

Arquitectura y Diseño de Sistemas WEB y C/S

Tema: Servlets



Grupo 6

Integrantes

Daniel Ferreiro Rodríguez

Bianca Marinela Lupu

Carlos Javier Hellín Asensio

Francisco Calles Esteban

Darius Dumitras Tamas

Grado de Ingeniería Informática

Curso: 2021-2022



Contenido

Introducción.....	3
Aspectos importantes que conocer	3
Ventajas	3
Ciclo de vida de un servlet	3
Práctica de Servlets	4
Hola Mundo	4
Cambiar el navegador WEB por defecto.....	9
Acceso básico a formularios	11
Calculadora	14
Primitiva	18
Acceso a ficheros de texto plano	21
Acceso a datos	23
Práctica de Java Server Pages	28
Acceso a datos (I)	28
Creación de una BBDD en Microsoft Access (EXTRA)	28
Creación de la Base de Datos en Derby	30
Acceso a datos (II)	33
Creación de la base de datos	33
Creación del JSP	41
Exportación de la base de datos	43
Sesiones	45
Conclusiones	47



Introducción

En esta práctica tiene como objetivo el manejo de la API de los Servlets y de la tecnología Java Server Pages del lenguaje de programación Java.

Aspectos importantes que conocer

Servlet: Clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor web. El uso más común de los *servlets* es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

Ventajas

- Integración fuerte entre servlets y servidor: permite colaboración entre ambos.
- Extensibilidad y flexibilidad.
- Comunidad grande de desarrolladores.
- Seguridad y elegancia.
- Portabilidad entre plataformas y servidores.

Ciclo de vida de un servlet

- Cuando arranca el servidor, se crea una instancia:
 - Se inicializa el servlet (método **init()**)
- Cuando llega una petición:
 - Se invoca el método **service()** sobre un nuevo hilo.
- Cuando se cierra el servidor:
 - Se invoca el método **destroy()** y después se destruye el servlet.

JavaServer Pages (JSP): Tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML y XML pudiendo incluir código Java en las páginas web. El denominado *contenedor JSP* es el encargado de tomar la página, sustituir el código Java que contiene por el resultado de su ejecución y enviarla al cliente.

Las JSPs son en realidad una forma alternativa de crear servlets ya que el código JSP se traduce a código de servlet Java la primera vez que se le invoca y en adelante es el código del nuevo servlet el que se ejecuta produciendo como salida el código HTML que compone la página web de respuesta.

A continuación, se harán una serie de ejercicios en los que se podrá trabajar con ambas tecnologías.



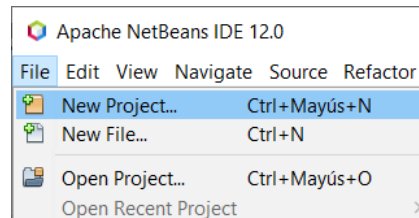
Práctica de Servlets

Hola Mundo

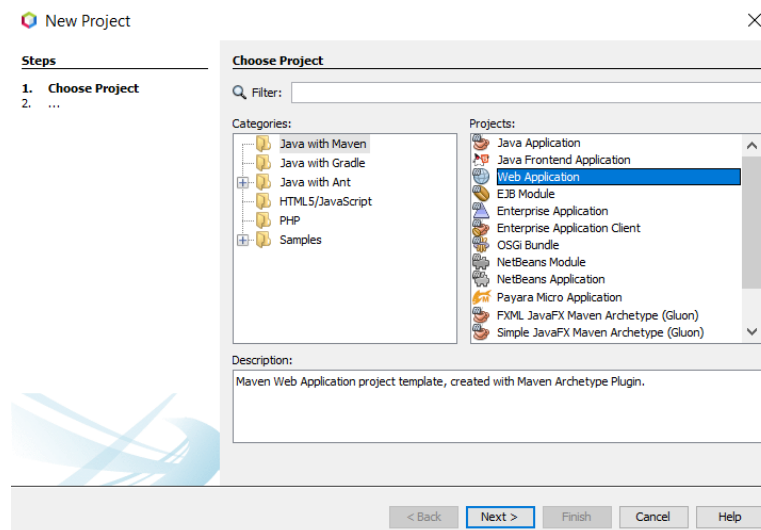
Con el proyecto **HolaMundo** se va a explicar cómo se crea un proyecto desde 0 en **Netbeans**.

Se instaló Netbeans 12, el JDK 1.8.311.

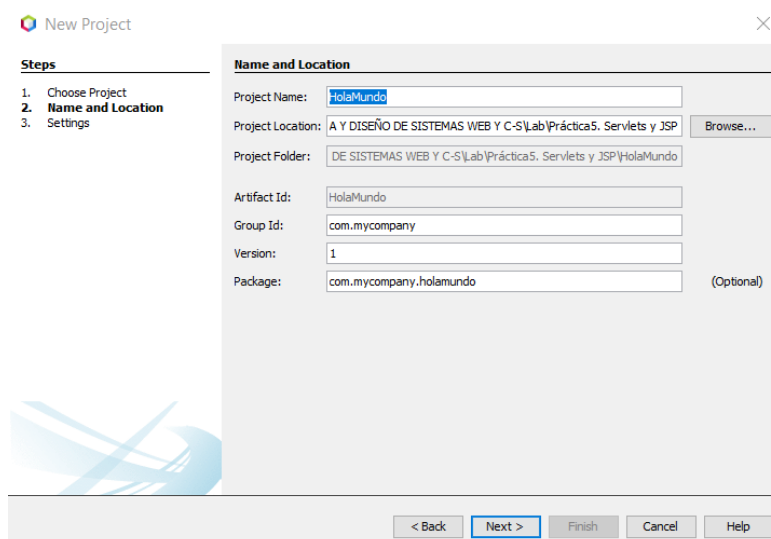
Para crear un nuevo proyecto: ir a la barra de menú > **File > New Project**.



Escoger el tipo de proyecto: **Java with Maven > Web Application**.

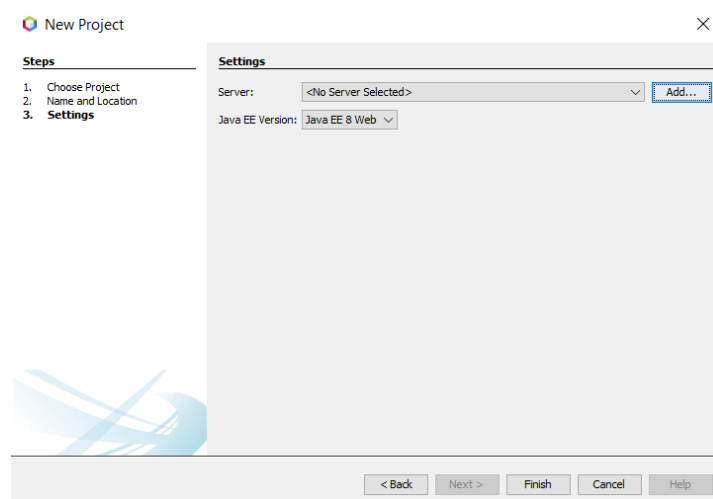


Se nombra el proyecto y clic en **Next**.

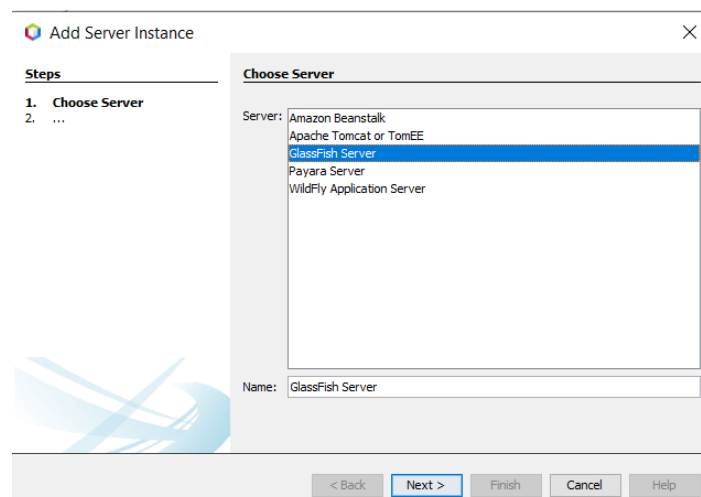




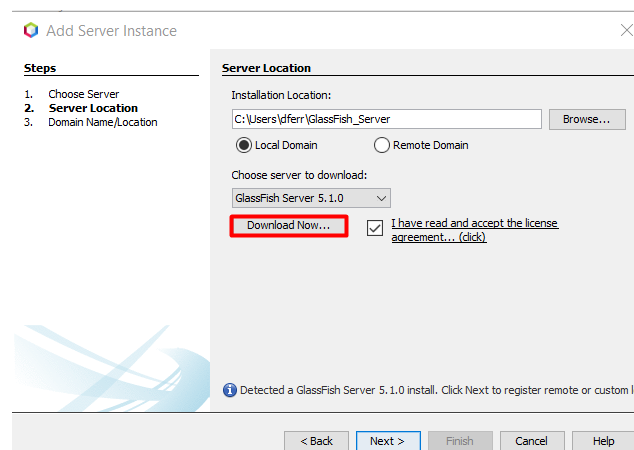
Si se le da clic en el botón **Add**, se escoge un servidor.



Escoger el servidor **GlassFish Server**.



Escoger la versión y la ruta en donde se quiere instalar el servidor. Clic en el check, **Download Now** y esperar a que se instale. Clic en **Next**.





Añadir más información si se desea y clic en **Finish**.

Add Server Instance

Steps

1. Choose Server
2. Server Location
3. **Domain Name/Location**

Domain Location

Domain:

Host: ☒ Loopback

DAS Port: HTTP Port: ☒ Default

Target:

User Name:

Password:

Register existing embedded domain: domain1

< Back Next > **Finish** Cancel Help

Escoger la versión **Java EE 8 Web**. Clic en **Finish**.

New Web Application

Steps

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

Settings

Server:

Java EE Version:

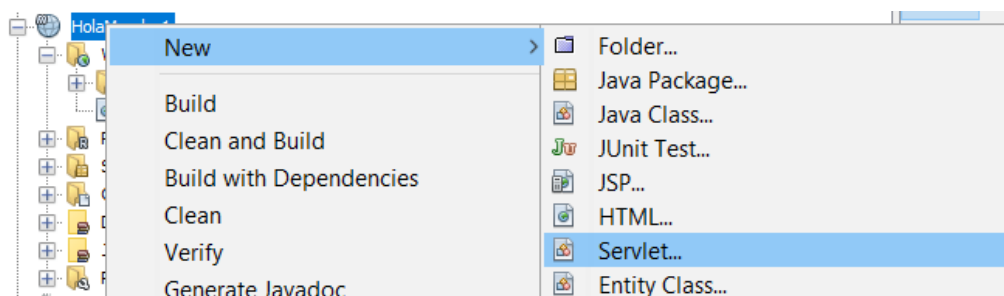
< Back Next > **Finish** Cancel Help



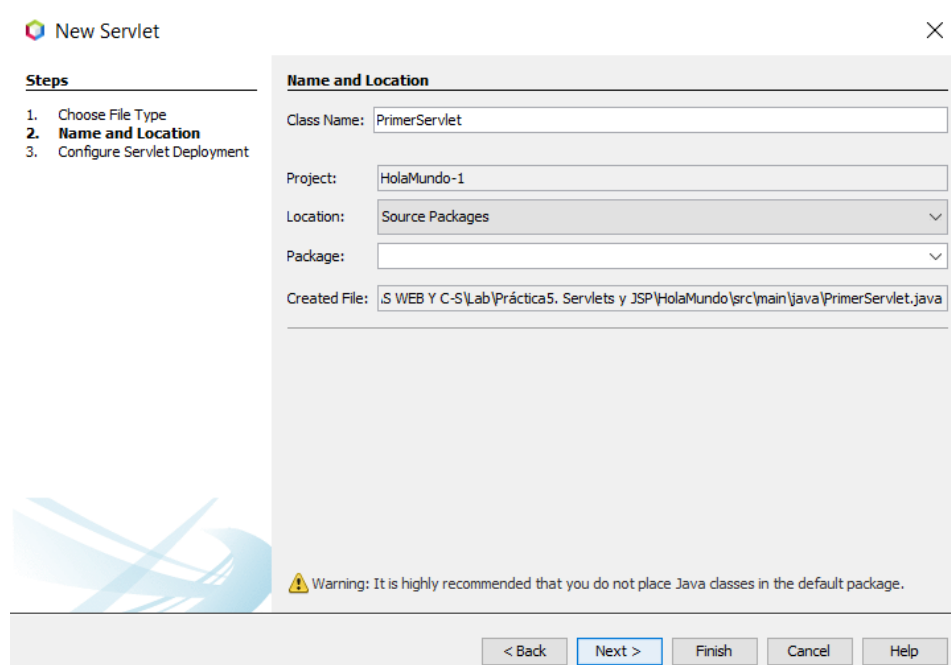
Incluir en **index.html** el siguiente este código.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>Primer Servlet</title>
6 </head>
7 <body>
8   <center><h2>Ejemplo de Ejecución de un Servlet</h2></center><br><br>
9   <form action=/HolaMundo/PrimerServlet method=GET>
10   <center>
11     <BR><input type=submit value="Ejecutar el Servlet"></form>
12   </center>
13 </body>
14 </html>
15
```

Para crear un servlet se le hace clic derecho en el proyecto, **New > Servlet**.



Nombrar el servlet y clic en **Next**.





Activar el check que se marca en rojo. Clic en **Finish**.

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

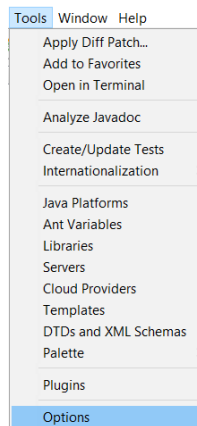
< Back Next > **Finish** Cancel Help

Código del servlet.

```
6
7 import java.io.*;
8 import javax.servlet.*;
9 import javax.servlet.http.*;
10
11 public class PrimerServlet extends HttpServlet {
12     @Override
13     public void doGet (HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException
14     {
15         res.setContentType("text/html");
16         try (PrintWriter out = new PrintWriter(res.getOutputStream())) {
17             out.println("<html>");
18             out.println("<head><title>HolaMundoServlet</title></head>");
19             out.println("<body>");
20             out.println("<h1><center>Hola Mundo desde el servidor WEB</center></h1>");
21             out.println("</body></html>");
22         }
23     }
24     @Override
25     public String getServletInfo()
26     {
27         return "Crea una página HTML que dice HolaMundo";
28     }
29 }
30
```

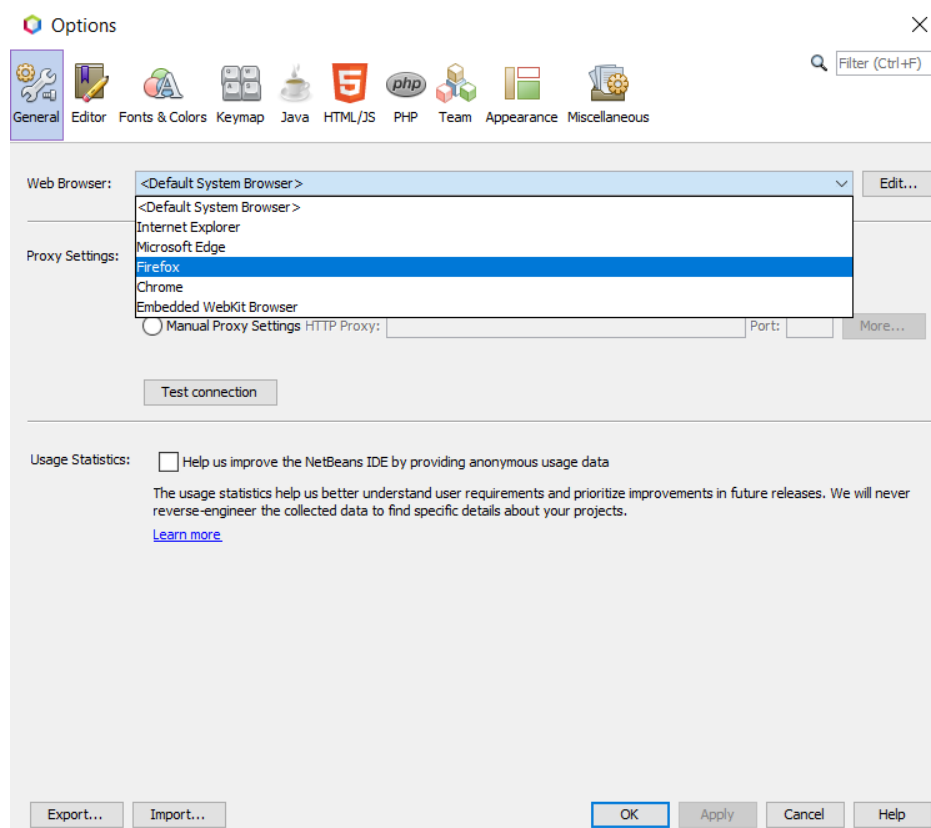

Cambiar el navegador WEB por defecto

En la barra de menú > **Tools** > **Options**



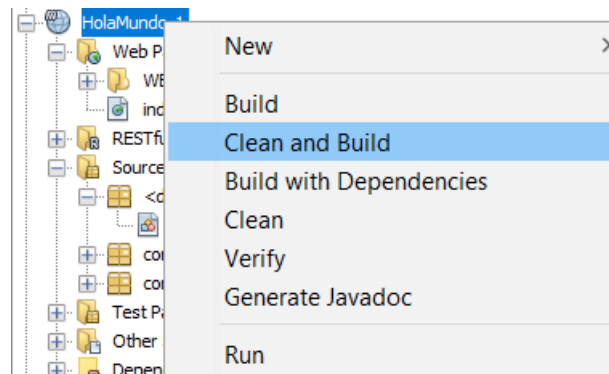
Desplegar la lista del apartado Web Browser.

Escoger su navegador preferido. Clic en **Apply** y luego **OK**.

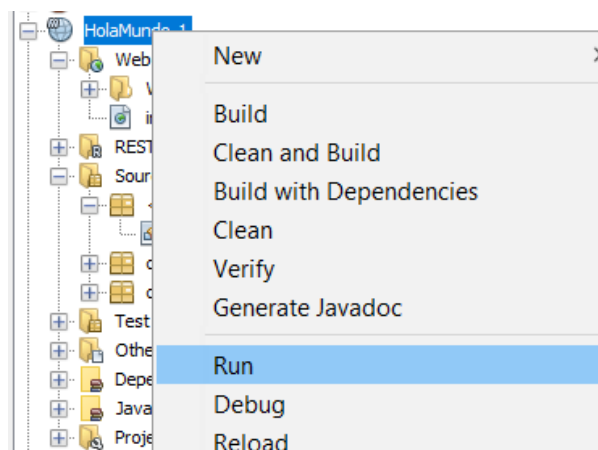




Clic derecho en el proyecto y **Clean and Build** para borrar las clases que han sido compiladas previamente y luego reconstruye todo el proyecto entero desde cero.



Clic derecho nuevamente en el proyecto y darle a **Run** para que se ejecute.



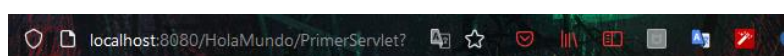
Si se da clic en el botón **Ejecutar el Servlet** se ejecuta un formulario que dara paso al servlet antes creado.



Ejemplo de Ejecución de un Servlet

Ejecutar el Servlet

El servlet muestra el siguiente mensaje en la página.



Hola Mundo desde el servidor WEB

Acceso básico a formularios

Para este ejercicio se crea un nuevo proyecto como se ha visto en el anterior punto. En él cambiamos el **index.html** que viene por defecto por el siguiente:

```
<!doctype html>
<html>

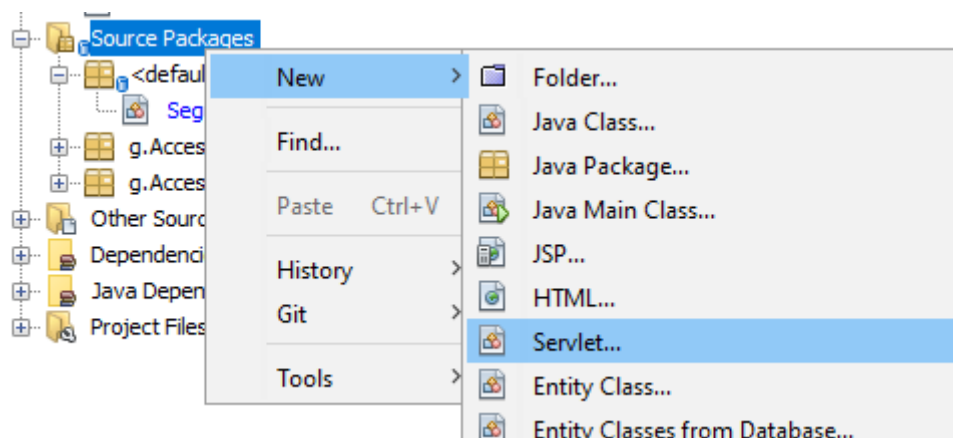
<head>
    <meta charset="utf-8">
    <title>Segundo Servlet</title>
</head>

<body>
    <center>
        <h2>Segundo Servlet</h2>
    </center>

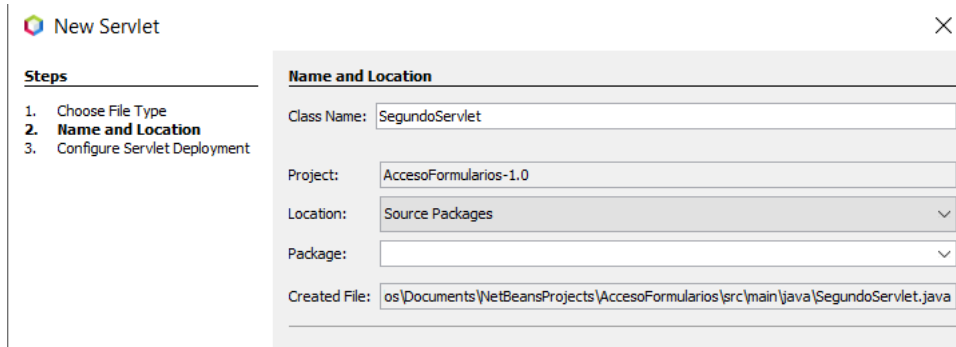
    <!-- Formulario que llama al Servlet creado como SegundoServlet -->
    <form action="SegundoServlet" method="POST">
        <center>
            Introduzca su nombre y pulse el botón de enviar
            <br><br>
            <input type="text" name="nombre">
            <br><br>
            <input type="submit" value="Enviar Nombre">
            <input type="reset" value="Borrar">
        </center>
    </form>
</body>

</html>
```

Este código HTML tiene un formulario que al pulsar el botón de **Enviar Nombre** realizará una petición **HTTP POST** a **SegundoServlet** enviado el contenido del input de nombre.



Se crea un nuevo Servlet en la carpeta de **Source Packages**



New Servlet

Steps

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

Name and Location

Class Name: SegundoServlet

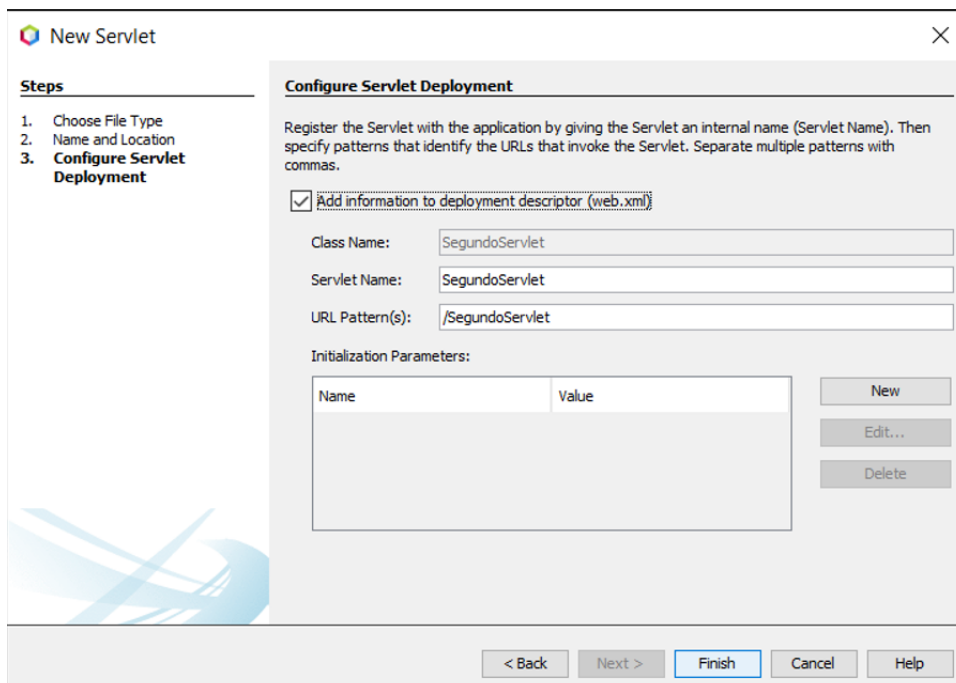
Project: AccesoFormularios-1.0

Location: Source Packages

Package:

Created File: os\Documents\NetBeansProjects\AccesoFormularios\src\main\java\SegundoServlet.java

Se le da un nombre a la clase, el mismo que el del enunciado.



New Servlet

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: SegundoServlet

Servlet Name: SegundoServlet

URL Pattern(s): /SegundoServlet

Initialization Parameters:

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

New Edit... Delete

< Back Next > Finish Cancel Help

Y en el paso 3, se marca la casilla que permite añadir la información del descriptor del Servlet al archivo **web.xml**.

```
public class SegundoServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP GET and POST
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            // Se obtiene el parámetro nombre del formulario
            String nombre = request.getParameter("nombre");

            // Se responde con código html mostrando el nombre:
            out.println("<!doctype html>");
            out.println("<html>");
            out.println("<head><title>HolaTalServlet</title></head>");
            out.println("<body>");
            out.println("<p><h1><center>Su nombre es: <strong>" + nombre + "</strong></center></h1></p>");
            out.println("</body></html>");
            out.close();
        }
    }
}
```



El código del Servlet trata de procesar la petición a través de la función ***processRequest***, se obtiene el parámetro nombre de la petición y se responde con código HTML, mostrando el nombre que ha sido enviado a través del formulario.

Se ejecuta el proyecto para lanzar el servidor web y, tras unos instantes, se abre una nueva ventana en el navegador mostrando el siguiente **index.html**:

The screenshot shows a web browser window with the title 'Segundo Servlet'. The address bar displays 'localhost:8080/AccesoFormularios/'. The page content includes the heading 'Segundo Servlet', the instruction 'Introduzca su nombre y pulse el botón de enviar', a text input field, and two buttons: 'Enviar Nombre' and 'Borrar'.

Si se introduce un nombre y se pulsa sobre **Enviar Nombre**, se ejecuta el Servlet y se obtiene el siguiente resultado de la ejecución:

The screenshot shows a web browser window with the title 'HolaTalServlet'. The address bar displays 'localhost:8080/AccesoFormularios/SegundoServlet'. The page content shows the result: 'Su nombre es: Carlos'.



Este fichero HTML cargado en cualquier navegador web (en este Google Chrome) mostrará la siguiente información:

← → ↻ localhost:8080/CalculadoraServlet/

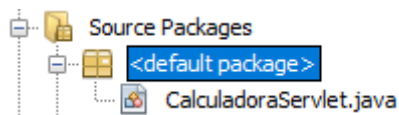
CALCULADORA

+

Calcular Limpiar

Se observa que hay dos inputs de texto, uno para el primer operando y otro para el segundo. Para elegir la operación se utiliza un **select** con los cuatro tipos de operaciones que realiza la calculadora. También hay dos inputs de botones, el de **Calcular** que será de tipo **submit** y ejecutará el Servlet que se explicará luego; y el de “Limpiar” que será de tipo **reset** y lo que hará es reiniciar la página con los valores por defecto de la operación y los operandos.

Para lograr el funcionamiento de la aplicación se creará un Servlet en el paquete por defecto de **Source Packages**. Este Servlet tendrá de nombre **CalculadoraServlet**.



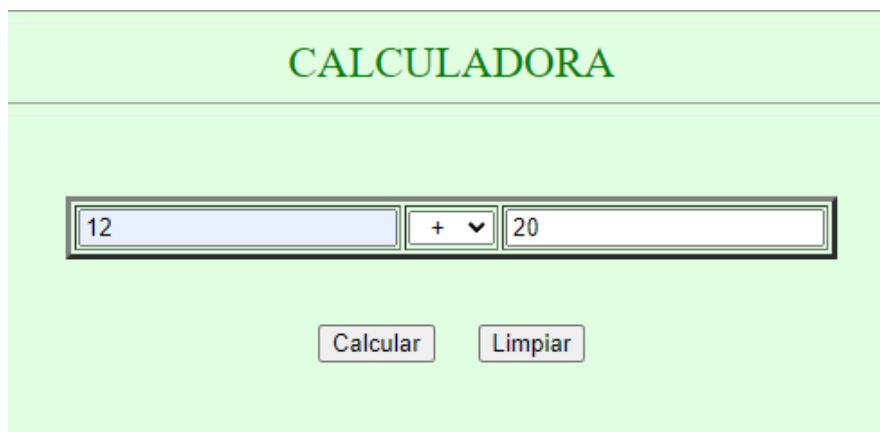
El contenido de este Servlet será el siguiente:

```
//
// CalculadoraServlet
//
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

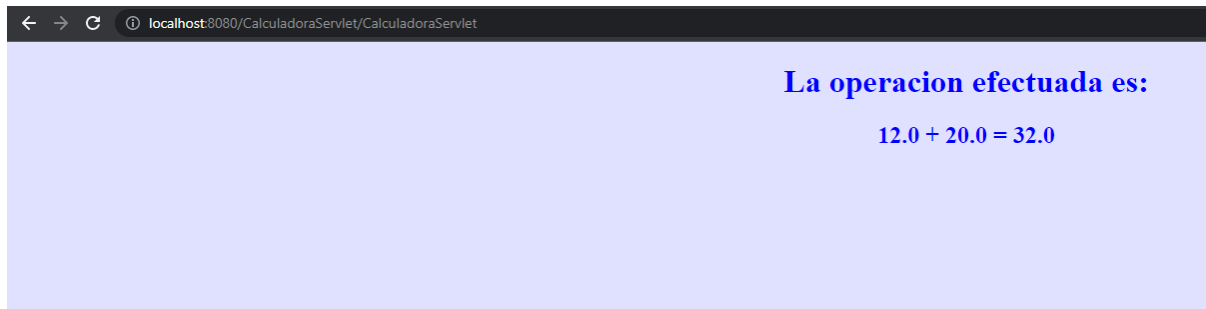
public class CalculadoraServlet extends HttpServlet {
    public void service (HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
        double op1, op2, result;
        int operacion;
        String simb_op[] = {"+", "-", "*", "/"};
        ServletOutputStream out = res.getOutputStream();
        op1 = Double.parseDouble(req.getParameter("operando1"));
        op2 = Double.parseDouble(req.getParameter("operando2"));
        operacion = Integer.parseInt(req.getParameter("operacion"));
        result = calcula(op1, op2, operacion);
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head><meta charset='utf-8'><title>Resultado de calcular con Servlet</title></head>");
        out.println("<body BGCOLOR = \"#E0E0FF\" TEXT= \"blue\">");
        out.println("<h1><center>La operacion efectuada es:</center></h1>");
        out.println("<h2> <b><center>"+ op1 + " " + simb_op[operacion - 1] + " " + op2 + " = " + result + "</center></b></h2>");
        out.println("</body>");
        out.println("</html>");
        out.close();
    }
    public double calcula(double op1, double op2, int operacion) {
        double result = 0;
        switch (operacion) {
            case 1:
                return op1 + op2;
            case 2:
                return op1 - op2;
            case 3:
                return op1 * op2;
            case 4:
                return op1 / op2;
        }
        return result;
    }
}
```

El Servlet utilizará los parámetros de la petición HTTP realizada por el formulario del HTML del primer operando, el segundo operando y la operación; de manera que se llame a la función **calcula** que los utilizará como argumentos para realizar la operación y obtener el resultado de ésta. En la respuesta HTTP se creará un documento HTML en el que se muestre la operación que se ha realizado y su resultado.

Tras ejecutar la aplicación web con el Servlet implementado se procede a realizar el siguiente ejemplo para mostrar el correcto funcionamiento de la aplicación.

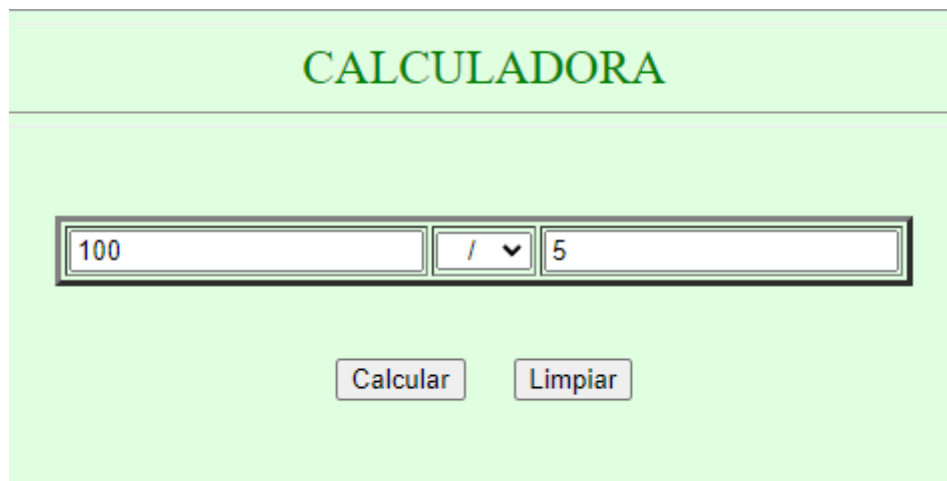


Se pulsa en **Calcular**.



Y se redirige al HTML del Servlet mostrando la operación y su resultado en formato decimal (double).

Otro ejemplo:

A screenshot of a web form titled 'CALCULADORA' in green capital letters. The form has a light green background. It contains two input fields: the first contains '100' and the second contains '5'. Between them is a dropdown menu showing a division symbol '/' and a downward arrow. Below the input fields are two buttons: 'Calcular' and 'Limpiar'.

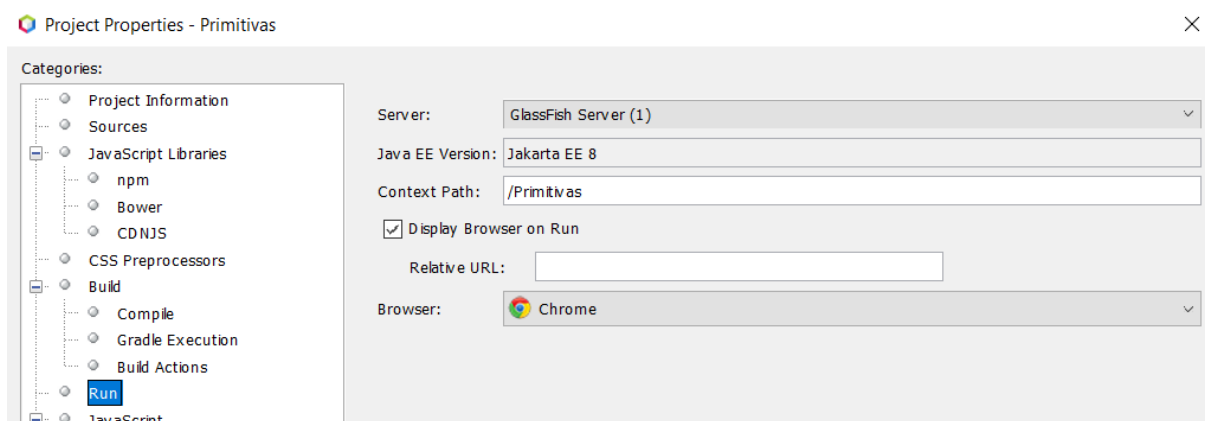
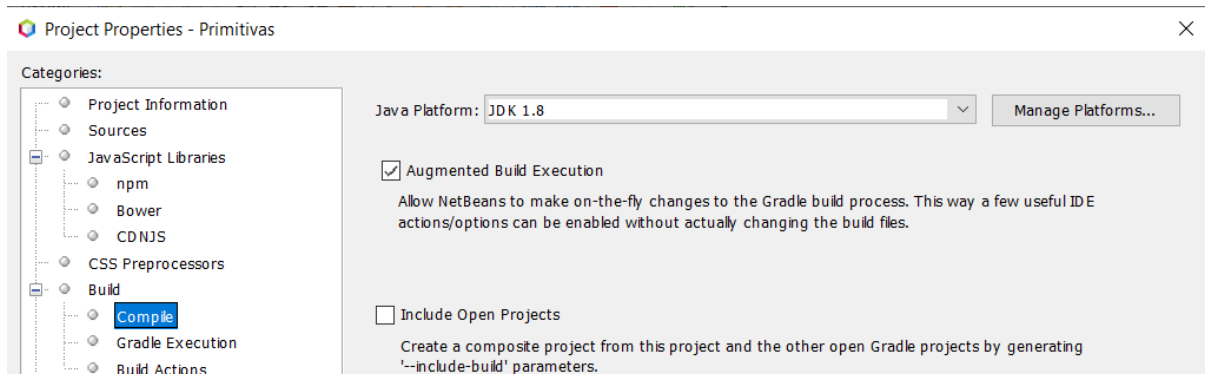
La operacion efectuada es:
100.0 / 5.0 = 20.0



Primitiva

Para la ejecución de este código proporcionado en el enunciado, se ha creado el proyecto del mismo modo comentado anteriormente en el primer apartado, es decir, seleccionando **Java with Gradle -> Web Application**. Para poder ejecutarlo se ha necesitado descargar la versión **jdk 8** que pudiera soportar la versión 5.1 de **glassfish**, ya que ni **jdk 11** ni **jdk 15** lo soportaban, todos ejecutados en la versión 12.3 de Netbeans. Además, se ha especificado el navegador encargado de abrir la página en ejecución, en este caso **Chrome**.

Dichas especificaciones se pueden observar en las imágenes que hay a continuación:





El archivo **index.html** necesitaba un par de correcciones para poder ejecutarse correctamente, ya que había ciertos fallos que no permitían la compilación, como son los valores de los atributos **input type**, **name**, **value**, **method** y **action** que necesitaban las comillas dobles para poder tomar los valores que se les asignaban. También se ha eliminado la etiqueta **<center>** que acompañaba a **<h2>** ya que se considera obsoleto.

```
<html>
<head>
  <title>Primitiva Servlet</title>
</head>
<body>
  <h2>Primitiva Servlet</h2>
  <form action="/Primitivas/CuartoServlet" method="POST">
    <center>
      Introduce tu combinación y pulsa el botón de enviar<br>
      <BR>NUM1:<input type="text" name="NUM1">
      <BR>NUM2:<input type="text" name="NUM2">
      <BR>NUM3:<input type="text" name="NUM3">
      <BR>NUM4:<input type="text" name="NUM4">
      <BR>NUM5:<input type="text" name="NUM5">
      <BR>NUM6:<input type="text" name="NUM6">
      <BR>
      <BR><input type="submit" value="Enviar Combinación">
      <input type="reset" value="Borrar">
    </center>
  </form>
</body>
</html>
```

Por otro lado, el código Java correspondiente, simplemente se ha incorporado como archivo Servlet y ajustado, junto con la agregación de tratamiento de errores en **service()**, y un pequeño cambio en la función booleana **comprueba()**.

```
private boolean comprueba(int arrayPrimi[], int num){
    for (int i=0; i<=5; i++){
        if (arrayPrimi[i]==num) return true;
    }
    return false;
}

@Override
public void service( HttpServletRequest petition, HttpServletResponse respuesta) throws ServletException, IOException{
    aciertos=0;
    try (ServletOutputStream out = respuesta.getOutputStream()) {
        combiUsuario[0] =
            Integer.parseInt(petition.getParameter("NUM1"));
    }
```

El programa se encarga de generar 6 números aleatorios y los almacena, posteriormente crea la página html de tipo **post** en la que pide al usuario que inserte 6 números a su gusto y pide pulsar **Enviar combinación** o **Borrar**, en el caso de que se quiera cambiar los actualmente escritos.

De esta manera, gracias al servlet http se accede al servidor conectado, en este caso **GlassFish**, que permite la conexión con la base de datos de Java y así mandar como respuesta de la petición los números creados aleatoriamente por el programa.

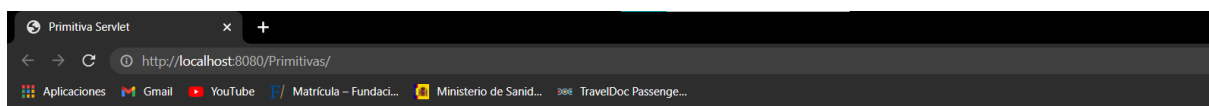
Estos últimos se usan para hacer una comprobación y ver si la combinación insertada por el cliente coincide o no con los generados. En cualquiera de los casos, bien se acierte o no, se muestra por pantalla la información, así como la combinación ganadora.



Puesto en ejecución, resultaría de la siguiente manera:

```
Notifications Output x
Run (Primitivas) x Java DB Database Process x GlassFish Server (1) x
JAVA_HOME="C:\Program Files (x86)\Java\jdk1.8.0_311"
cd C:\Users\biano\OneDrive\Documentos\NetBeansProjects\Primitivas; .\gradlew.bat --configure-on-demand -x check war
Configuration on demand is an incubating feature.
> Task :compileJava UP-TO-DATE
> Task :processResources NO-SOURCE
> Task :classes UP-TO-DATE
> Task :explodeWar UP-TO-DATE
> Task :war UP-TO-DATE

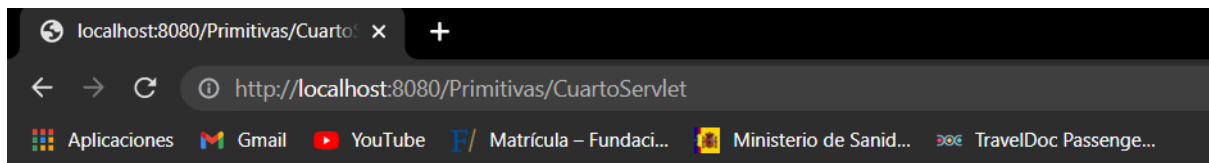
BUILD SUCCESSFUL in 215ms
3 actionable tasks: 3 up-to-date
Undeploying ...
In-place deployment at C:\Users\biano\OneDrive\Documentos\NetBeansProjects\Primitivas\build\exploded\Primitivas.war
```



Primitiva Servlet

Introduce tu combinación y pulsa el botón de enviar

NUM1:	<input type="text"/>
NUM2:	<input type="text"/>
NUM3:	<input type="text"/>
NUM4:	<input type="text"/>
NUM5:	<input type="text"/>
NUM6:	<input type="text"/>



Primitiva Servlet

Tu combinación es: 23 4 33 12 44 3

Número acertado:23

Números acertados:1

La combinación ganadora es: 1 17 23 34 38 48

Acceso a ficheros de texto plano

En este proyecto llamado **FicheroServlet** se crea una página con un campo para que se introduzca el nombre del usuario y se introduce dos botones:

- El botón **Enviar** manda el contenido del campo por medio de un formulario a el servlet.
- El botón **Borrar** elimina el contenido del campo.

Código del **index.html**.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Segundo Servlet</title>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6   </head>
7   <body>
8     <h2><center>Segundo Servlet</center></h2>
9     <form action="/FicheroServlet/FicheroServlet" method=POST>
10      <BR><BR>
11      <center>
12        Introduzca su nombre y pulse el botón de enviar<BR>
13        <BR><input type=text name=TEXTO><BR>
14        <BR><input type=submit value=Enviar Nombre>
15        <input type=reset value=Borrar>
16      </center>
17    </form>
18  </body>
19 </html>
20

```

En el campo nombre se introduce **Daniel** y se da a **Enviar**.



El servlet **FicheroServlet** lee y muestra el contenido de **fichero.txt** (cambiar el valor de la variable **ruta** por la ruta absoluta en donde se encuentra el fichero en el ordenador). Escribe también el nombre que se introdujo en el campo nombre del formulario del **index.html**.

```

8 public class FicheroServlet extends HttpServlet
9 {
10     StringBuffer mensaje = null;
11     FileOutputStream fos = null;
12     String[] strTEXTO;
13
14     @Override
15     public void init(ServletConfig config) throws ServletException
16     {
17         super.init(config);
18         FileInputStream fis = null;
19         //ATENCIÓN: INTRODUZCA LA RUTA ABSOLUTA DEL FICHERO. Ejemplo: C:/Users/dferr/OneDrive/Desktop/fichero.txt
20         String ruta = "C:/Users/dferr/OneDrive/Desktop/";
21         try
22         {
23             fis = new FileInputStream(ruta + "fichero.txt");
24         } catch (java.io.FileNotFoundException e) {e.printStackTrace();}
25
26         mensaje = new StringBuffer();
27         try
28         {
29             int caracter;
30             while ( (caracter = fis.read()) != -1 )
31             {
32                 mensaje.append((char)caracter);
33             }
34             fis.close();
35             fos = new FileOutputStream(ruta + "log.txt");
36         } catch (java.io.IOException e) {e.printStackTrace();}
37     }
38 }

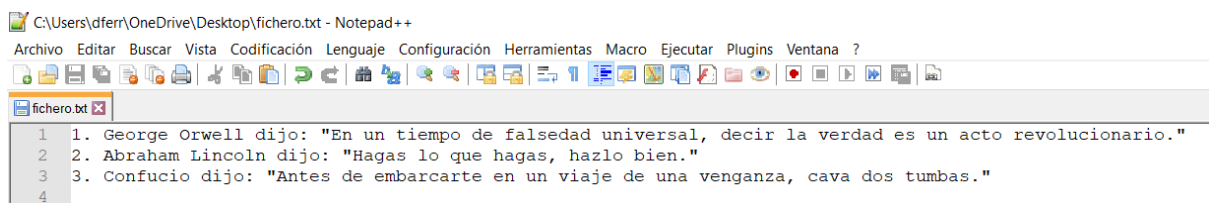
```

```

40 @Override
41 public void service( HttpServletRequest petición, HttpServletResponse respuesta ) throws ServletException, IOException
42 {
43     respuesta.setContentType("text/html;charset=UTF-8");
44     strTEXTO = petición.getParameterValues("TEXTO");
45     try (ServletOutputStream out = respuesta.getOutputStream()) {
46         out.println("<p>" + mensaje + "</p>");
47         out.println("<p>Su nombre es: " + strTEXTO[0] + "</p>");
48     }
49     registrar();
50 }
51
52 @Override
53 public void destroy()
54 {
55     try
56     {
57         fos.close();
58     } catch (java.io.IOException e) {e.printStackTrace();}
59 }
60
61 public synchronized void registrar()
62 {
63     try
64     {
65         // fos.write(strTEXTO[0].getBytes());
66         fos.write(strTEXTO[0].getBytes());
67         destroy();
68     } catch (java.io.IOException e) {e.printStackTrace();}
69 }
70 }

```

Se accederá al contenido de **fichero.txt** por ruta absoluta.



C:\Users\dferr\OneDrive\Desktop\fichero.txt - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?

fichero.txt

```

1 1. George Orwell dijo: "En un tiempo de falsedad universal, decir la verdad es un acto revolucionario."
2 2. Abraham Lincoln dijo: "Hagas lo que hagas, hazlo bien."
3 3. Confucio dijo: "Antes de embarcarte en un viaje de una venganza, cava dos tumbas."
4

```

La página servlet muestra el contenido del fichero leído y el nombre introducido.

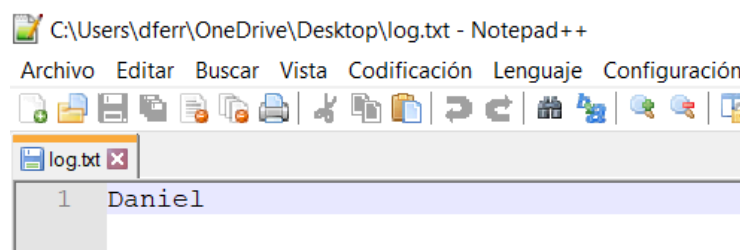


localhost:8080/FicheroServlet/FicheroServlet

1. George Orwell dijo: "En un tiempo de falsedad universal, decir la verdad es un acto revolucionario." 2. Abraham Lincoln dijo: "Hagas lo que hagas, hazlo bien." 3. Confucio dijo: "Antes de embarcarte en un viaje de una venganza, cava dos tumbas."

Su nombre es: Daniel

Contenido del archivo **log.txt**.



C:\Users\dferr\OneDrive\Desktop\log.txt - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración

log.txt

```

1 Daniel

```



Acceso a datos

El proyecto será creado en un servlet, **Java with Maven** > web **Application**, **GlassFish Server** y **Java EE 8** web, como se mencionó también previamente.

En la creación del servlet del proyecto, se activará **Add information to deployment descriptor** (web.xml)"

Y dentro de este servlet se encuentra el siguiente código (el cual fue proporcionado en la blackboard):

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class EncuestaServlet extends HttpServlet {
    Statement mandato = null;
    Connection conexion = null;

    public void init(ServletConfig config) throws ServletException {
        try {
            {
                conexion = DriverManager.getConnection("jdbc:derby://localhost:1527/sample", "app", "app");
                mandato = conexion.createStatement();
            }
        } catch (Exception e) {
            {
                System.out.println("Problemas al conectar con la base de datos");
            }
        }
    }

    public void service( HttpServletRequest petition, HttpServletResponse respuesta )throws ServletException, IOException
    {
        ServletOutputStream out = respuesta.getOutputStream();
        respuesta.setContentType("text/html");
        String strNombre = petition.getParameter("NOMBRE");
        String strEmail = petition.getParameter("EMAIL");
        String strRespuesta = petition.getParameter("RESPUESTA");
        try {
            {
                mandato.executeUpdate("INSERT INTO ENCUESTA VALUES( '" + strNombre + "', '" + strEmail + "', '" + strRespuesta + "')");
            }
        } catch (SQLException e) {
            {
                System.out.println(e);
                return;
            }
        }
        try {
            {
                int intSI = 0;
                int intNO = 0;
                ResultSet resultado = mandato.executeQuery("SELECT RESPUESTA FROM ENCUESTA");
                while(resultado.next()) {
                    {
                        String resp = resultado.getString("RESPUESTA");
                        if(resp.compareTo("SI")==0) intSI++; else intNO++;
                    }
                }
                out.println("<h2><center>Encuesta Servlet</center></h2>");
                out.println("<BR>Gracias por participar en esta encuesta.");
                out.println("<BR>Los resultados hasta este momento son :");
                out.println("<BR> SI : "+intSI);out.println("<BR> NO : "+intNO);
            }
        } catch (Exception e) {
            {
                System.out.println(e);
                return;
            }
        }
    }

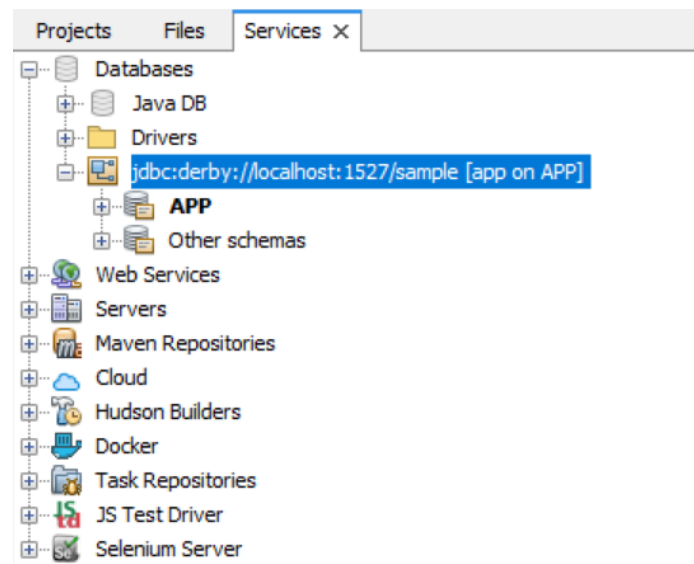
    public void destroy()
    {
        try {
            {
                conexion.close();
            }
        } catch (SQLException e) {
            {
                System.out.println(e);
                return;
            }
        }
    }
}
```



Y el formulario HTML:

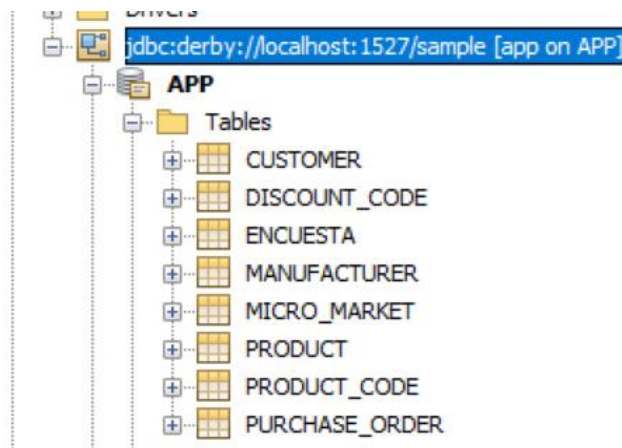
```
<html>
<head>
  <title>Ejemplo Encuesta</title>
</head>
<body bgcolor=white>
<center>
  <h2>Por favor rellene todos los datos</h2>
  <form action=/EncuestaServlet/EncuestaServlet method=POST>
    <BR>Nombre:<BR>
    <BR><input type=text name=NOMBRE><BR>
    <BR>E-Mail:<BR><BR> <input type=text name=EMAIL><BR>
    <BR><B>Pregunta:</B>¿Piensas utilizar a los Servlets para los proyectos a partir de ahora?<BR>
    <BR>Si<input type=radio name=RESPUESTA value=SI>No<input type=radio name=RESPUESTA value=NO><BR>
    <BR><input type=submit value=Enviar><input type=reset value=Borrar>
  </form>
</center>
</body>
</html>
```

Para poder almacenar estos datos, se utilizará la base de datos **Derby** incorporada en Netbeans:

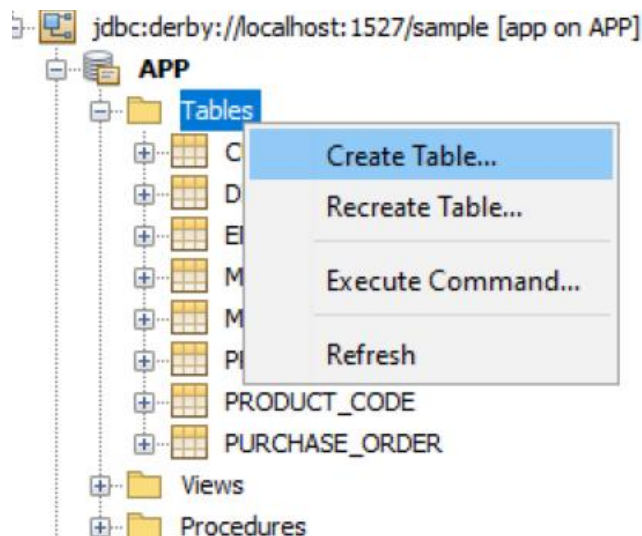


Para iniciarla, basta con darle doble clic a la opción seleccionada:

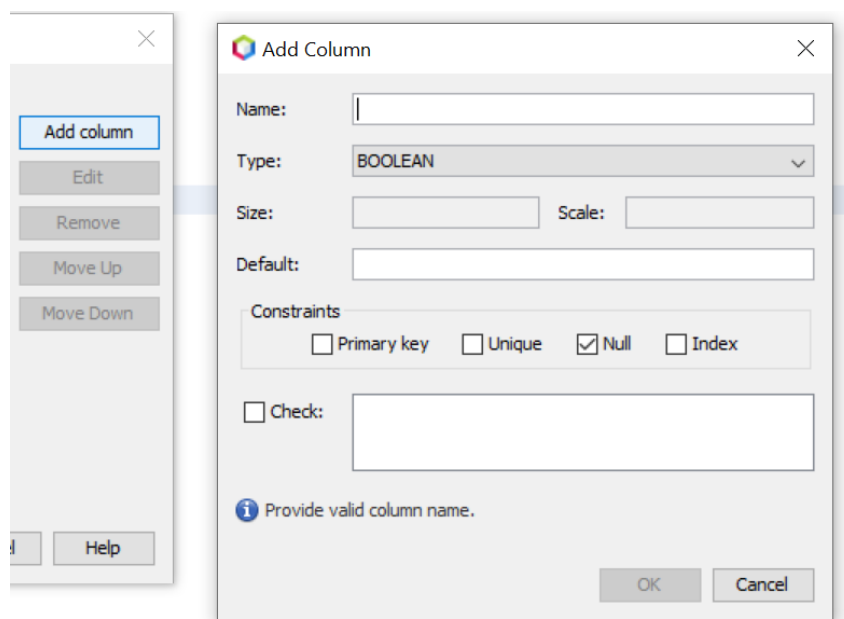
Expandiendo la opción **APP** y **Tables**, podemos ver todas las tablas que se encuentran en **Derby**:



Se creará una nueva tabla **Encuesta**

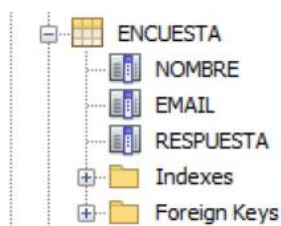


Pulsando añadir columna se pueden añadir los nombres de los parámetros y sus tipos:

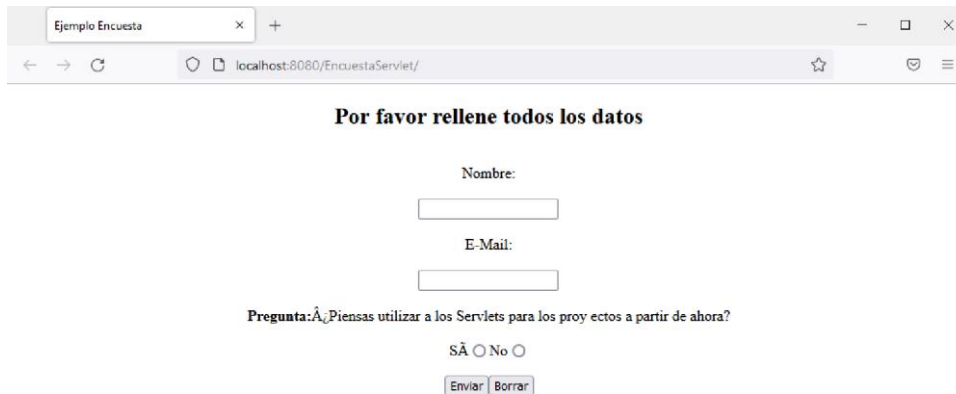


Los tres parámetros creados son de tipo varchar, el nombre y email son de tamaño 30 y la respuesta de tamaño 2 (sí/no).

Una vez creado:



Ejecutando el proyecto:



Por favor rellene todos los datos

Nombre:

E-Mail:

Pregunta: ¿Piensas utilizar a los Servlets para los proyectos a partir de ahora?

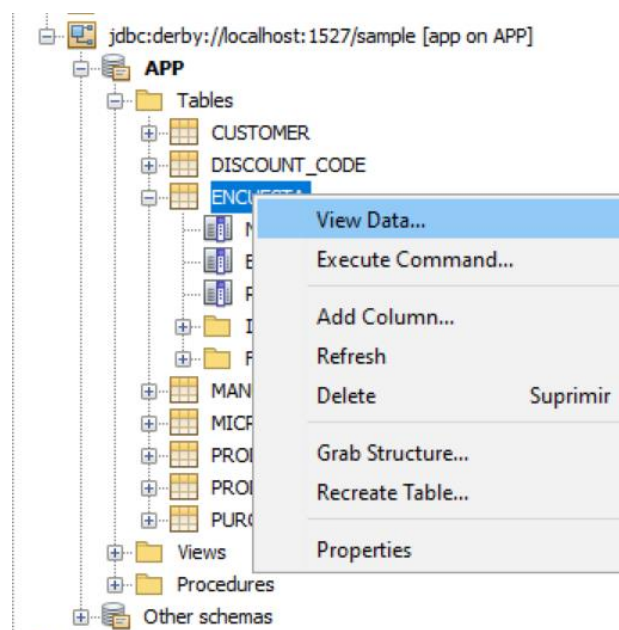
SI ☐ No ☐

Una vez rellenos los datos:

Encuesta Servlet

Gracias por participar en esta encuesta.
 Los resultados hasta este momento son :
 SI : 2
 NO : 1

Y se pueden visualizar todos los datos almacenados:





Y aquí se pueden observar los datos insertados.

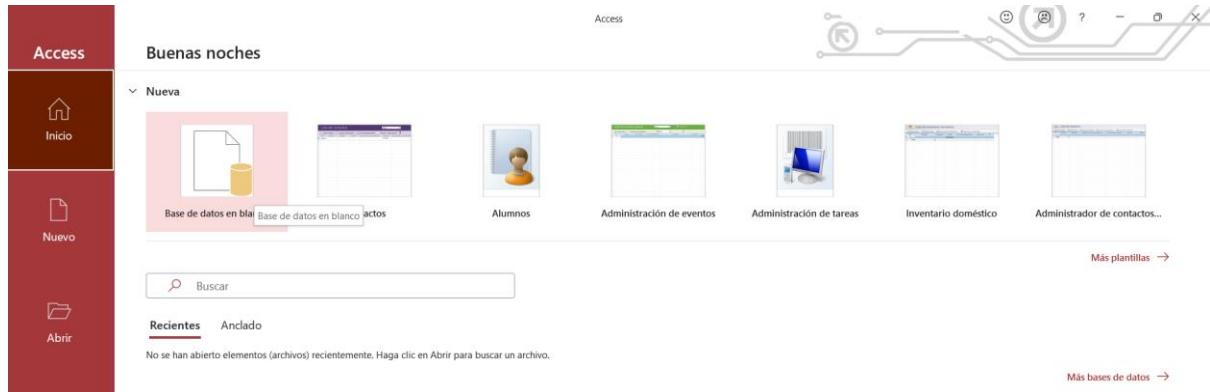
#	NOMBRE	EMAIL	RESPUESTA
1	ddd	123	SI
2	elnombre	elEmail@email.com	SI

Práctica de Java Server Pages

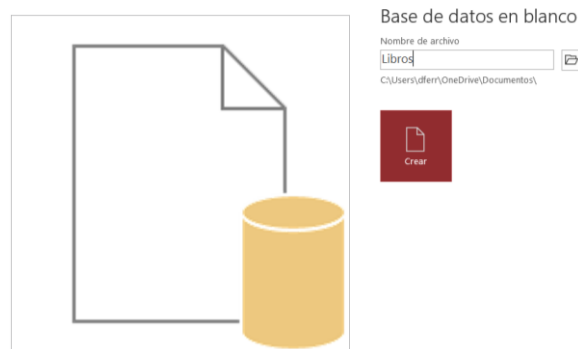
Acceso a datos (I)

Creación de una BBDD en Microsoft Access (EXTRA)

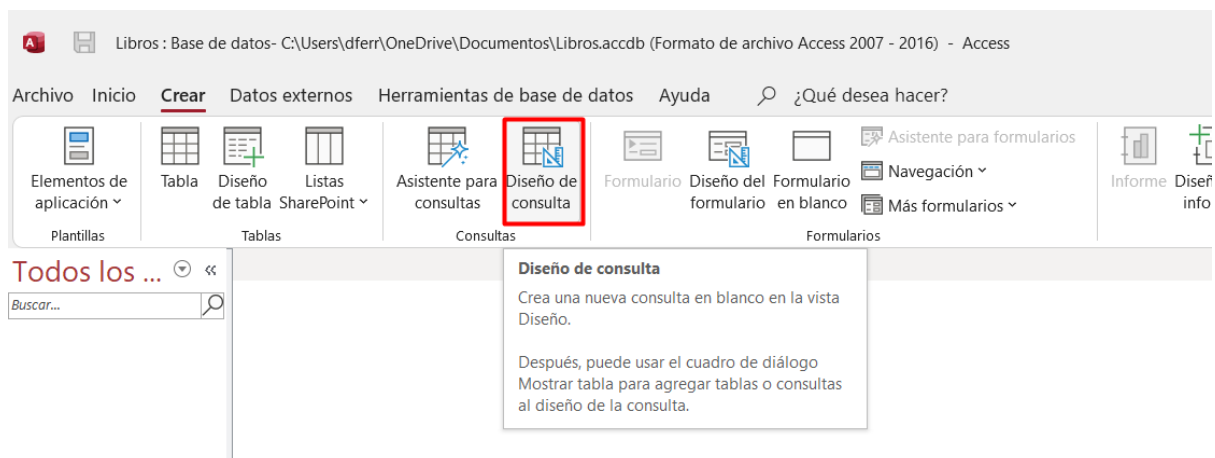
Se debe abrir Microsoft Access. Clic en **Base de datos en blanco**.



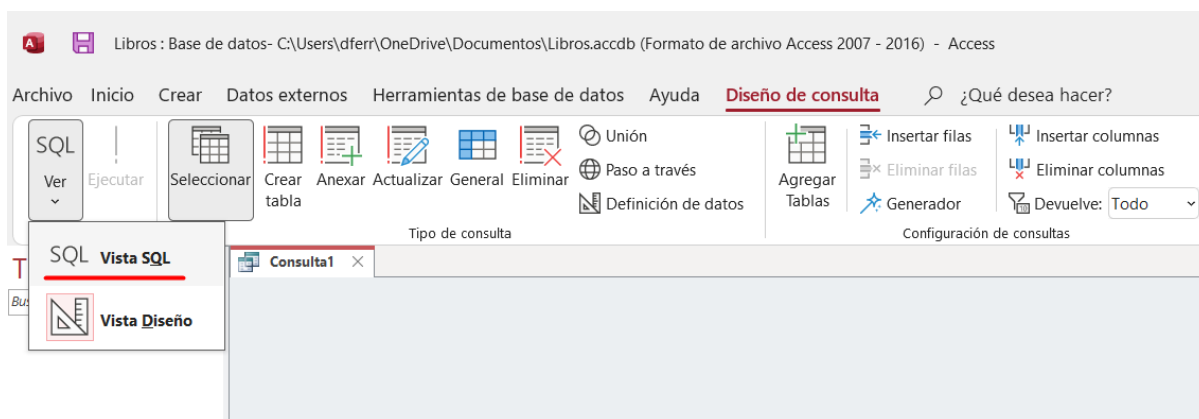
Nombrar la Base de datos (Libros).



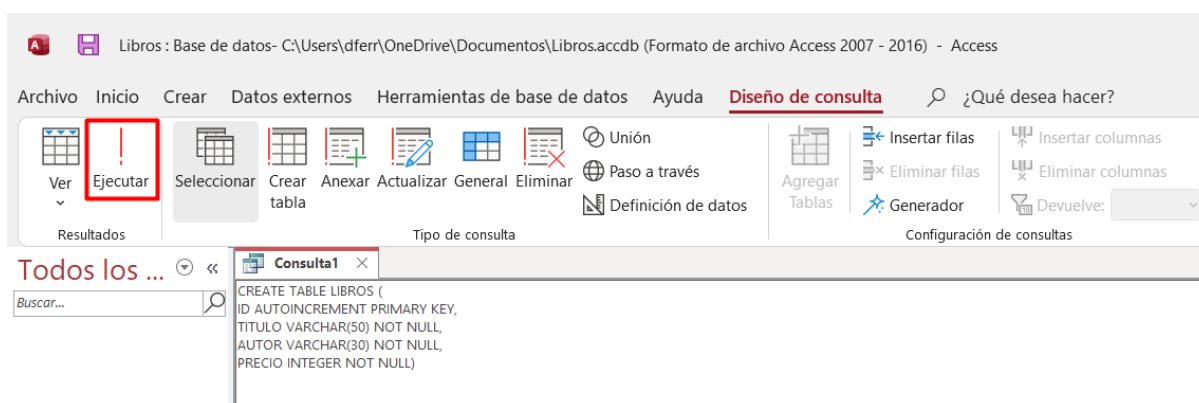
Eliminar la tabla por defecto. Clic en Diseño de consulta.



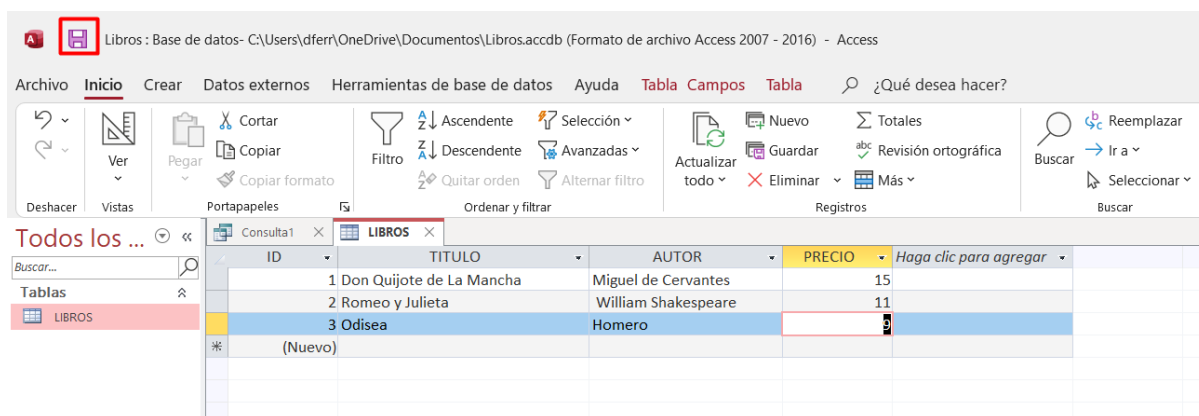
Clic en **Diseño de consulta > SQL > SQL Vista SQL**.



Clic en **Ejecutar** una vez que se escriba el código sql para crear la tabla LIBROS.



Guardar una vez que se escriban los datos de estas tres filas.

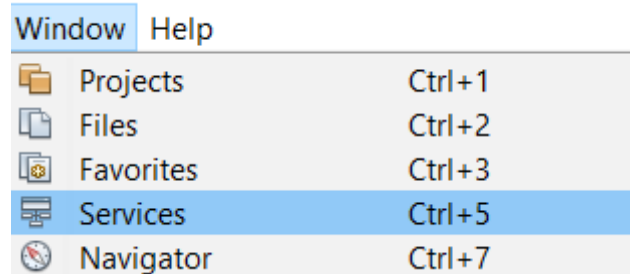




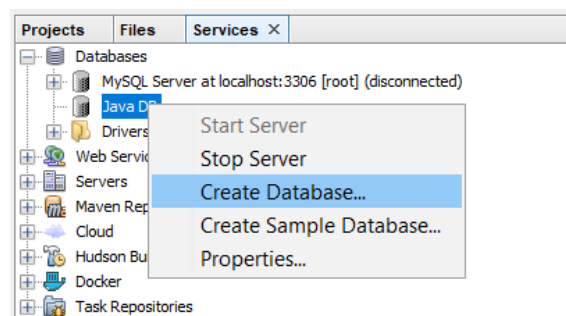
Creación de la Base de Datos en Derby

Apache Derby es un sistema gestor de base de datos relacional escrito en Java que puede ser empotrado en aplicaciones Java y utilizado para procesos de transacciones en línea.

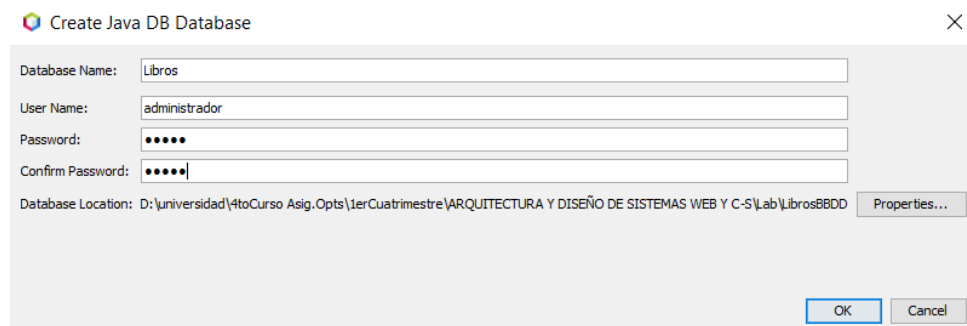
Se hace clic en **Window**, en la barra de menú, y **Services** o **Ctrl+5** si se prefiere usar el teclado.



Se escoge la opción de **Databases**. Clic derecho en **Java DB** > **Create Database**



Se nombra la BBDD y el usuario. Escribir una contraseña y especificar la ruta en donde se desea almacenar la Base de Datos. Clic en **OK**.

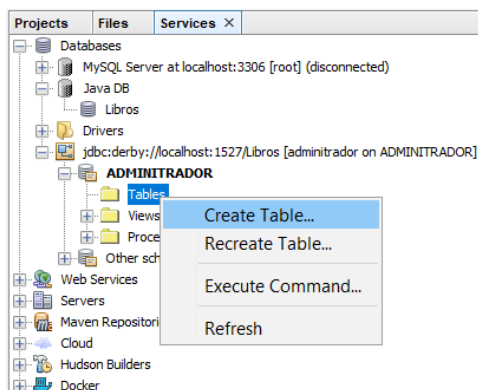


Si no se encuentra la conexión se debe hacer clic derecho en el nombre de la Base de Datos y escoger la opción Conectar.

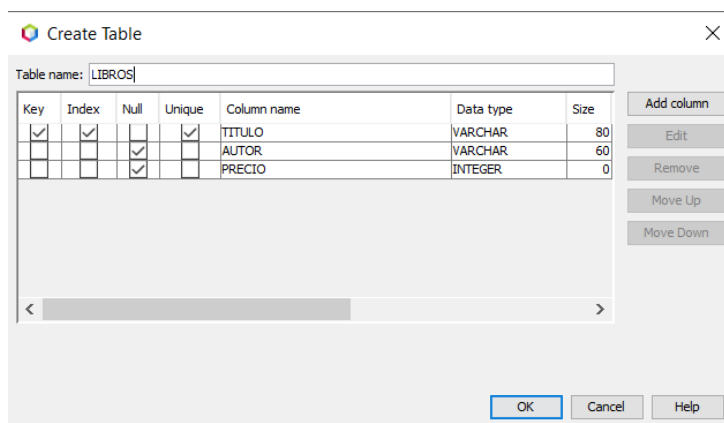
Doble clic en la conexión **jdbc:derby://localhost.....**

Manual

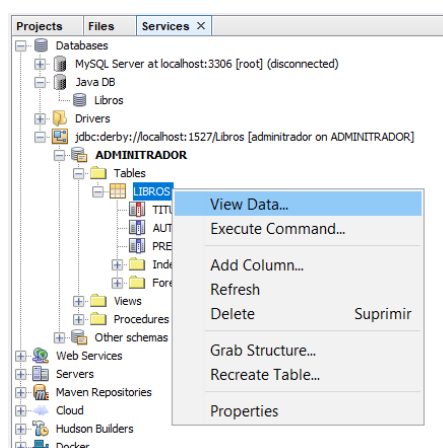
Clic derecho en la carpeta **Tables**, usuario **ADMINISTRADOR**, escoger la opción **Create Table**.



Dar nombre a la Tabla: **LIBROS**. **Add column** para añadir columna, se le da nombre, se especifica el tipo y se le da un tamaño al dato. Clic en **OK**.



Clic derecho sobre la nueva tabla > **View Data**.





Se añadirá 3 filas a la tabla **LIBROS**.

The screenshot shows a SQL Developer window with the following SQL code:

```

1 INSERT INTO LIBROS VALUES ('Don Quijote de La Mancha', 'Miguel de Cervantes', 11);
2 INSERT INTO LIBROS VALUES ('Romeo y Julieta', 'William Shakespeare', 11);
3 INSERT INTO LIBROS VALUES ('Odisea', 'Homero', 9);
4
5 SELECT * FROM LIBROS;
6
7

```

Below the code, the query results are displayed in a table:

#	TITULO	AUTOR	PRECIO
1	Don Quijote de La Mancha	Miguel de Cervantes	11
2	Romeo y Julieta	William Shakespeare	11
3	Odisea	Homero	9

Código

En el fichero **tablaLIBROS.sql** está el código SQL para la creación de tabla LIBROS y sus filas.

En la línea en rojo, único cambio en el ejercicio de la práctica a nivel de código, se añade el nombre de la Base Datos, el usuario y la contraseña.

```

<%=
//Declaraciones de las variables utilizadas para la conexión a la base de datos y para la recuperación de datos de las tablas
Connection c;
Statement s;
ResultSet rs;
ResultSetMetaData rsmd; %>
<%
// Inicialización de las variables necesarias para la conexión a la base de datos y realización de consultas.
c = DriverManager.getConnection("jdbc:derby://localhost:1527/Libros", "administrador", "admin");
s = c.createStatement();
rs = s.executeQuery("SELECT * FROM LIBROS");
rsmd = rs.getMetaData();
%>

```

Resultado: Muestra las columnas y las filas de la tabla antes creada.

TITULO	AUTOR	PRECIO
Don Quijote de La Mancha	Miguel de Cervantes	11
Romeo y Julieta	William Shakespeare	11
Odisea	Homero	9

Comprobación

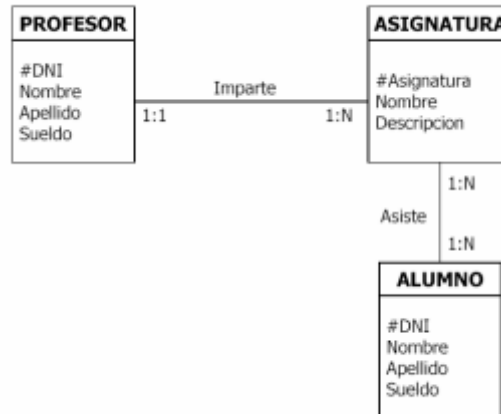
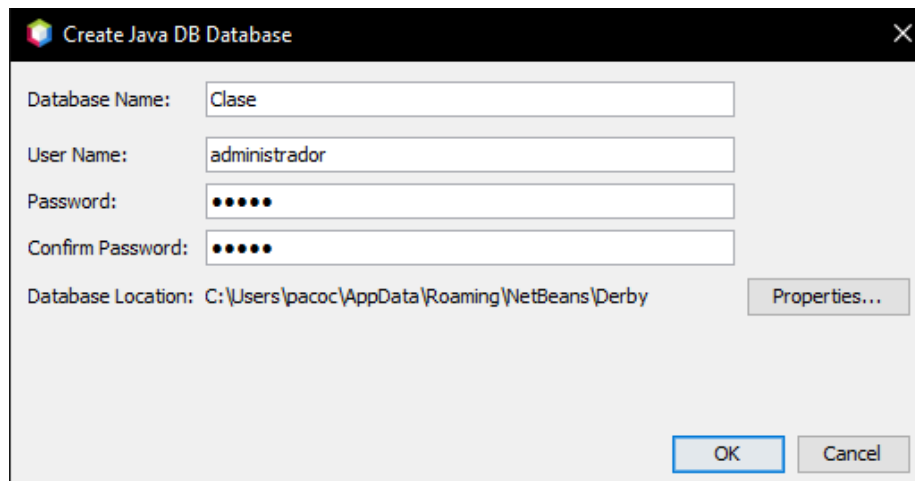
Para comprobar el correcto funcionamiento del ejercicio se debe descargar y descomprimir el archivo **Libros.zip** (Base de Datos Libros) en la carpeta en donde se tenga las Bases de Datos de Derby. Ejemplo con la Bases de Datos **Clase y Libros**:

Nombre	Fecha de modificación	Tipo	Tamaño
Clase	25/11/2021 20:49	Carpeta de archivos	
Libros	25/11/2021 20:49	Carpeta de archivos	
derby	25/11/2021 23:05	Documento de tex...	7 KB
derby.properties	24/11/2021 23:20	Archivo PROPERTI...	1 KB

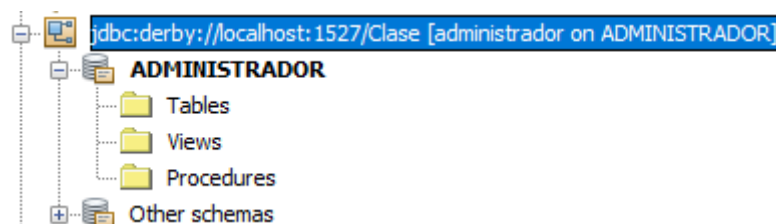
Acceso a datos (II)

Creación de la base de datos

Para este apartado se crea una base de datos de la misma manera que en el apartado anterior, cuyo nombre va a ser **Clase** y el modelo a seguir será el siguiente:

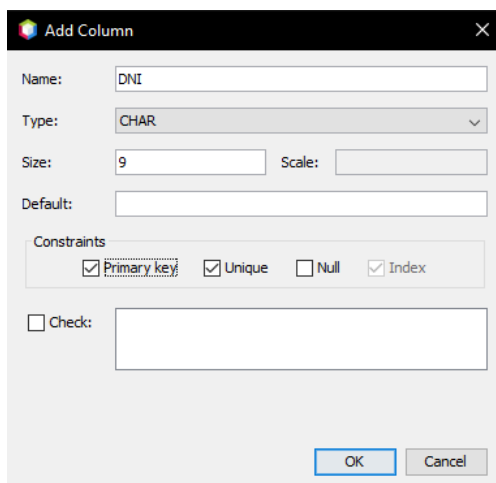
Base de datos creada:



A continuación, se crean las tablas para los profesores, las asignaturas y los alumnos.

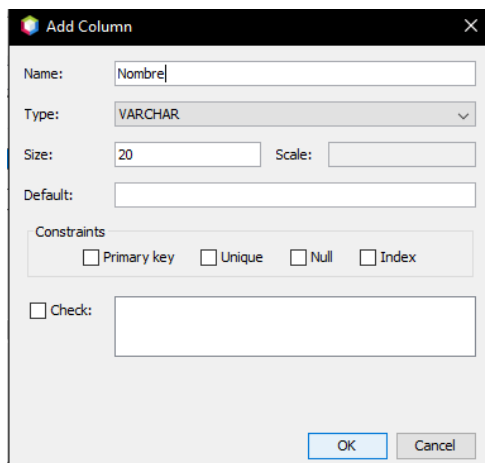
La tabla de los profesores estará formada por los siguientes atributos:

DNI



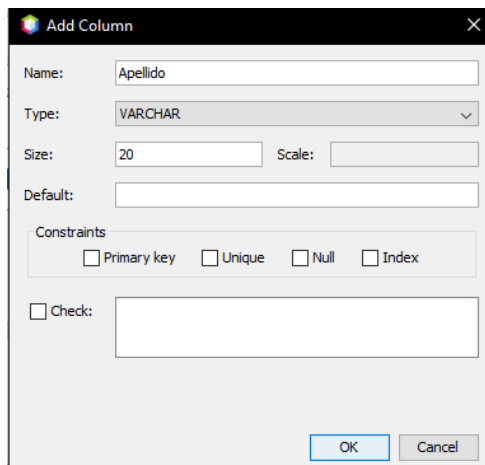
The 'Add Column' dialog box for the 'DNI' attribute. The 'Name' field contains 'DNI'. The 'Type' dropdown is set to 'CHAR'. The 'Size' field is '9' and the 'Scale' field is empty. The 'Default' field is empty. In the 'Constraints' section, the checkboxes for 'Primary key', 'Unique', and 'Index' are checked, while 'Null' is unchecked. There is a 'Check' checkbox which is unchecked, followed by an empty text box. At the bottom are 'OK' and 'Cancel' buttons.

Nombre



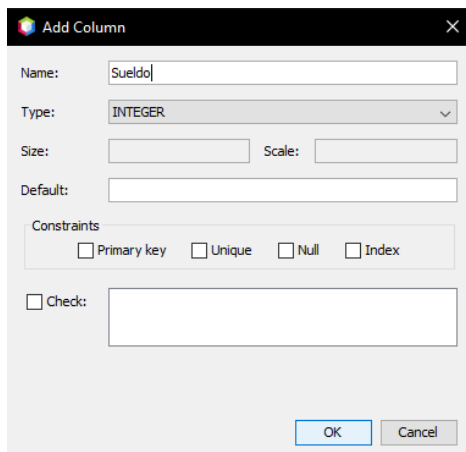
The 'Add Column' dialog box for the 'Nombre' attribute. The 'Name' field contains 'Nombre'. The 'Type' dropdown is set to 'VARCHAR'. The 'Size' field is '20' and the 'Scale' field is empty. The 'Default' field is empty. In the 'Constraints' section, all checkboxes ('Primary key', 'Unique', 'Null', 'Index') are unchecked. There is a 'Check' checkbox which is unchecked, followed by an empty text box. At the bottom are 'OK' and 'Cancel' buttons.

Apellido



The 'Add Column' dialog box for the 'Apellido' attribute. The 'Name' field contains 'Apellido'. The 'Type' dropdown is set to 'VARCHAR'. The 'Size' field is '20' and the 'Scale' field is empty. The 'Default' field is empty. In the 'Constraints' section, all checkboxes ('Primary key', 'Unique', 'Null', 'Index') are unchecked. There is a 'Check' checkbox which is unchecked, followed by an empty text box. At the bottom are 'OK' and 'Cancel' buttons.

Sueldo



Add Column

Name:

Type:

Size: Scale:

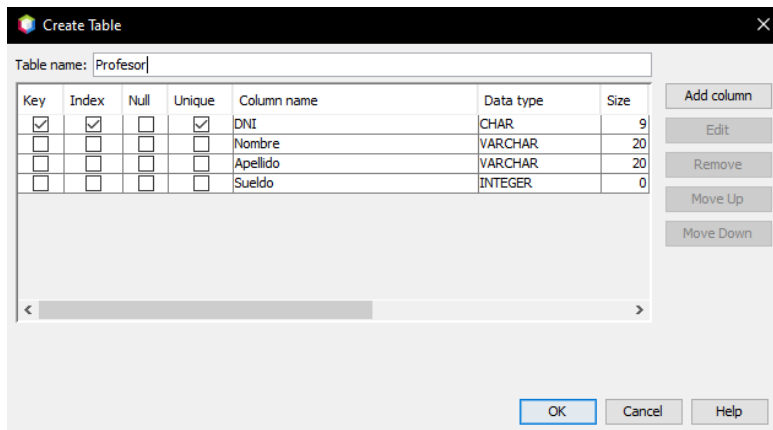
Default:

Constraints

☐ Primary key ☐ Unique ☐ Null ☐ Index

☐ Check:

Tabla con todos los atributos



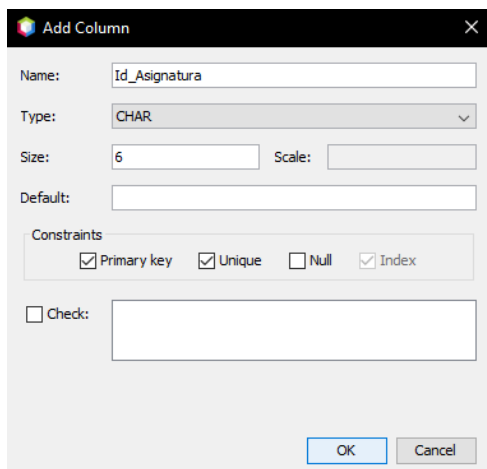
Create Table

Table name:

Key	Index	Null	Unique	Column name	Data type	Size
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	DNI	CHAR	9
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Nombre	VARCHAR	20
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Apellido	VARCHAR	20
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sueldo	INTEGER	0

Atributos de la tabla de las asignaturas:

ID de asignatura



Add Column

Name:

Type:

Size: Scale:

Default:

Constraints

☒ Primary key ☒ Unique ☐ Null ☒ Index

☐ Check:

Nombre

Add Column

Name:

Type: VARCHAR

Size: Scale:

Default:

Constraints

☐ Primary key ☐ Unique ☐ Null ☐ Index

☐ Check:

OK Cancel

Descripción

Add Column

Name:

Type: VARCHAR

Size: Scale:

Default:

Constraints

☐ Primary key ☐ Unique ☒ Null ☐ Index

☐ Check:

OK Cancel

Tabla con todos los atributos

Create Table

Table name:

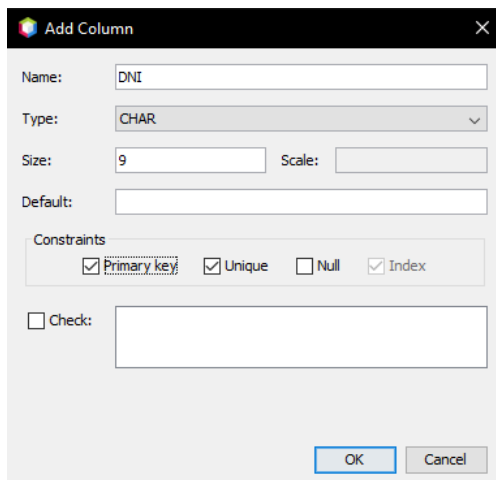
Key	Index	Null	Unique	Column name	Data type	Size
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Id_Asignatura	CHAR	6
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Nombre	VARCHAR	50
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Descripción	VARCHAR	200

Add column
Edit
Remove
Move Up
Move Down

OK Cancel Help

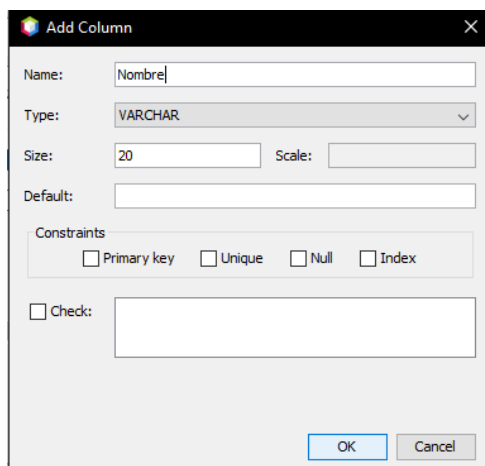
Atributos de la tabla de alumnos:

DNI



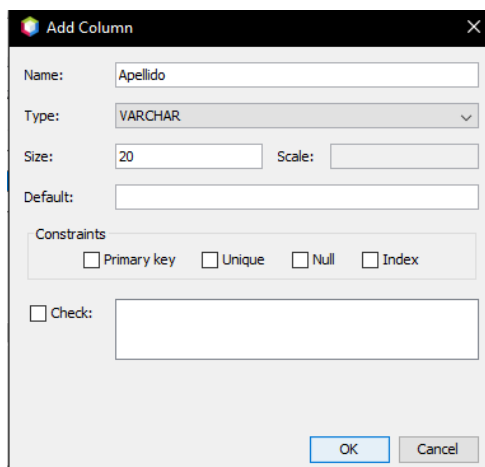
The 'Add Column' dialog box for the 'DNI' attribute. The 'Name' field contains 'DNI'. The 'Type' dropdown is set to 'CHAR'. The 'Size' field is '9' and the 'Scale' field is empty. The 'Default' field is empty. In the 'Constraints' section, the checkboxes for 'Primary key', 'Unique', and 'Index' are checked, while 'Null' is unchecked. There is an unchecked 'Check' checkbox with an empty text area next to it. At the bottom are 'OK' and 'Cancel' buttons.

Nombre



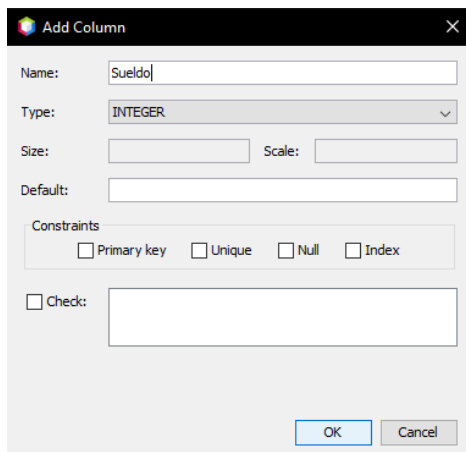
The 'Add Column' dialog box for the 'Nombre' attribute. The 'Name' field contains 'Nombre'. The 'Type' dropdown is set to 'VARCHAR'. The 'Size' field is '20' and the 'Scale' field is empty. The 'Default' field is empty. In the 'Constraints' section, all checkboxes ('Primary key', 'Unique', 'Null', 'Index') are unchecked. There is an unchecked 'Check' checkbox with an empty text area next to it. At the bottom are 'OK' and 'Cancel' buttons.

Apellido



The 'Add Column' dialog box for the 'Apellido' attribute. The 'Name' field contains 'Apellido'. The 'Type' dropdown is set to 'VARCHAR'. The 'Size' field is '20' and the 'Scale' field is empty. The 'Default' field is empty. In the 'Constraints' section, all checkboxes ('Primary key', 'Unique', 'Null', 'Index') are unchecked. There is an unchecked 'Check' checkbox with an empty text area next to it. At the bottom are 'OK' and 'Cancel' buttons.

Sueldo



Add Column

Name:

Type:

Size: Scale:

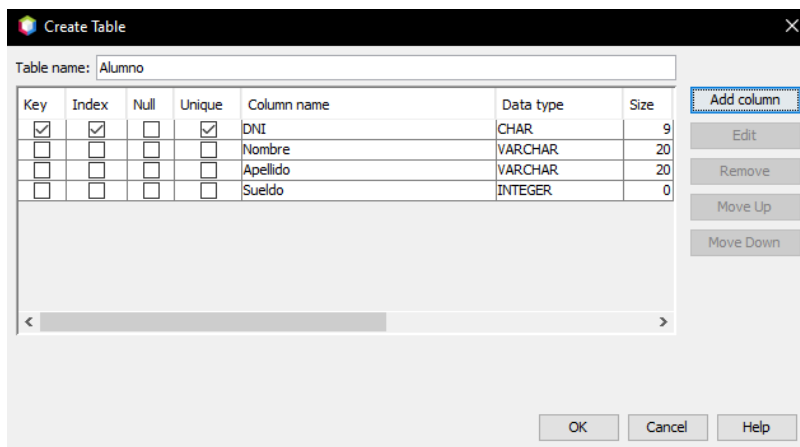
Default:

Constraints

☐ Primary key ☐ Unique ☐ Null ☐ Index

☐ Check:

Tabla con todos los atributos



Create Table

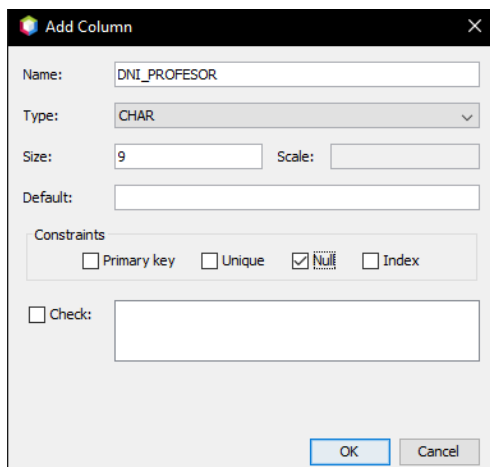
Table name:

Key	Index	Null	Unique	Column name	Data type	Size
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	DNI	CHAR	9
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Nombre	VARCHAR	20
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Apellido	VARCHAR	20
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Sueldo	INTEGER	0

Para crear las relaciones que hay entre las tablas se siguen los siguientes pasos.

En la relación **Imparte** de las tablas **Profesor** y **Asignatura** un profesor imparte una o muchas asignaturas y una asignatura es impartida por un profesor. Por lo que habrá una relación **1:N** y la tabla de las asignaturas tendrá una **clave foránea** del **DNI** del profesor que la imparte.

Se crea el atributo del DNI del profesor en la tabla de asignaturas:



Add Column

Name:

Type:

Size: Scale:

Default:

Constraints

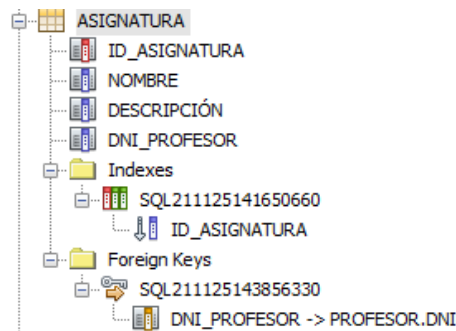
☐ Primary key ☐ Unique ☒ Null ☐ Index

☐ Check:

Se ejecuta la siguiente sentencia:

```
1 ALTER TABLE ADMINISTRADOR.ASIGNATURA
2 ADD FOREIGN KEY (DNI_PROFESOR)
3 REFERENCES ADMINISTRADOR.PROFESOR (DNI) ;
```

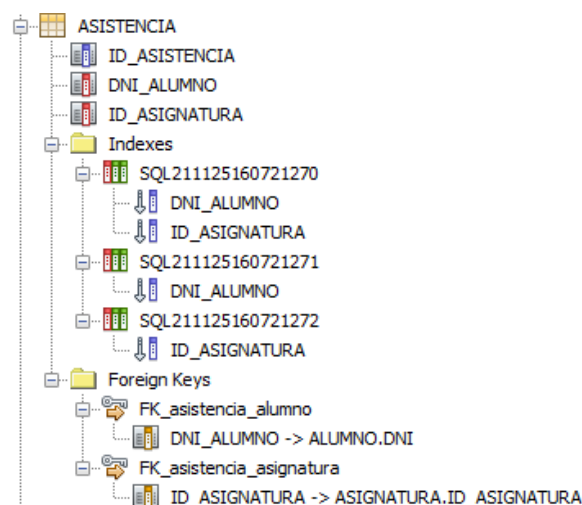
Se refresca la base de datos y se observa la clave foránea que se ha creado haciendo referencia a la clave de la tabla de los profesores.



Para la relación **Asistir** entre la tabla de alumnos y la de asignaturas se ve que un alumno puede asistir a una o muchas asignaturas y una asignatura puede ser asistida por uno o muchos alumnos, por lo que habrá una relación **N:M**. Esta relación se representa creando una nueva tabla llamada **Asistencia** que tendrá dos claves foráneas, el DNI del alumno y el id de la asignatura. La tabla se creará mediante una sentencia SQL de la siguiente manera:

```
1 CREATE TABLE ADMINISTRADOR.ASISTENCIA(
2     id_asistencia char(4) not null,
3     DNI_alumno char(9) not null,
4     id_asignatura char(6) not null,
5     primary key(DNI_alumno, id_asignatura),
6     constraint "FK_asistencia_alumno" foreign key(DNI_alumno) references ADMINISTRADOR.ALUMNO (DNI),
7     constraint "FK_asistencia_asignatura" foreign key(id_asignatura) references ADMINISTRADOR.ASIGNATURA (ID_ASIGNATURA)
8 );
```

Tras ejecutar la sentencia y refrescar la base de datos se observa la nueva tabla creada y sus claves.



Tras esto, se procede a incorporar una serie mínima de datos en las tablas creadas.

Alumnos

```
1 INSERT INTO ALUMNO VALUES ('12345678A', 'Antonio', 'Flores', 600);
2 INSERT INTO ALUMNO VALUES ('12345678B', 'Marcos', 'Fernández', 1000);
3 INSERT INTO ALUMNO VALUES ('45678901C', 'Francisco', 'Domínguez', 2500);
4 INSERT INTO ALUMNO VALUES ('67890123D', 'María', 'Castaño', 1300);
```

#	DNI	NOMBRE	APELLIDO	SUELDO
1	12345678A	Antonio	Flores	600
2	12345678B	Marcos	Fernández	1000
3	45678901C	Francisco	Domínguez	2500
4	67890123D	María	Castaño	1300

Profesores

```
1 INSERT INTO PROFESOR VALUES ('32109876P', 'Jesús', 'Escobar', 2600);
2 INSERT INTO PROFESOR VALUES ('98765432Z', 'Pedro', 'Alonso', 2000);
3 INSERT INTO PROFESOR VALUES ('87654321F', 'Marta', 'Casas', 2300);
4 INSERT INTO PROFESOR VALUES ('54321098K', 'Rocío', 'Pérez', 2300);
```

#	DNI	NOMBRE	APELLIDO	SUELDO
1	32109876P	Jesús	Escobar	2600
2	98765432Z	Pedro	Alonso	2000
3	87654321F	Marta	Casas	2300
4	54321098K	Rocío	Pérez	2300

Asignaturas

```
1 INSERT INTO ASIGNATURA VALUES ('280011', 'Matemáticas', 'Conceptos matemáticos, álgebra, geometría, ecuaciones', '32109876P');
2 INSERT INTO ASIGNATURA VALUES ('280015', 'Lengua y Literatura', 'Historia de la literatura española, sintaxis, ortografía', '98765432Z');
3 INSERT INTO ASIGNATURA VALUES ('280013', 'Inglés', 'Nivel de aprendizaje B1 de la lengua inglesa, listening, writing, reading, vocabulary', '87654321F');
4 INSERT INTO ASIGNATURA VALUES ('280018', 'Física y Química', 'Conocimientos básicos de los fenómenos físicos y químicos', '54321098K');
```

#	ID_ASIGNATURA	NOMBRE	DESCRIPCIÓN	DNI_PROFESOR
1	280011	Matemáticas	Conceptos matemáticos, álgebra, geometría...	32109876P
2	280015	Lengua y Literatura	Historia de la literatura española, sintaxis, ...	98765432Z
3	280013	Inglés	Nivel de aprendizaje B1 de la lengua inglesa...	87654321F
4	280018	Física y Química	Conocimientos básicos de los fenómenos físi...	54321098K

Asistencias

```

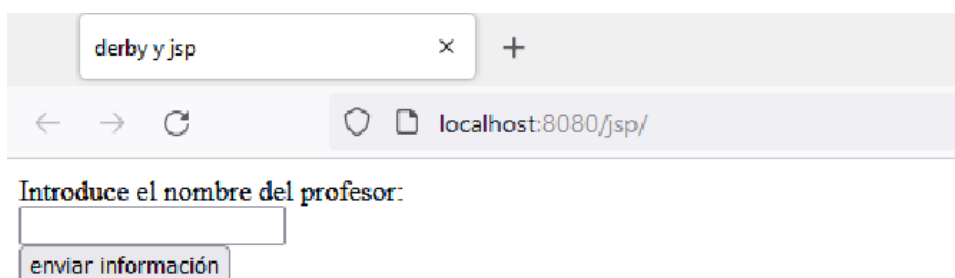
1  INSERT INTO ASISTENCIA VALUES ('1234', '12345678A', '280011');
2  INSERT INTO ASISTENCIA VALUES ('1234', '12345678B', '280011');
3  INSERT INTO ASISTENCIA VALUES ('1234', '45678901C', '280011');
4  INSERT INTO ASISTENCIA VALUES ('1234', '67890123D', '280011');
5
6  INSERT INTO ASISTENCIA VALUES ('1245', '12345678A', '280015');
7  INSERT INTO ASISTENCIA VALUES ('1245', '12345678B', '280015');
8  INSERT INTO ASISTENCIA VALUES ('1245', '45678901C', '280015');
9
10 INSERT INTO ASISTENCIA VALUES ('1250', '12345678A', '280013');
11 INSERT INTO ASISTENCIA VALUES ('1250', '12345678B', '280013');
12
13 INSERT INTO ASISTENCIA VALUES ('1251', '12345678A', '280018');
14 INSERT INTO ASISTENCIA VALUES ('1251', '45678901C', '280018');
15 INSERT INTO ASISTENCIA VALUES ('1251', '67890123D', '280018');

```

#	ID_ASISTENCIA	DNI_ALUMNO	ID_ASIGNATURA
1	1234	12345678A	280011
2	1234	12345678B	280011
3	1234	45678901C	280011
4	1234	67890123D	280011
5	1245	12345678A	280015
6	1245	12345678B	280015
7	1245	45678901C	280015
8	1250	12345678A	280013
9	1250	12345678B	280013
10	1251	67890123D	280018
11	1251	12345678A	280018
12	1251	45678901C	280018

Creación del JSP

Creando un fichero **index.jsp** se puede crear una página web principal, como si de HTML se tratase, en ésta se halla un formulario el cual pide como dato el nombre del profesor.



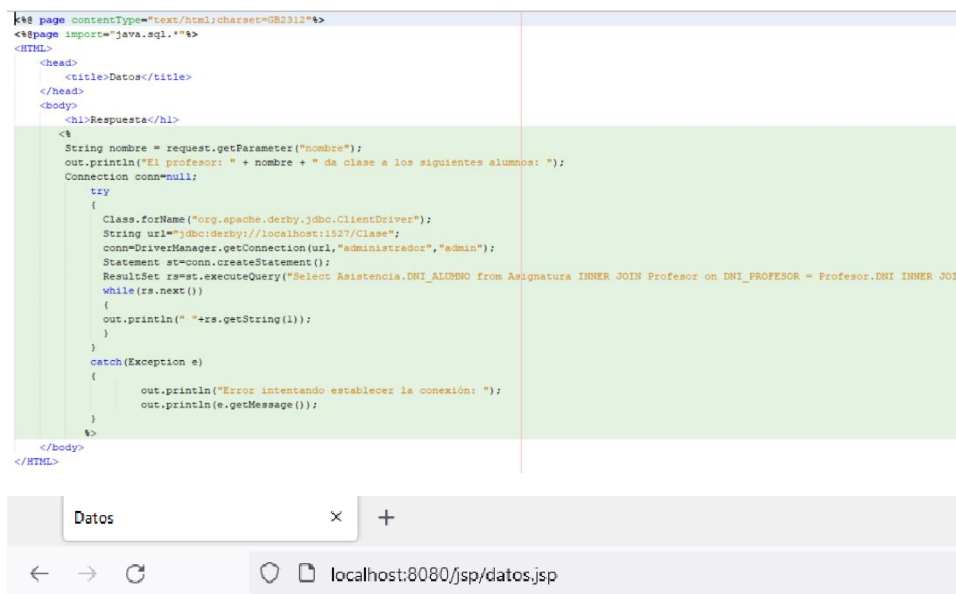
Fichero index.jsp

```

<html>
<head>
<title>derby y jsp</title>
</head>
<body>
<form method="post" action="datos.jsp">
<label for="nombre">Introduce el nombre del profesor:</label><br>
<input type="text" id="name" name="nombre"><br>
<input type="submit" value="enviar información">
</form>
</body>
</html>

```

Al hacer clic en enviar información, esta información se envía al fichero especificado **datos.jsp**, donde es almacenada y tratada. El tratamiento de estos datos consiste en una consulta a la base de datos SQL, se muestran por pantalla los alumnos a los que les da clase el profesor llamado como se haya indicado en la pantalla 1, con el **index.jsp**



```

<%@ page contentType="text/html; charset="GB2312"%>
<%@ page import="java.sql.*"%>
<HTML>
<head>
<title>Datos</title>
</head>
<body>
<h1>Respuesta</h1>
<%
String nombre = request.getParameter("nombre");
out.println("El profesor: " + nombre + " da clase a los siguientes alumnos: ");
Connection conn=null;
try
{
Class.forName("org.apache.derby.jdbc.ClientDriver");
String url="jdbc:derby://localhost:1527/Clase";
conn=DriverManager.getConnection(url,"administrador","admin");
Statement st=conn.createStatement();
ResultSet rs=st.executeQuery("Select Asistencia.DNI_ALUMNO from Asistencia INNER JOIN Profesor on DNI_PROFESOR = Profesor.DNI INNER JOIN");
while(rs.next())
{
out.println(" "+rs.getString(1));
}
}
catch(Exception e)
{
out.println("Error intentando establecer la conexión: ");
out.println(e.getMessage());
}
}
%>
</body>
</HTML>

```

Respuesta

El profesor: Pedro da clase a los siguientes alumnos: 12345678A 12345678B 45678901C



Exportación de la base de datos

Para exportar la base de datos se puede realizar de dos maneras como se explica en el primer apartado.

La primera sería creando ficheros SQL de las sentencias con las que se crea la base de datos, se insertan los datos y se visualizan estos. Los ficheros serían los siguientes:

```
crear_tablas.sql X
crear_tablas.sql
1 CREATE TABLE PROFESOR (
2     DNI char(9) not null primary key,
3     NOMBRE varchar(20) not null,
4     APELLIDO varchar(20) not null,
5     SUELDO integer not null
6 );
7
8 CREATE TABLE ALUMNO (
9     DNI char(9) not null primary key,
10    NOMBRE varchar(20) not null,
11    APELLIDO varchar(20) not null,
12    SUELDO integer not null
13 );
14
15 CREATE TABLE ASIGNATURA (
16     ID_ASIGNATURA char(6) not null primary key,
17     NOMBRE varchar(50) not null,
18     DESCRIPCION varchar(200),
19     DNI_PROFESOR char(9) not null,
20     constraint "FK_asignatura_profesor" FOREIGN KEY (DNI_PROFESOR) REFERENCES PROFESOR(DNI)
21 );
22
23 CREATE TABLE ASISTENCIA (
24     ID_ASISTENCIA char(4) not null,
25     DNI_ALUMNO char(9) not null,
26     ID_ASIGNATURA char(6) not null,
27     primary key(DNI_ALUMNO, ID_ASISTENCIA),
28     constraint "FK_asistencia_alumno" FOREIGN KEY (DNI_ALUMNO) REFERENCES ALUMNO(DNI),
29     constraint "FK_asistencia_asignatura" FOREIGN KEY (ID_ASIGNATURA) REFERENCES ASIGNATURA(ID_ASIGNATURA)
30 );
31

insertar_datos.sql
1 INSERT INTO ALUMNO VALUES ('12345678A', 'Antonio', 'Flores', 600);
2 INSERT INTO ALUMNO VALUES ('12345678B', 'Marcos', 'Fernandez', 1000);
3 INSERT INTO ALUMNO VALUES ('45678901C', 'Francisco', 'Dominguez', 2500);
4 INSERT INTO ALUMNO VALUES ('67890123D', 'Maria', 'Castano', 1300);
5
6 INSERT INTO PROFESOR VALUES ('32109876P', 'Jesus', 'Escobar', 2600);
7 INSERT INTO PROFESOR VALUES ('98765432Z', 'Pedro', 'Alonso', 2000);
8 INSERT INTO PROFESOR VALUES ('87654321F', 'Marta', 'Casas', 2300);
9 INSERT INTO PROFESOR VALUES ('54321098K', 'Rocio', 'Perez', 2300);
10
11 INSERT INTO ASIGNATURA VALUES ('280011', 'Matematicas', 'Conceptos matematicos, algebra, geometria, ecuaciones', '32109876P');
12 INSERT INTO ASIGNATURA VALUES ('280015', 'Lengua y Literatura', 'Historia de la literatura espanola, sintaxis, ortografia', '98765432Z');
13 INSERT INTO ASIGNATURA VALUES ('280013', 'Ingles', 'Nivel de aprendizaje B1 de la lengua inglesa, listening, writting, reading, vocabulary', '87654321F');
14 INSERT INTO ASIGNATURA VALUES ('280018', 'Fisica y Quimica', 'Conocimientos basicos de lso fenomenos fisicos y quimicos', '54321098K');
15
16 INSERT INTO ASISTENCIA VALUES ('1234', '12345678A', '280011');
17 INSERT INTO ASISTENCIA VALUES ('1234', '12345678B', '280011');
18 INSERT INTO ASISTENCIA VALUES ('1234', '45678901C', '280011');
19 INSERT INTO ASISTENCIA VALUES ('1234', '67890123D', '280011');
20
21 INSERT INTO ASISTENCIA VALUES ('1245', '12345678A', '280015');
22 INSERT INTO ASISTENCIA VALUES ('1245', '12345678B', '280015');
23 INSERT INTO ASISTENCIA VALUES ('1245', '45678901C', '280015');
24
25 INSERT INTO ASISTENCIA VALUES ('1250', '12345678A', '280013');
26 INSERT INTO ASISTENCIA VALUES ('1250', '12345678B', '280013');
27
28 INSERT INTO ASISTENCIA VALUES ('1251', '12345678A', '280018');
29 INSERT INTO ASISTENCIA VALUES ('1251', '45678901C', '280018');
30 INSERT INTO ASISTENCIA VALUES ('1251', '67890123D', '280018');
```



```
visualizar_datos.sql
1  SELECT * FROM ALUMNO;
2  SELECT * FROM PROFESOR;
3  SELECT * FROM ASIGNATURA;
4  SELECT * FROM ASISTENCIA;
5
```

La segunda es importando directamente la base de datos creada en **Derby** (conociendo el usuario y contraseña que el creador estableció).

Calles Esteban > AppData > Roaming > NetBeans > Derby > Clase >				
Nombre	Fecha de modificación	Tipo	Tamaño	
log	25/11/2021 13:52	Carpeta de archivos		
seg0	25/11/2021 16:07	Carpeta de archivos		
README_DO_NOT_TOUCH_FILES	25/11/2021 13:52	Documento de te...	1 KB	
service	25/11/2021 13:52	Archivo PROPERTI...	1 KB	



Sesiones

El objeto **Session** se utiliza para rastrear una sesión de cliente entre las solicitudes recibidas. El propio **JSP** utiliza la interfaz **HttpSession** proporcionada por el servlet, que permite identificar a un usuario a través de una solicitud de una página, la visita a un sitio web o almacenando información sobre el usuario en cuestión.

De forma predeterminada, las **JSP** tienen habilitado el seguimiento en sesiones y se crea una instancia de un nuevo objeto **HttpSession** para cada nuevo cliente automáticamente, de esta manera en los códigos implementados a continuación no se muestran dichas instancias o creaciones, sino que se accede a ellas directamente.

Se ha implementado el código proporcionado, tanto del html como de los archivos correspondientes jsp. El **index.html** es el encargado de recibir el campo insertado, en este caso el nombre del usuario, y llamar a **sesionEje.jsp** encargado de realizar la acción correspondiente a pulsar el botón **enviar información**.



Ejemplo de sesión

Por favor, introduce tu nombre:

Dicho archivo recoge el campo insertado, considerado como parte de la petición, y lo guarda como atributo dentro del objeto **session** de manera que pueda ser accedido en varias páginas relacionadas con la actual.

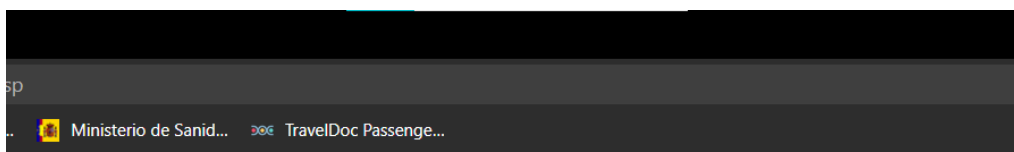


Ejemplo de Sesión

Donde quieres ir!!! [Ir a Página 1](#) [Ir a Página 2](#)

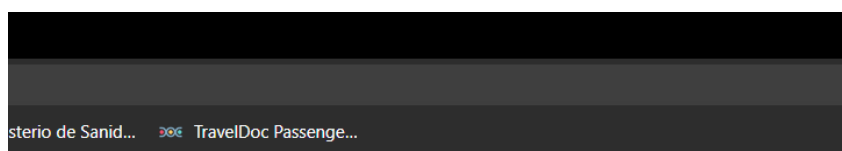


Dependiendo de la opción elegida, bien sea **Ir a Página 1** o **Ir a Página 2**, se muestra una página u otra. Cada una de ellas obtiene el atributo guardado anteriormente en el objeto **session** para poder emplearlo y mostrarlo en pantalla con el mensaje personalizado.



Ejemplo de Sesión

Hola, Bianca Bienvenido a la página 1



Ejemplo de Sesión

Hola, Bianca Bienvenido a la página 2



Conclusiones

En esta práctica se ha aprendido como hacer una aplicación web desde 0 en **Netbeans** utilizando **GlassFish Server** y **Java 8**. Se ha hecho páginas **JSP** básicas para aprender su funcionamiento. Se conectado el proyecto con las bases de datos creadas en **Derby**, conexión por `jdbc:derby`, para mostrar los datos de sus tablas en la **WEB**. Por último, se ha utilizado el objeto **session** para guardar información relevante del usuario. Todos los apartados de esta práctica se han completado con éxito.