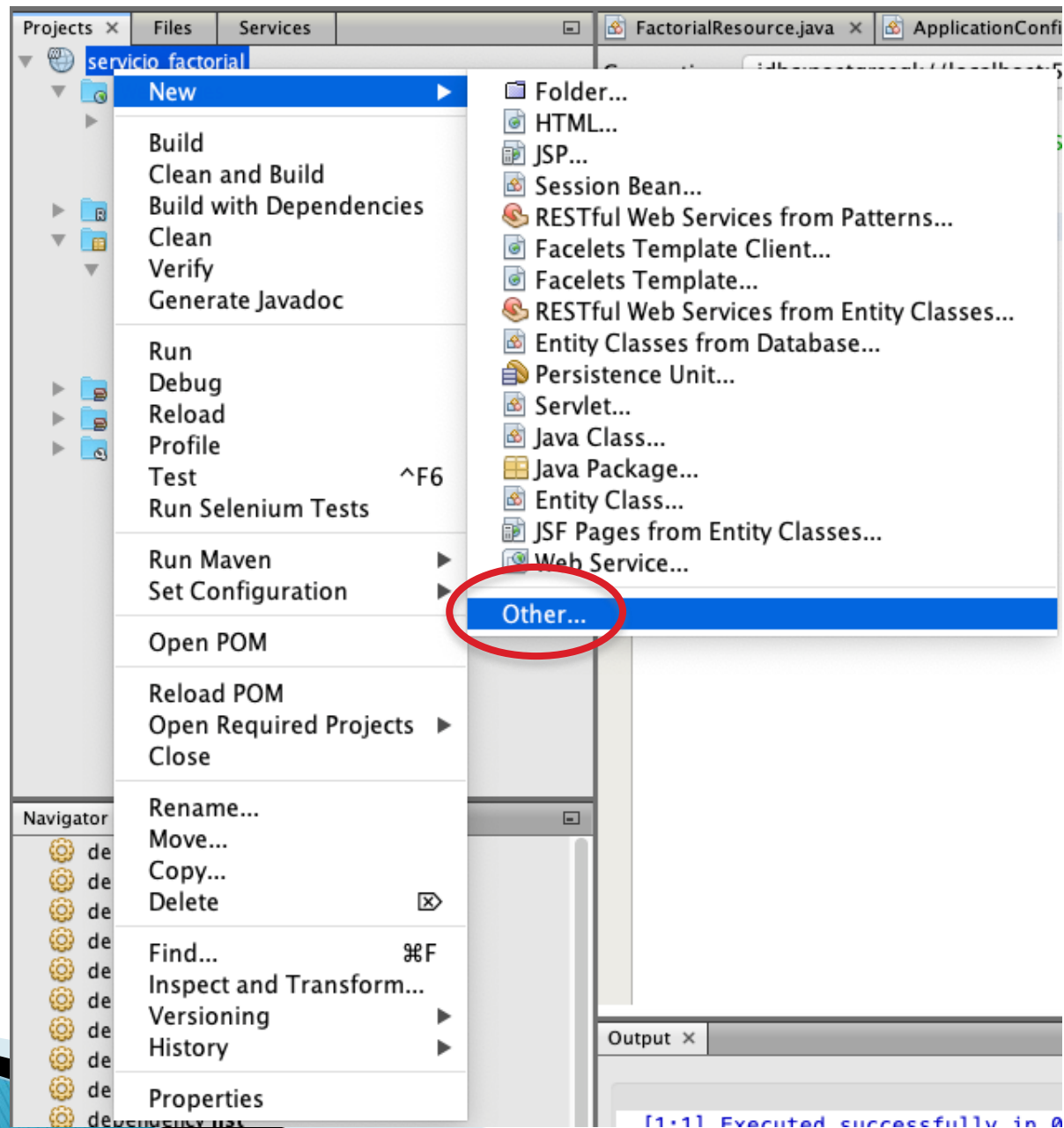
- ▶ Vamos a utilizar EclipseLink para implementar los componentes lógicos de acceso a datos
 - EclipseLink implementa JPA
- ▶ Primero crearemos todos los componentes de la base de datos necesarios, después modificaremos nuestro fichero “prueba_ajax.html” para que envíe la información a guardar a un servlet, por último, el servlet usará JPA para escribir la información en la base de datos.



Crear los ficheros de entidad

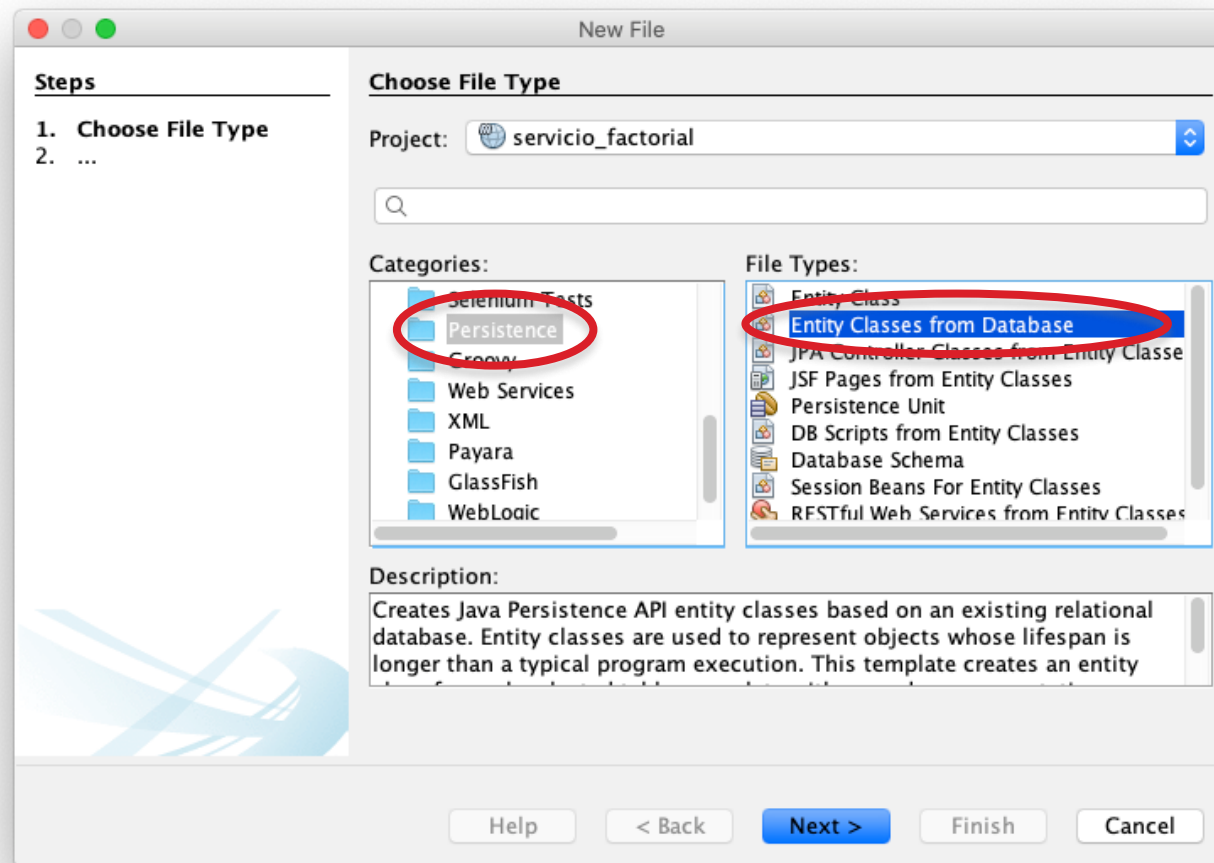
- ▶ Creamos los ficheros de entidad de manera automática con Netbeans y EclipseLink
 - Los ficheros de entidad contendrán la información a almacenar y los métodos básicos.

Crear los ficheros de entidad

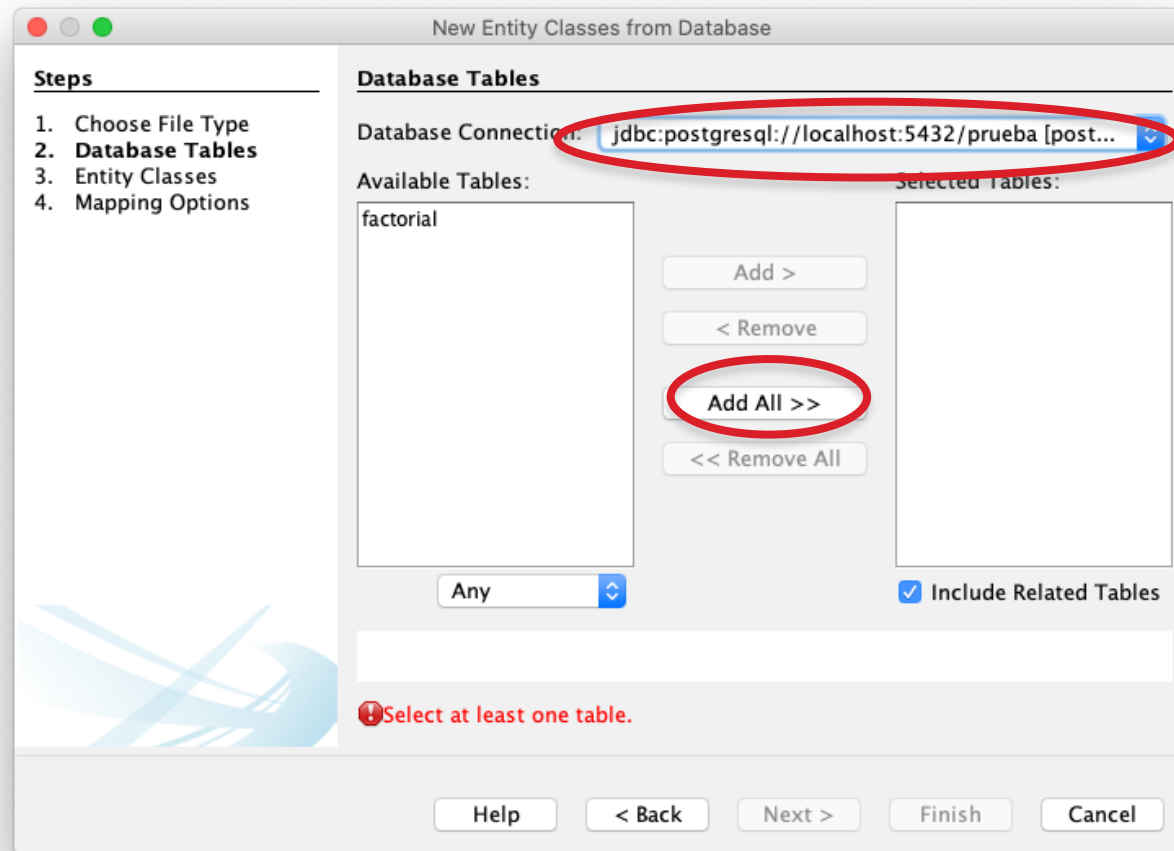


Crear los ficheros de entidad

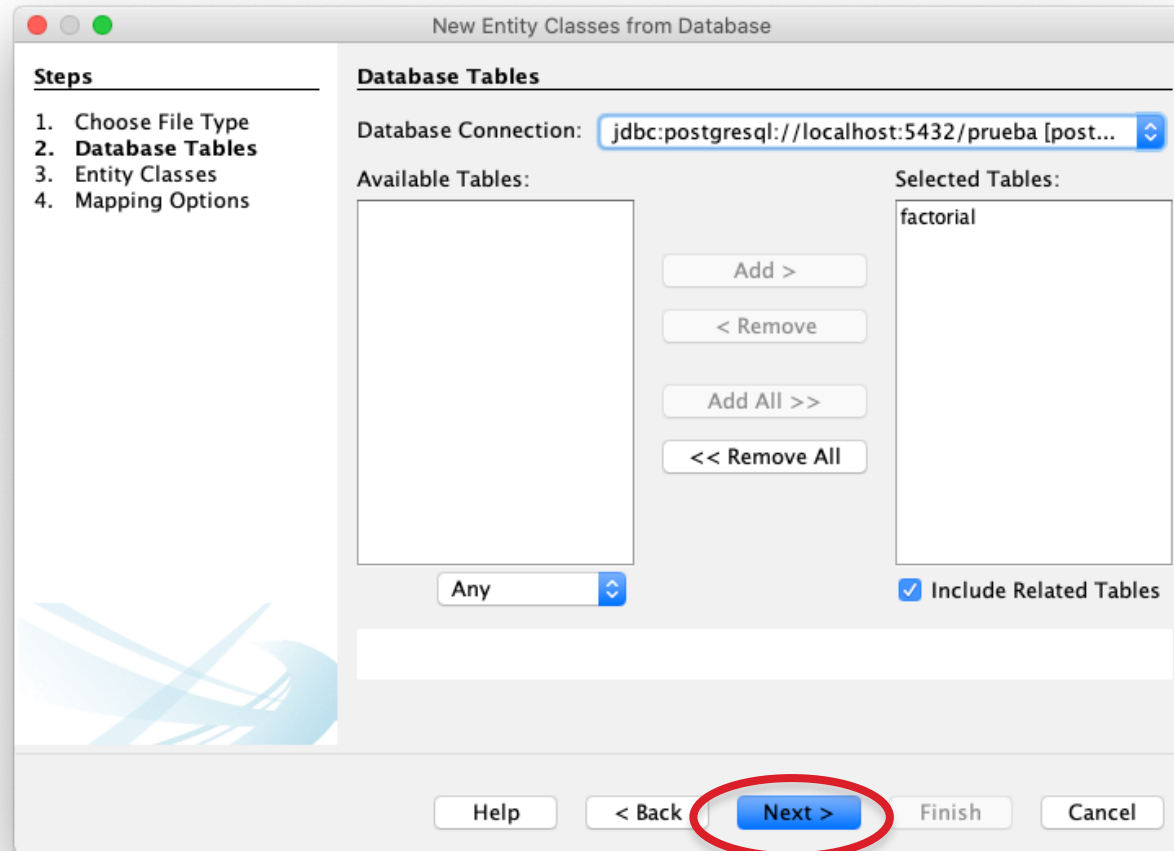
- ▶ Creamos los ficheros entidad desde la base de datos ya existente



Crear los ficheros de entidad



Crear los ficheros de entidad



Crear los ficheros de entidad

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
	factorial	Factorial	New

Project:

Location:

Package:

☒ Generate Named Query Annotations for Persistent Fields

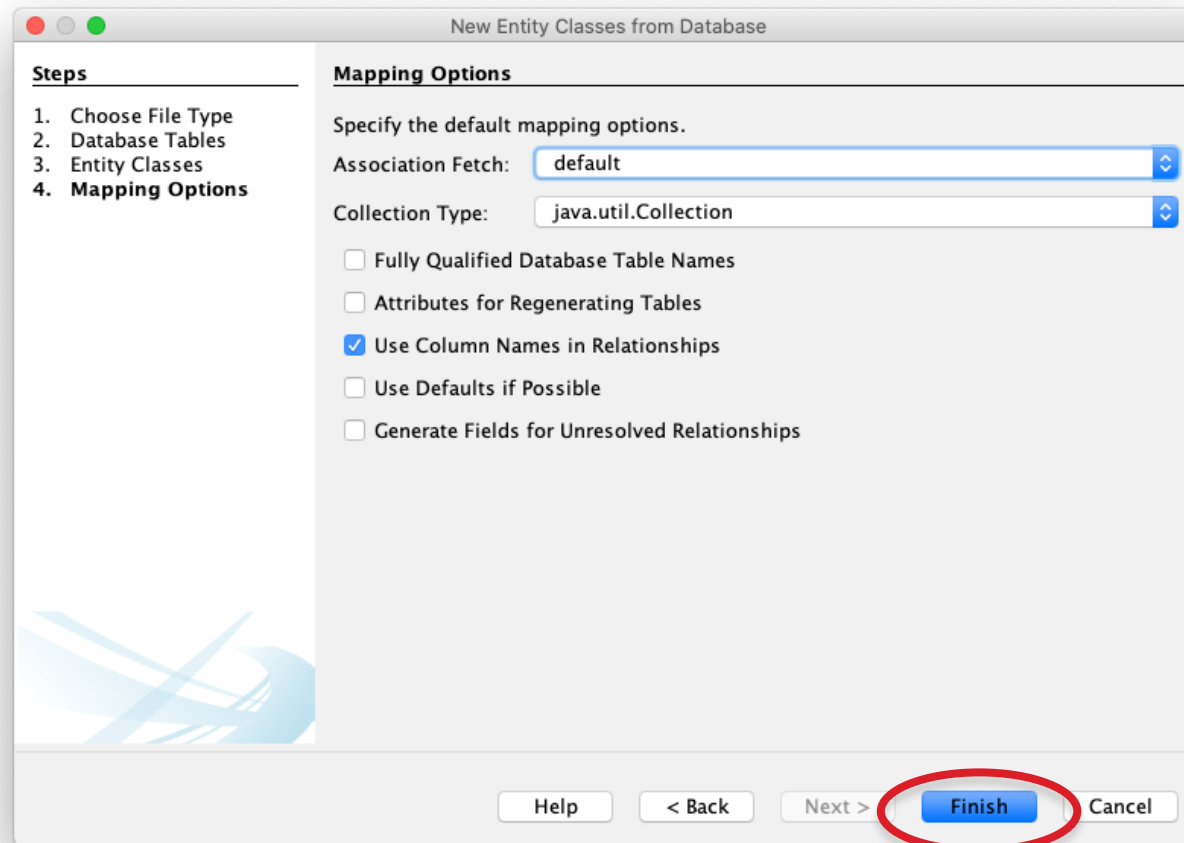
☒ Generate JAXB Annotations

☐ Generate MappedSuperclasses instead of Entities

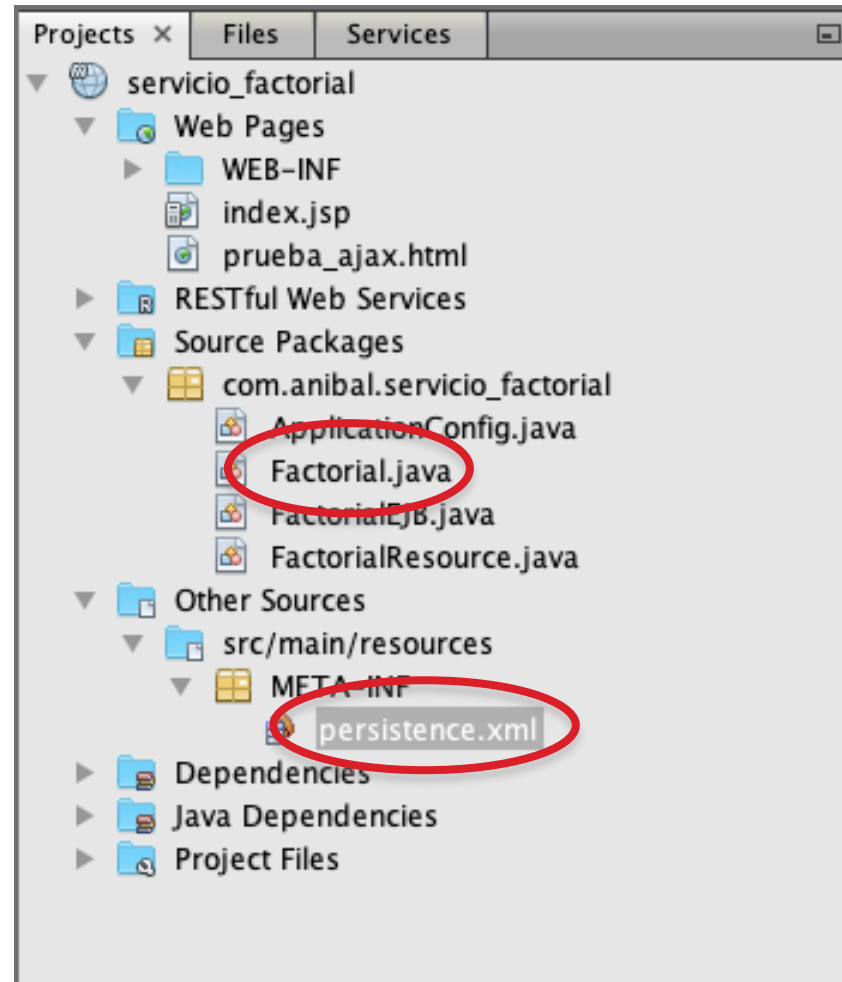
☒ Create Persistence Unit

Help < Back **Next >** Finish Cancel

Crear los ficheros de entidad



Crear los ficheros de entidad



Crear los ficheros de entidad

► Fichero de entidad

```
@Entity
@Table(name = "factorial")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Factorial.findAll", query = "SELECT f FROM Factorial f"),
    @NamedQuery(name = "Factorial.findById", query = "SELECT f FROM Factorial f WHERE f.id = :id"),
    @NamedQuery(name = "Factorial.findByBase", query = "SELECT f FROM Factorial f WHERE f.base = :base"),
    @NamedQuery(name = "Factorial.findByUsuario", query = "SELECT f FROM Factorial f WHERE f.usuario = :usuario"))
public class Factorial implements Serializable {

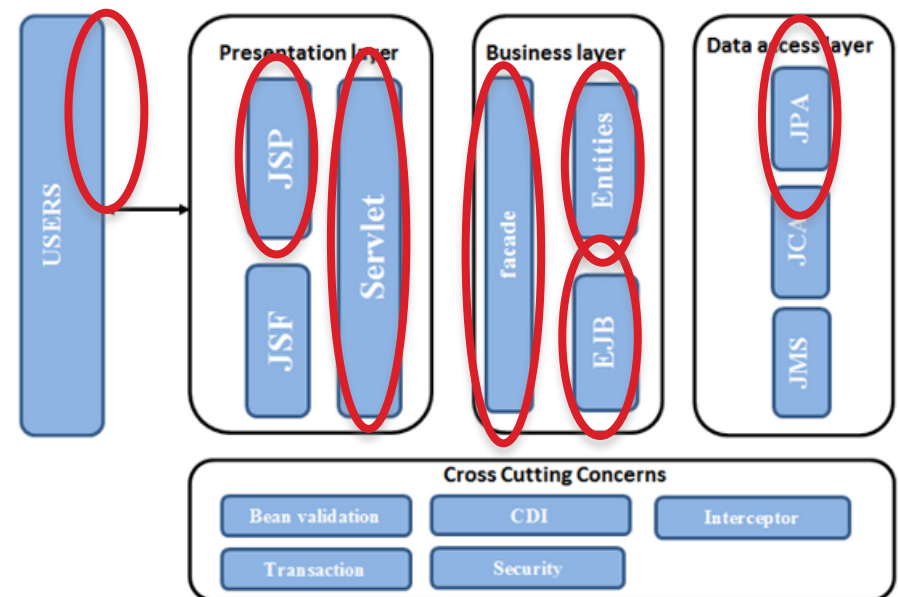
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id")
    private Integer id;
    @Size(max = 2147483647)
    @Column(name = "base")
    private String base;
    @Size(max = 2147483647)
    @Column(name = "usuario")
    private String usuario;

    public Factorial() {
    }

    public Factorial(Integer id) {
        this.id = id;
    }
}
```

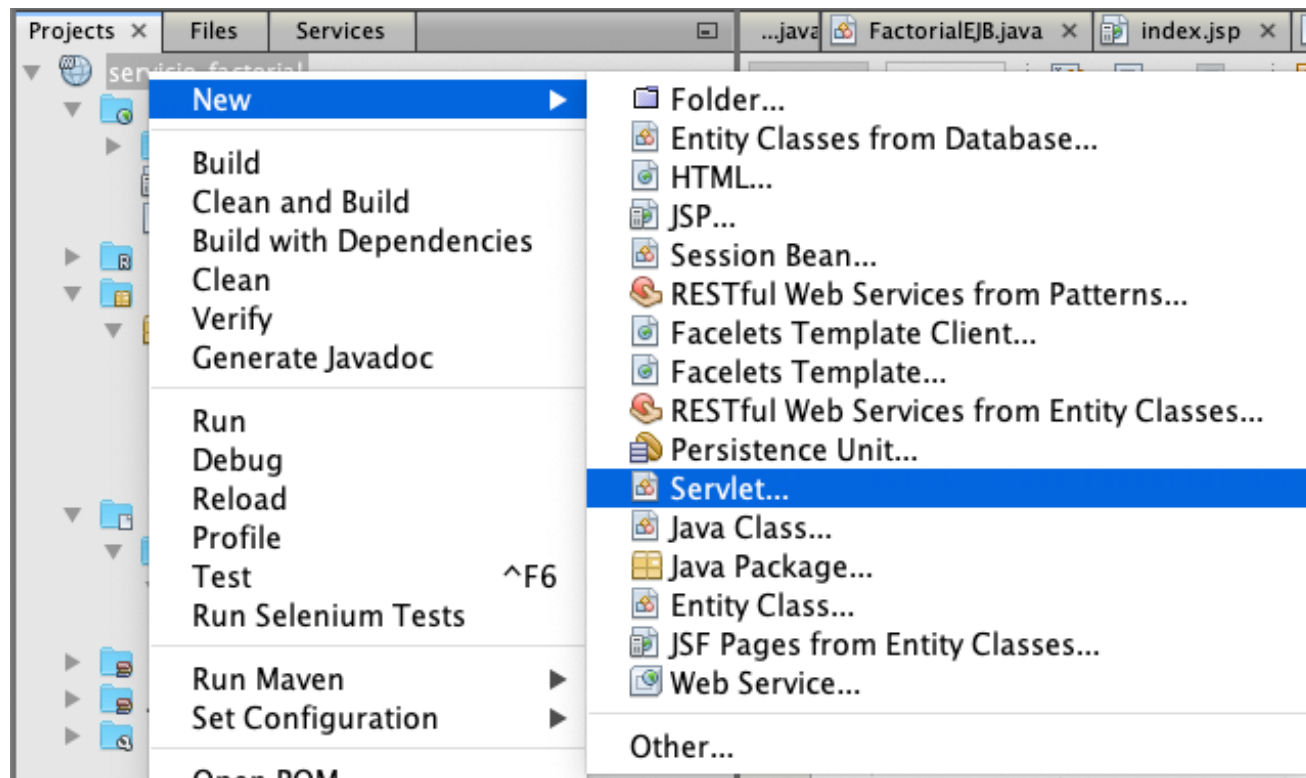
Añadir servlet y modificar prueba_ajax

- ▶ Sólo nos queda añadir un servlet que reciba la orden de guardar los datos por parte del usuario y lo almacene en la base de datos.
- ▶ Y modificar el fichero “prueba_ajax.html” para lanzar el servlet con la información necesaria.



Añadir servlet y modificar prueba_ajax

- ▶ Creamos el servlet

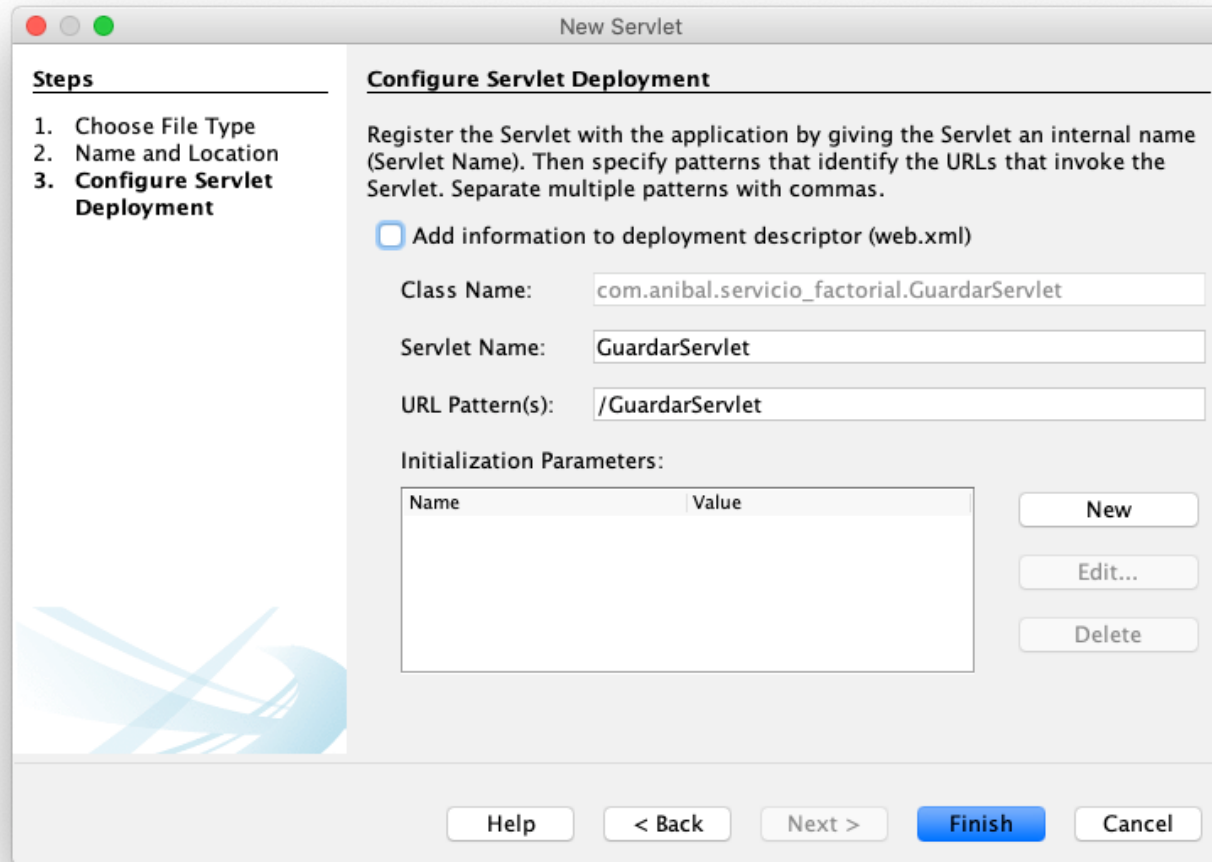


Añadir servlet y modificar prueba_ajax

The screenshot shows the 'New Servlet' dialog box with the following details:

- Steps:**
 1. Choose File Type
 2. **Name and Location**
 3. Configure Servlet Deployment
- Name and Location:**
 - Class Name:** GuardarServlet
 - Project:** servicio_factorial
 - Location:** Source Packages
 - Package:** com.anibal.servicio_factorial
 - Created File:** .rc/main/java/com/anibal/servicio_factorial/GuardarServlet.java
- Buttons:** Help, < Back, Next > (highlighted), Finish, Cancel

Añadir servlet y modificar prueba_ajax



The screenshot shows a 'New Servlet' dialog box with a 'Steps' sidebar on the left and a 'Configure Servlet Deployment' main area on the right. The 'Steps' sidebar lists three steps: '1. Choose File Type', '2. Name and Location', and '3. Configure Servlet Deployment', with the third step being the active one. The 'Configure Servlet Deployment' section contains instructions to register the servlet with an internal name and specify URL patterns. It includes a checkbox for 'Add information to deployment descriptor (web.xml)', which is currently unchecked. Below this are three text input fields: 'Class Name' with the value 'com.anibal.servicio_factorial.GuardarServlet', 'Servlet Name' with the value 'GuardarServlet', and 'URL Pattern(s)' with the value '/GuardarServlet'. There is also an 'Initialization Parameters' section with a table for Name and Value, and buttons for 'New', 'Edit...', and 'Delete'. At the bottom of the dialog are buttons for 'Help', '< Back', 'Next >', 'Finish' (highlighted in blue), and 'Cancel'.

Steps

1. Choose File Type
2. Name and Location
3. **Configure Servlet Deployment**

Configure Servlet Deployment

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

☐ Add information to deployment descriptor (web.xml)

Class Name:

Servlet Name:

URL Pattern(s):

Initialization Parameters:

Name	Value
------	-------

New Edit... Delete

Help < Back Next > **Finish** Cancel

Añadir servlet y modificar prueba_ajax

- ▶ Actualizamos el fichero “prueba_ajax.html” para añadir la opción de lanzar el servlet con la información necesaria.

```
<body>
  <form action="GuardarServlet">
    <h2>Calcular factorial</h2>
    Número:<input type="text" name="base" id="base"/>
    <button type="button" id="calcularBtn">Calcular</button>
    <div id="resultado">
      Resultado: <span></span>
    </div>
    <script type="text/javascript">
      jQuery("#calcularBtn").click(function(){
        var base = jQuery("#base").val();
        jQuery.get("http://localhost:8080/servicio_factorial/resources/factorial",{
          base:base
        },function(resultado){
          jQuery("#resultado span").text(resultado)
        });
      });
    </script>
    Usuario:<input type="text" name="usuario" id="usuario"/>
    <input type="submit" value="Guardar"/>
  </form>
</body>
```

Añadir servlet y modificar prueba_ajax

- ▶ Ahora actualizamos la información que tiene que procesar el servlet (introducir la información en la base de datos) – lo ponemos en el `processRequest`

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

```
String base = request.getParameter("base");  
String usuario = request.getParameter("usuario");
```

```
Factorial fact = new Factorial();  
fact.setBase(base);  
fact.setUsuario(usuario);
```

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("com.anibal_servicio_factorial_war_v1PU");  
EntityManager em = emf.createEntityManager();  
try {
```

```
    em.getTransaction().begin();  
    em.persist(fact);  
    em.getTransaction().commit();
```

```
} catch (Exception e) {  
    e.printStackTrace();  
}  
} finally {  
    em.close();  
}
```

```
try (PrintWriter out = response.getWriter()) {  
    ...  
}
```

Obtenemos los parámetros de entrada, que nos llegarán de la página que llama al Servlet (lo vemos más adelante)

Cuidado: nos conectamos a una “Unidad de Persistencia” (Persistence Unit, PU)... aquí tendréis que poner el nombre de vuestra PU según lo que tengáis en `persistence.xml`

Enviamos los datos a la base de datos

Cambiad también lo que viene a continuación, que es el código HTML que genera si queréis algo más “informativo” (ejemplo en la siguiente transparencia)

Y añadid los import que falten!!!

Añadir servlet y modificar prueba_ajax

- ▶ Cambiad también el HTML que genera si queréis algo más “informativo”

```
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet GuardarServlet</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1 style='text-align:center;'>Servlet Guardar Datos en la Base de Datos</h1>");
out.println("<p style='text-align:center;'>Datos guardados correctamente en la base de datos!!</p><br>");
out.println("<img style='display:block; margin:auto;'src='https://www.alfabetajuega.com/abj_public_files/multimedia/imagenes/201310/53112.ho mer_yuhu_simpsons.jpg' alt='Homer_Feliz'>");
out.println("<br><br>");
out.println("<a href='/servicio_factorial/prueba_ajax.html'><p style='text-align:center;'>Volver a calcular factorial.</p></a>");
out.println("</body>");
out.println("</html>");
```

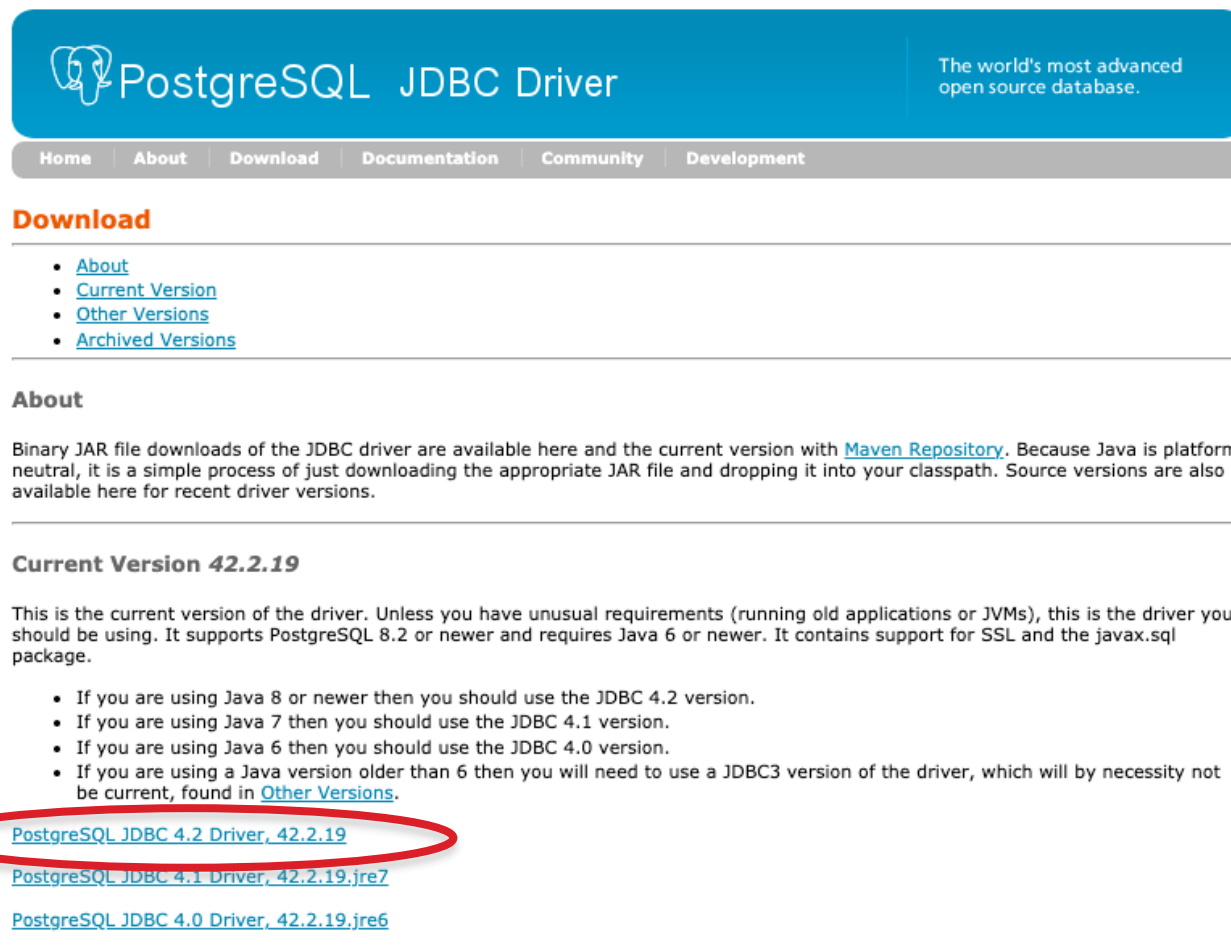
Importar el driver JDBC

- ▶ Nos falta un último paso: importar el driver para PostgreSQL
- ▶ Se podría hacer con la opción “Add Dependency”, pero como la vamos a usar en más proyectos, lo mejor es copiar el driver al directorio de librerías de Payara y así nos olvidamos

Importar el driver JDBC

- Nos descargamos la última versión del driver JDBC de:

<https://jdbc.postgresql.org/download.html>



The screenshot shows the PostgreSQL JDBC Driver download page. The header is blue with the PostgreSQL logo and the text 'PostgreSQL JDBC Driver'. To the right, it says 'The world's most advanced open source database.' Below the header is a navigation bar with links: Home, About, Download, Documentation, Community, and Development. The 'Download' section is highlighted in orange. It contains a list of links: About, Current Version, Other Versions, and Archived Versions. The 'About' section follows, explaining that binary JAR file downloads are available here and that the current version is available on the Maven Repository. The 'Current Version 42.2.19' section states that this is the current version of the driver, supporting PostgreSQL 8.2 or newer and requiring Java 6 or newer. It also contains a list of instructions for different Java versions. At the bottom, there are three links: 'PostgreSQL JDBC 4.2 Driver, 42.2.19' (circled in red), 'PostgreSQL JDBC 4.1 Driver, 42.2.19.jre7', and 'PostgreSQL JDBC 4.0 Driver, 42.2.19.jre6'.

PostgreSQL JDBC Driver

The world's most advanced open source database.

Home About Download Documentation Community Development

Download

- [About](#)
- [Current Version](#)
- [Other Versions](#)
- [Archived Versions](#)

About

Binary JAR file downloads of the JDBC driver are available here and the current version with [Maven Repository](#). Because Java is platform neutral, it is a simple process of just downloading the appropriate JAR file and dropping it into your classpath. Source versions are also available here for recent driver versions.

Current Version 42.2.19

This is the current version of the driver. Unless you have unusual requirements (running old applications or JVMs), this is the driver you should be using. It supports PostgreSQL 8.2 or newer and requires Java 6 or newer. It contains support for SSL and the javax.sql package.

- If you are using Java 8 or newer then you should use the JDBC 4.2 version.
- If you are using Java 7 then you should use the JDBC 4.1 version.
- If you are using Java 6 then you should use the JDBC 4.0 version.
- If you are using a Java version older than 6 then you will need to use a JDBC3 version of the driver, which will by necessity not be current, found in [Other Versions](#).

[PostgreSQL JDBC 4.2 Driver, 42.2.19](#)

[PostgreSQL JDBC 4.1 Driver, 42.2.19.jre7](#)

[PostgreSQL JDBC 4.0 Driver, 42.2.19.jre6](#)

Importar el driver JDBC

- ▶ Y lo copiáis en la siguiente ruta de vuestro servidor payara...

.../Payara_Server/glassfish/lib

- ▶ Después reiniciad el servidor Payara

Ejecutamos...

Calcular factorial

Número:

Resultado:

Usuario:

Cuidado!!

Puede ser necesario desplegar desde cero, limpiando la versión anterior en el despliegue – para ello:

- Botón derecho sobre servicio_factorial y hacer “Clean and Build”



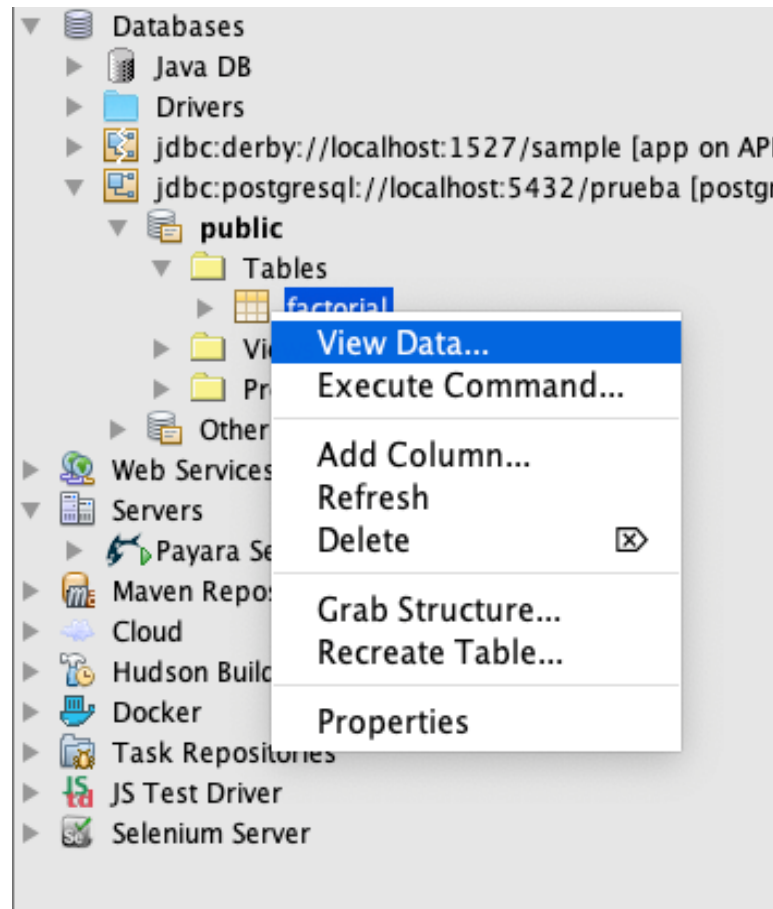
Servlet Guardar Datos en la Base de Datos

Datos guardados correctamente en la base de datos!!

[Volver a calcular factorial.](#)

Ejecutamos y comprobamos en la base de datos si se han guardado los datos...

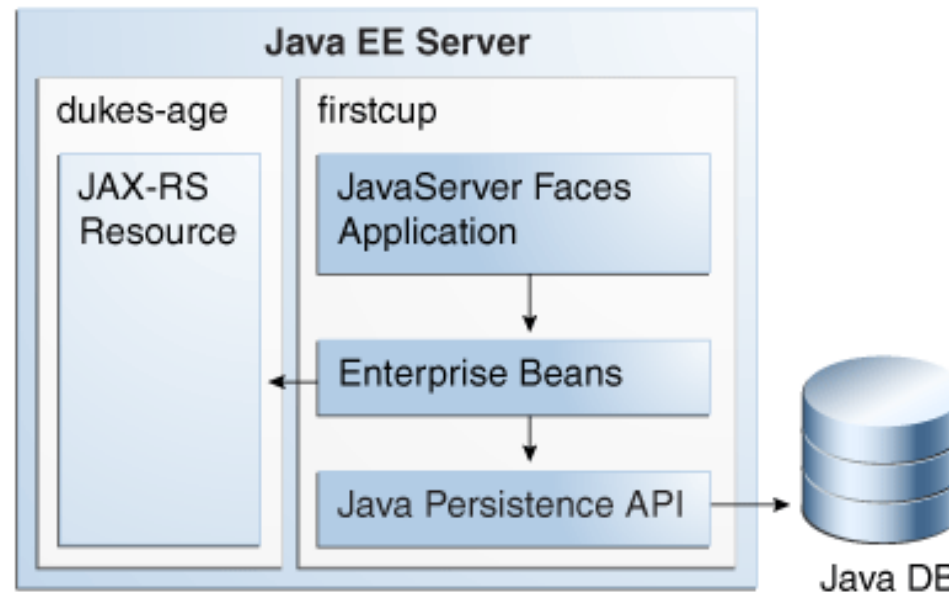
- ▶ Lo podéis hacer directamente en Netbeans o mediante la herramienta de administración de postgres



Siguientes pasos...

- ▶ Your First Cup example (para ver un primer ejemplo de JavaServer Faces, JPA, Enterprise Beans, Entity Beans):

<https://javaee.github.io/firstcup/>



- ▶ Manejo de Frameworks, etc...
- ▶ **El próximo miércoles comenzaremos con JSFs**

Primera práctica evaluable de PSE

- ▶ Esta primera práctica evaluable consistirá en la entrega de la aplicación Factorial que hemos visto en clase
- ▶ La entrega tiene que ser completa, e incluir todos los artefactos presentados durante las dos últimas clases (ficheros html, jsp, servlet, servicio REST, lógica de conexión a la base de datos, etc...)
- ▶ Se deberá enviar en un fichero comprimido la carpeta completa del proyecto creado
- ▶ Es suficiente con que lo envíe un miembro de cada grupo
- ▶ La fecha límite para la entrega es el próximo jueves, 31 de marzo, a las 23:55 horas.

¿PREGUNTAS?