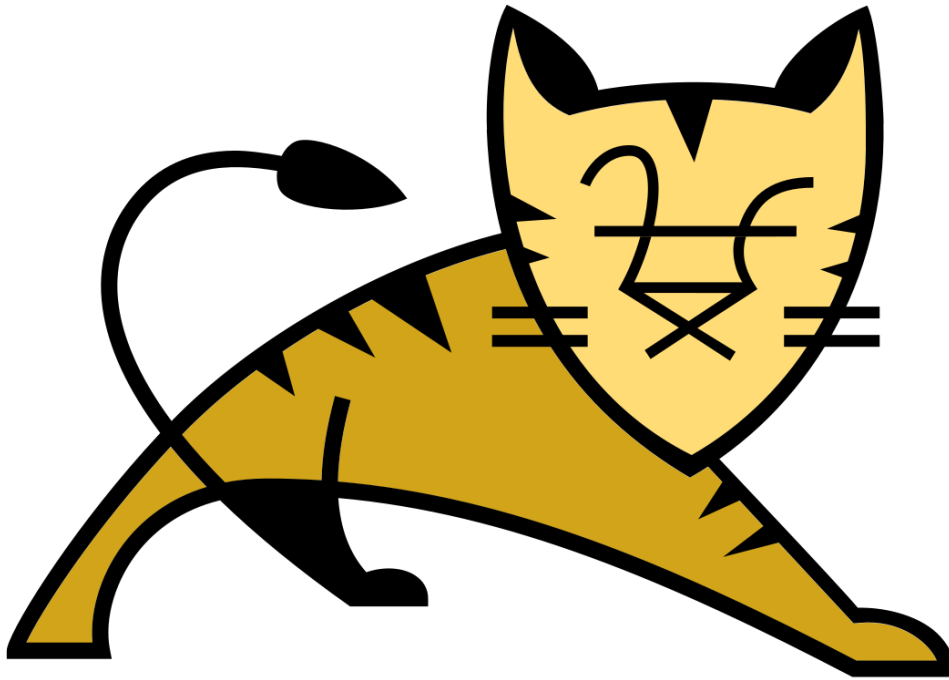


# Instalación y uso de Tomcat



Cristian Leandro Sánchez Mego

## Índice:

1.	Instalación de Tomcat:	4
a.	Descargar el instalable de la página:	4
b.	Abrir el .exe del programa y seguir los pasos hasta instalar el XAMPP:	4
c.	Abrir el panel principal del XAMPP y activar todos los servicios:	5
d.	Comprobar el funcionamiento de los servicios activados:	5
e.	Instalar el JDK:	6
f.	Instalar el Tomcat:	6
g.	Configurar variables de entorno:	10
•	Ingresar a la configuración avanzada:	10
•	Cambiar las variables:	11
h.	Comprobar funcionamiento:	14
•	Funcionamiento en XAMPP:	15
2.	Instalación de Eclipse y empleo del IDE junto a Tomcat:	17
a.	Instalación de Eclipse IDE:	17
b.	Configuración de Eclipse junto a Tomcat:	19
c.	Comprobación del funcionamiento de Tomcat dentro de Eclipse:	22
3.	Despliegue de Servlets:	24
a.	Crear el proyecto:	24
b.	Estructura del proyecto:	25
c.	Añadir un nuevo archivo:	25
d.	Exportar y mostrar en servidor:	26
•	Desplegar en Eclipse:	28
e.	Despliegue de un servlet:	29
4.	Servlet con acceso a base de datos con JDBC	32
a.	Instalación y habilitación del MySQL y phpMyAdmin:	32
b.	Implementación y consulta a la base de datos	35
c.	Resultados:	39
5.	Sesiones con Servlets:	40
a.	Crear proyecto y establecer por cookies:	40
b.	Despliegue y resultados:	42
c.	Despliegue sin uso de cookies (usando el ID):	43
6.	Pool de conexiones con JNDI y Servlets:	44
a.	Creación de proyecto y definición de recurso JNDI:	44
b.	Codificación:	45
c.	Comprobación de resultados:	51

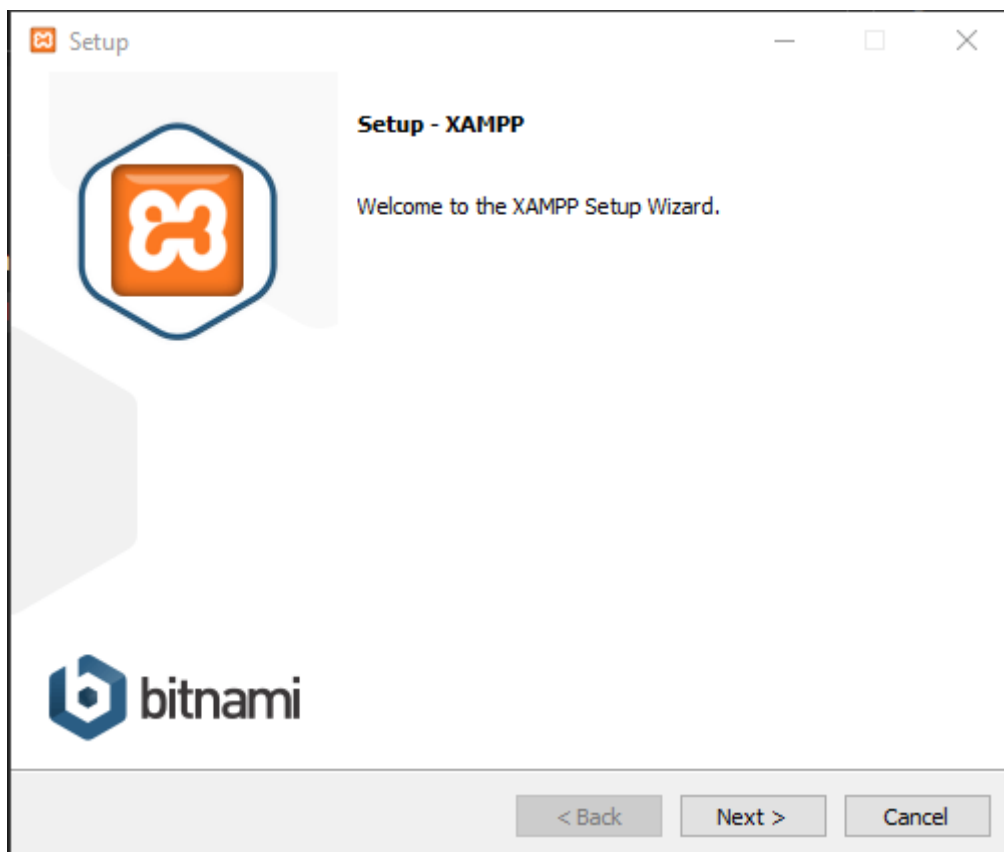
7.	Creación de Páginas con JSP: .....	54
a.	Primer ejemplo con expresiones básicas: .....	54
b.	Segundo ejemplo con contador de veces visitadas: .....	56
c.	Tercer ejemplo con conexión a la base de datos: .....	57
8.	Conclusiones: .....	59
9.	Bibliografía: .....	60

## 1. Instalación de Tomcat:

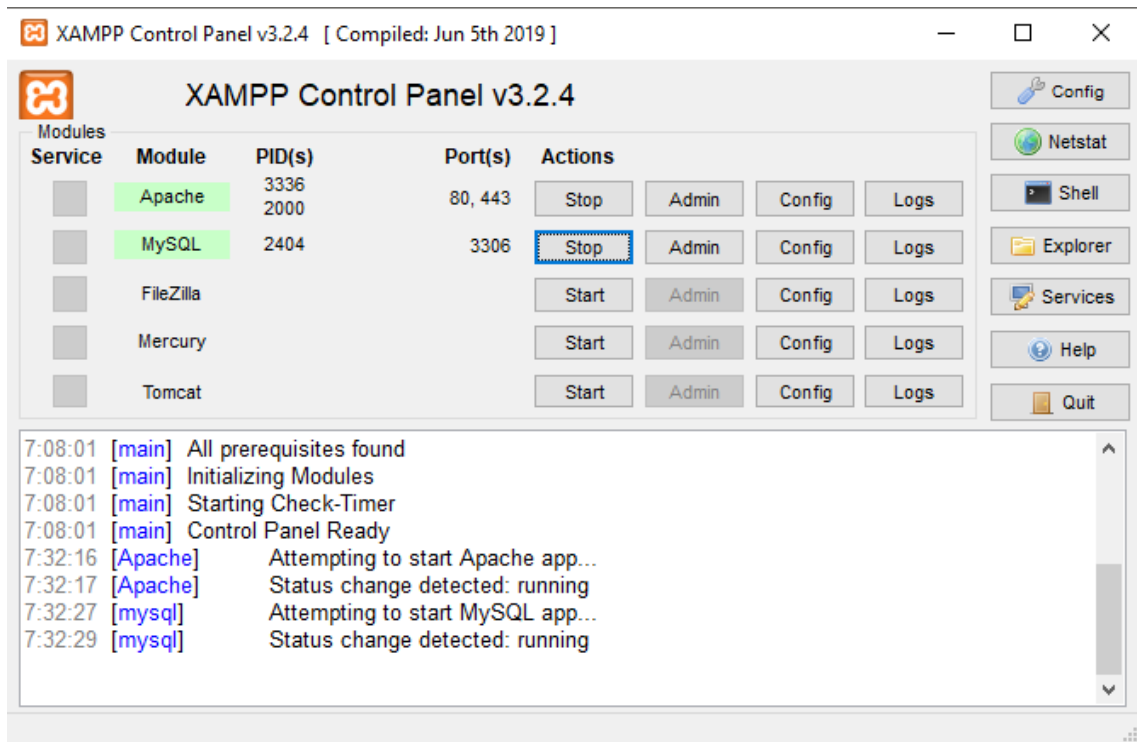
- Descargar el instalable de la página:



- Abrir el .exe del programa y seguir los pasos hasta instalar el XAMPP:

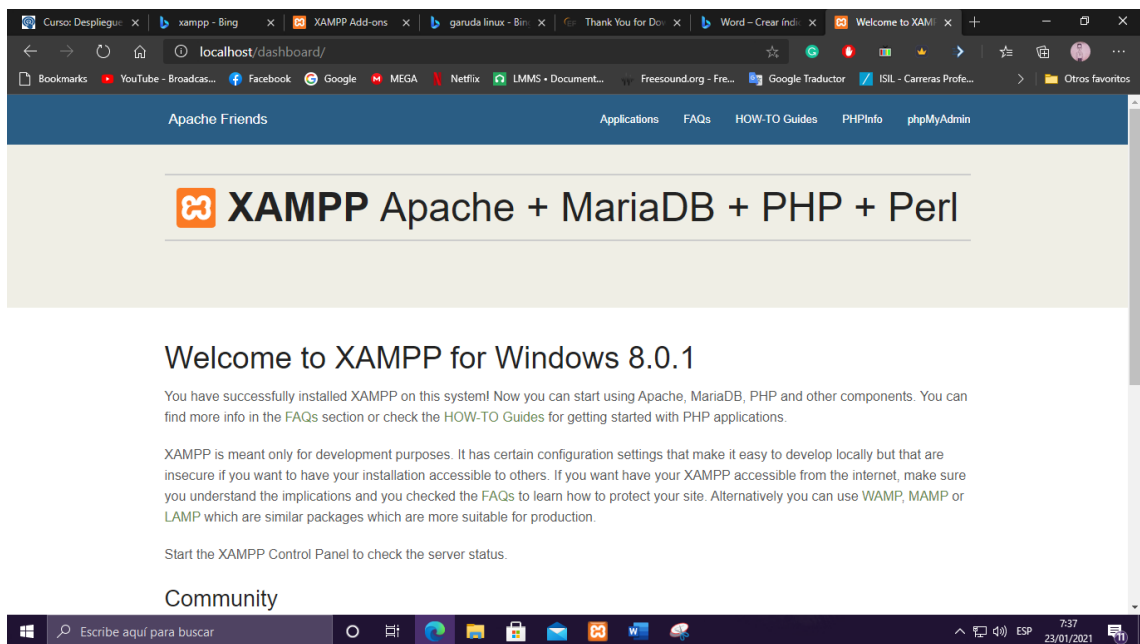


c. Abrir el panel principal del XAMPP y activar todos los servicios:

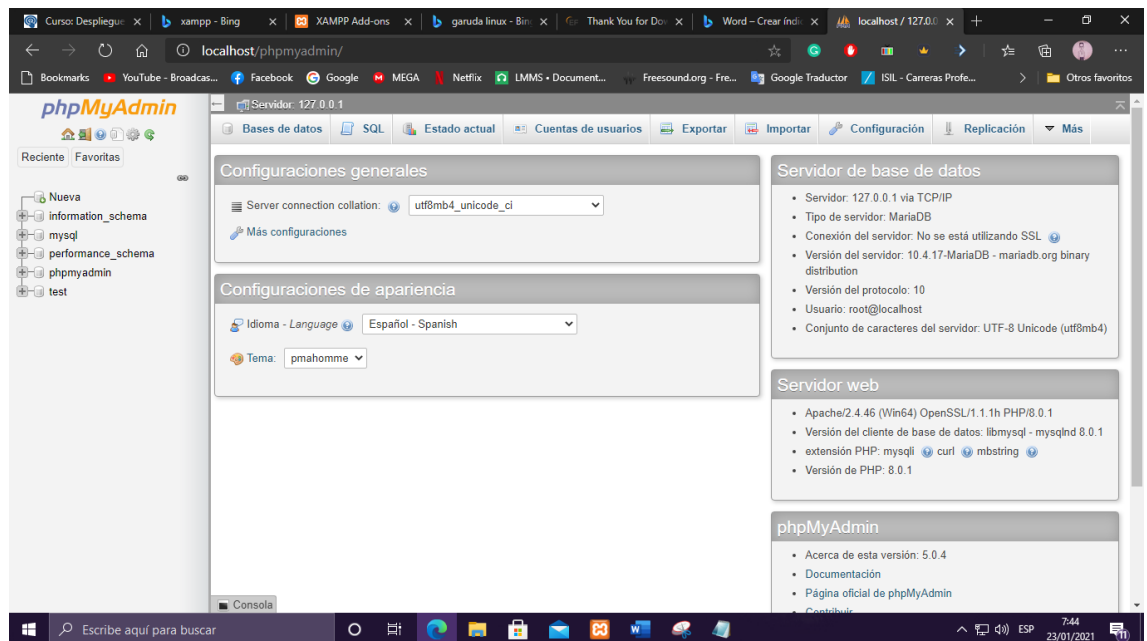


Con esto se activa el Apache, la base de datos y también tiene un acceso para empezar Tomcat cuándo ya esté instalado.

d. Comprobar el funcionamiento de los servicios activados:

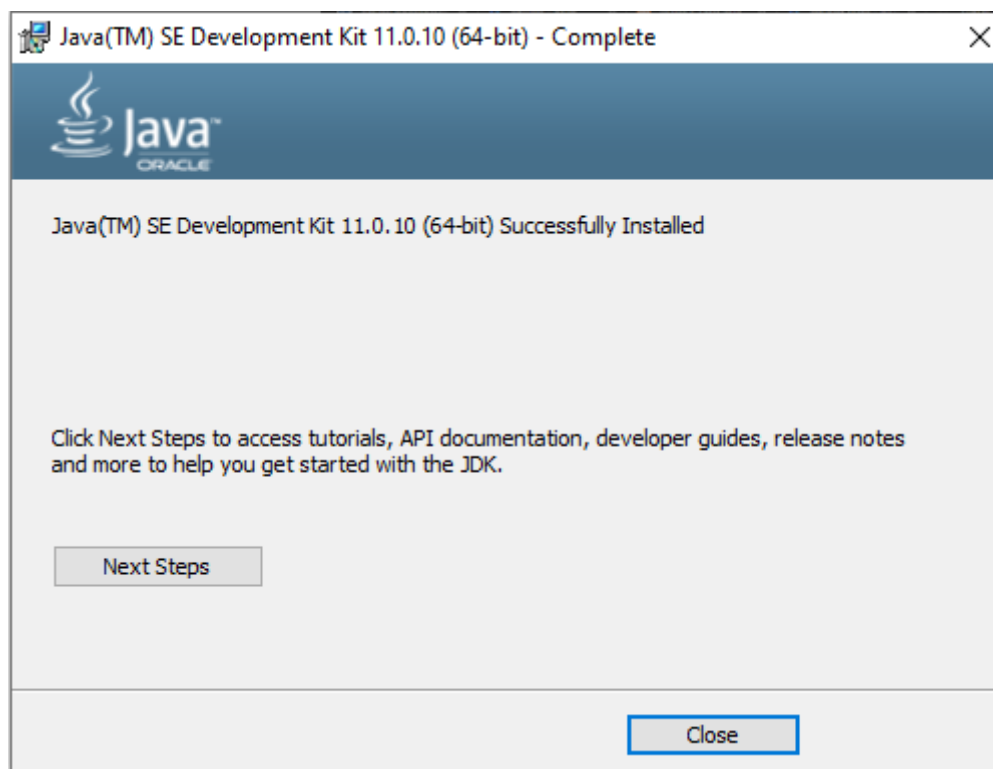


Con esto se comprueba el funcionamiento del Apache.



Haciendo click en la pantalla previa, en la pestaña “*PhpmyAdmin*”, se puede ingresar al IDE de MySQL para manejar las bases de datos.

e. Instalar el JDK:



f. Instalar el Tomcat:

La página oficial de Tomcat tiene los instaladores necesarios, en este caso trabajaré con la versión 9 del mismo.

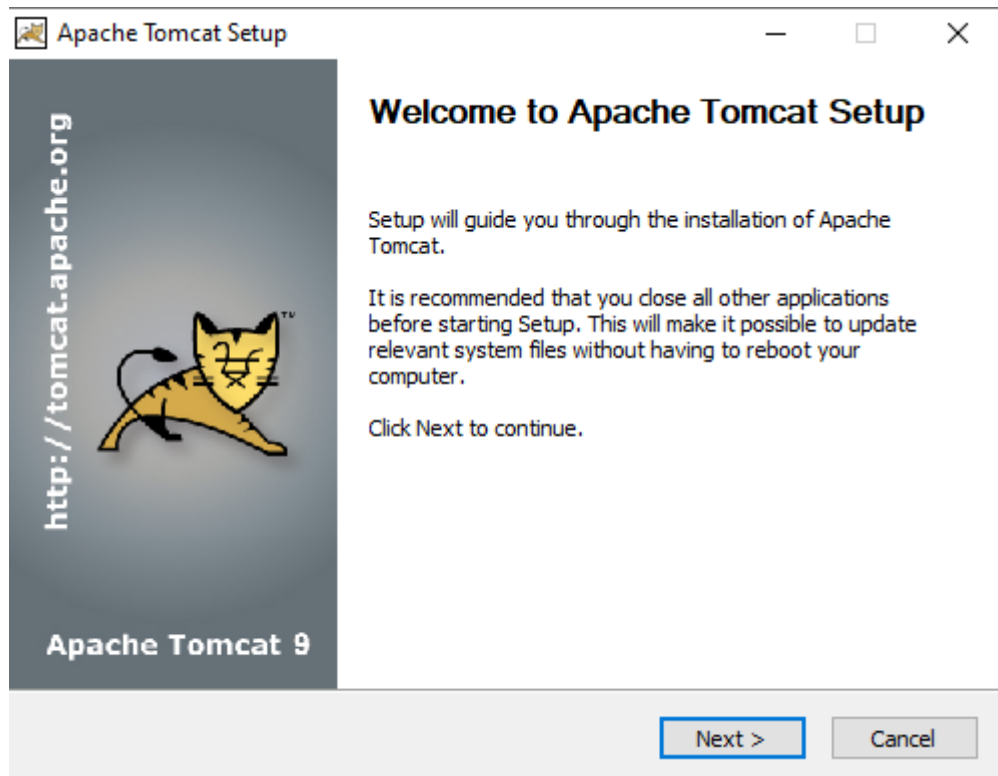
9.0.41

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

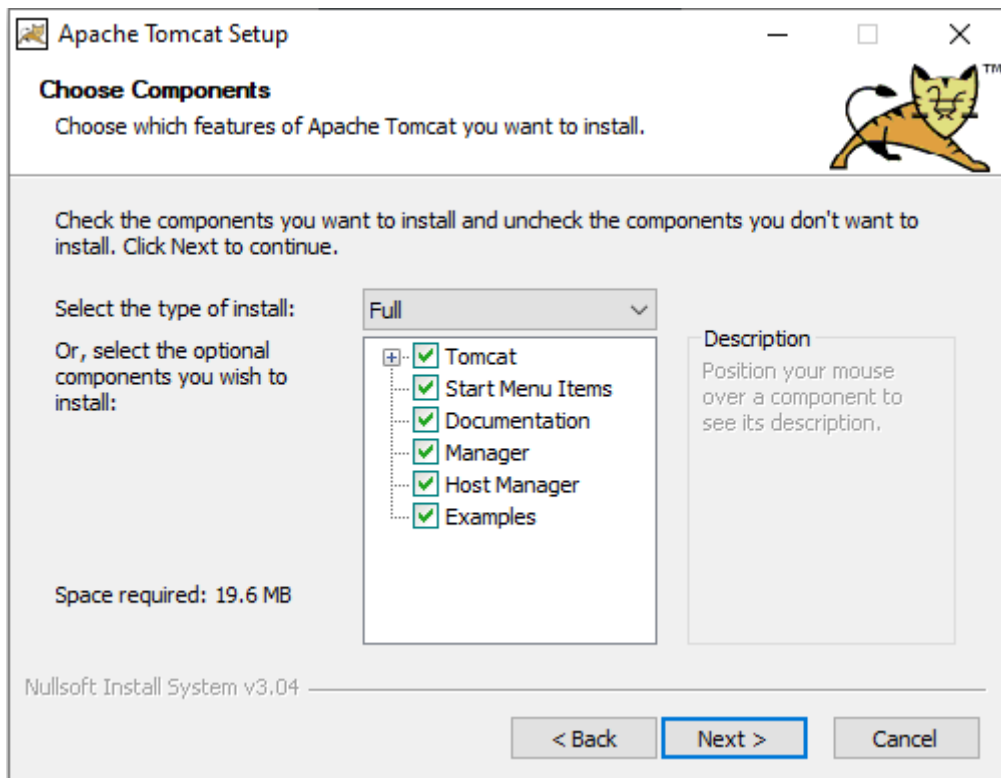
#### Binary Distributions

- Core:
  - [zip](#) (pgp, sha512)
  - [tar.gz](#) (pgp, sha512)
  - [32-bit Windows zip](#) (pgp, sha512)
  - [64-bit Windows zip](#) (pgp, sha512)
  - [32-bit/64-bit Windows Service Installer](#) (pgp, sha512)
- Full documentation:
  - [tar.gz](#) (pgp, sha512)
- Deployer:
  - [zip](#) (pgp, sha512)
  - [tar.gz](#) (pgp, sha512)
- Embedded:
  - [tar.gz](#) (pgp, sha512)
  - [zip](#) (pgp, sha512)

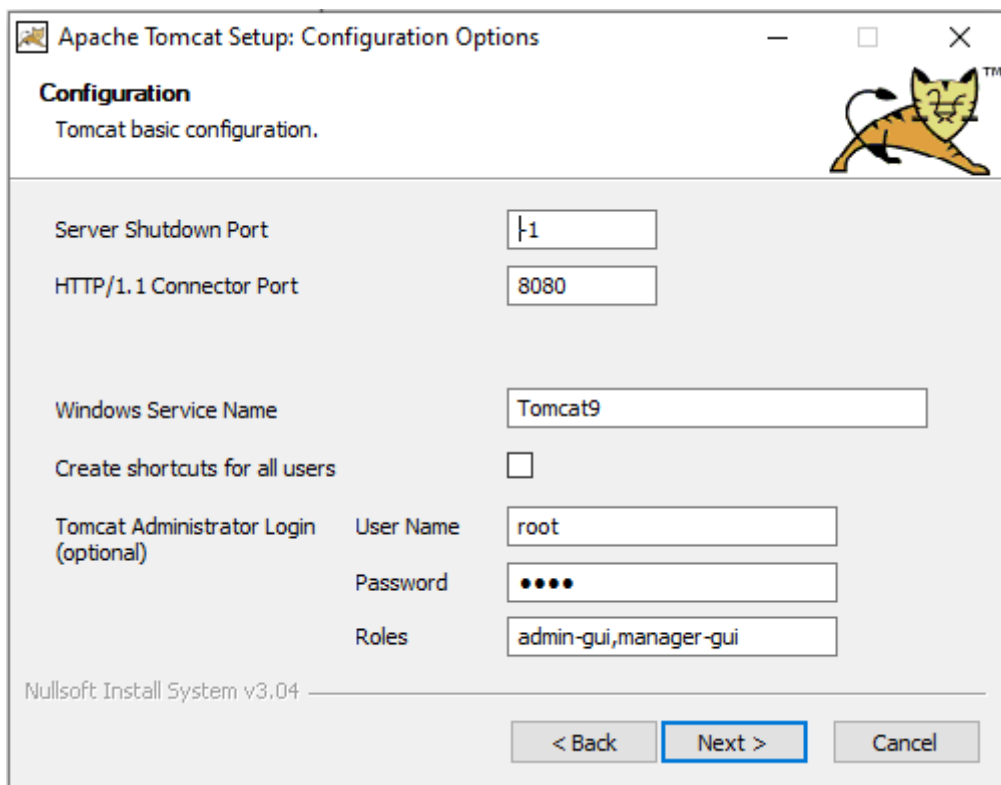
Se abrirá un instalador:



Se acepta la licencia. En este caso elegiré trabajar con la versión full para poder aprovechar los ejemplos.

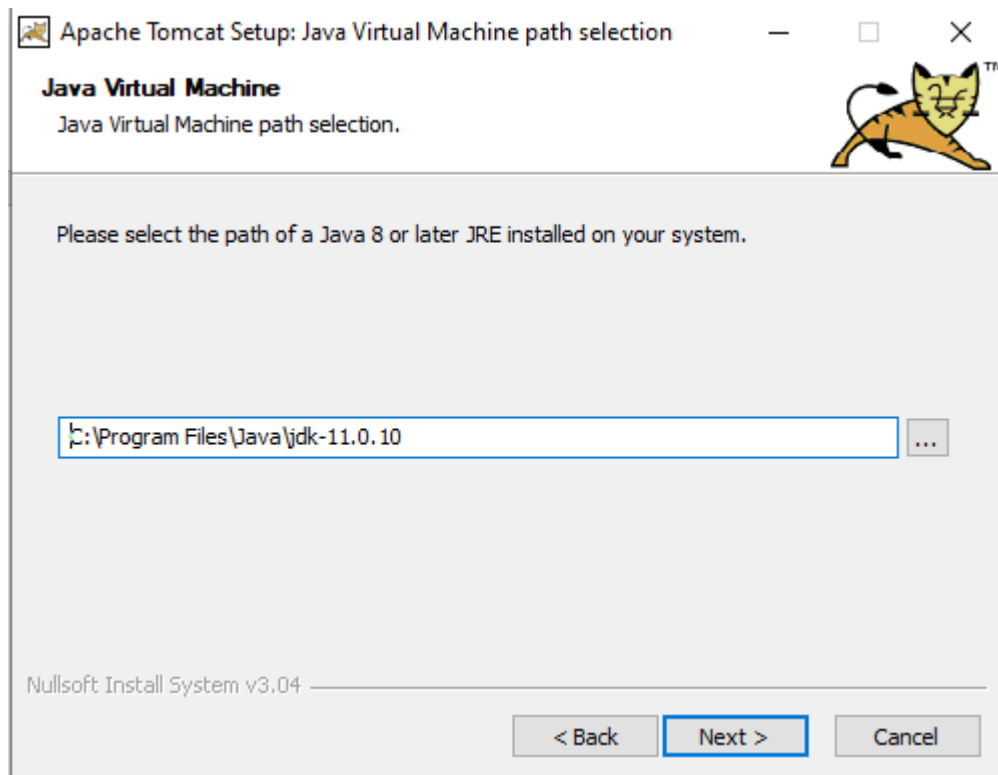


En la siguiente pestaña preguntará por el puerto que se va a utilizar para trabajar. Al ser una experiencia de prueba y la primera vez que se usa el Tomcat, se tiene que dejar tal cual. Lo que sí se debe agregar es el usuario y la contraseña que, nuevamente, al ser de prueba, se utilizará una fácil. Para proyectos sofisticados y el uso profesional se recomienda usar contraseñas robustas.

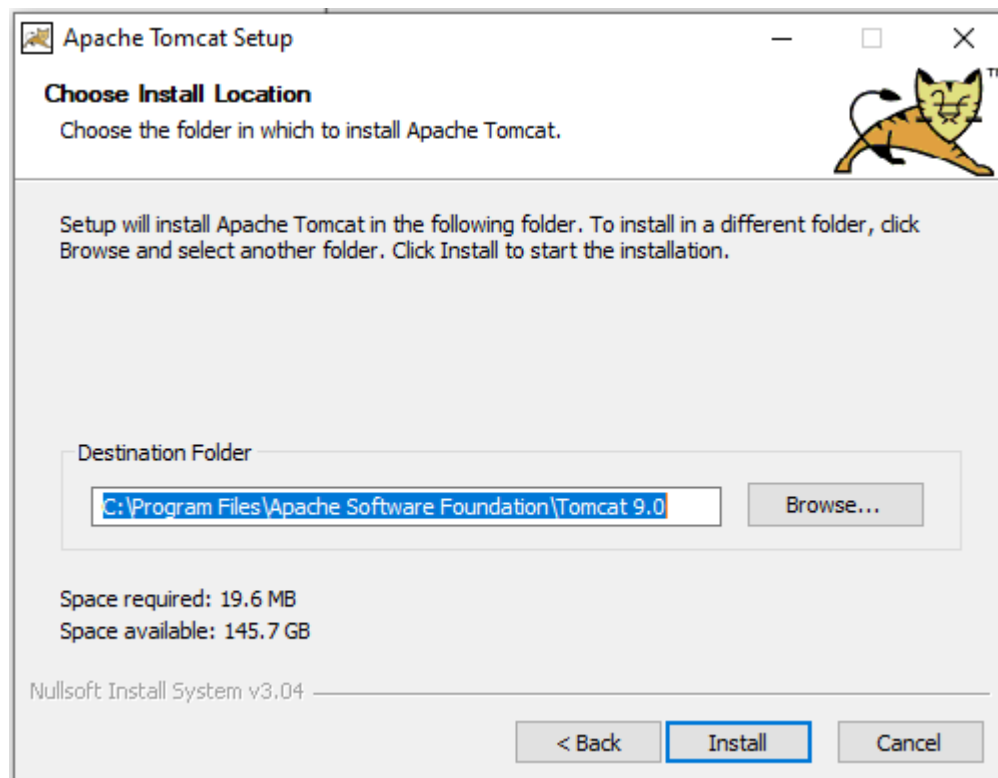




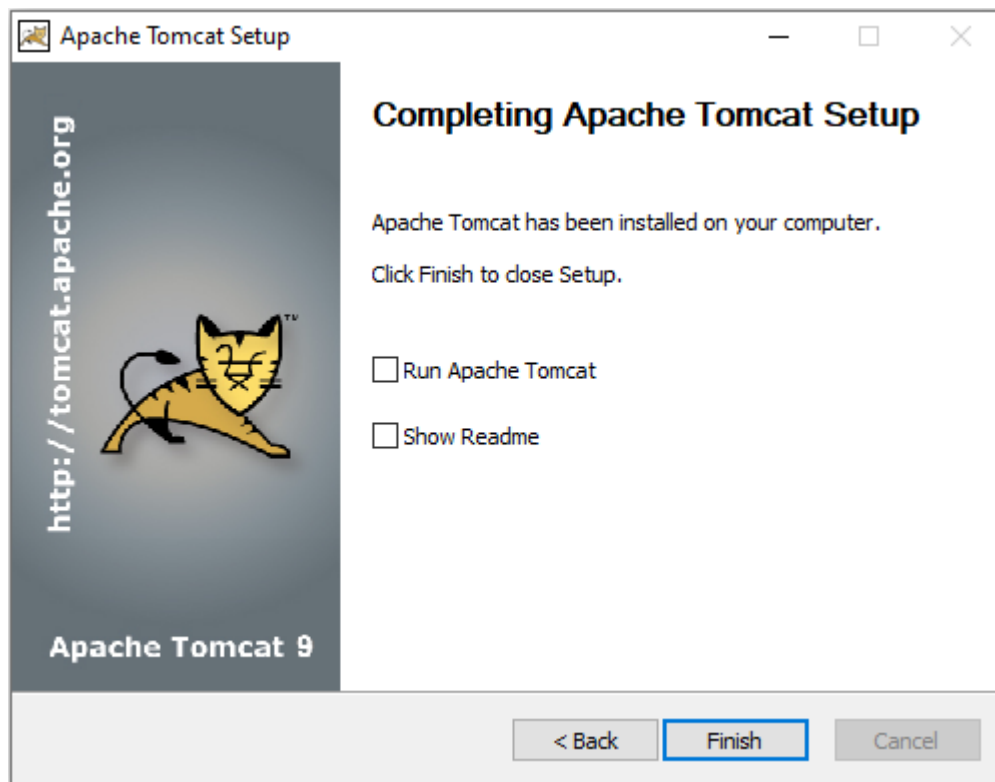
En la siguiente pestaña preguntará por la ubicación de JDK, la cuál si nos reconocida automáticamente, se puede especificar su ubicación en la PC.



Se ubica en un punto específico o por defecto (yo escogí el por defecto) y se procede a instalar.



Antes de concluir con la instalación, se desmarcan las opciones de empezar el programa, ya que todavía hay ciertos elementos más por adicionar a la instalación.



#### g. Configurar variables de entorno:

Para trabajar con el Tomcat se necesita trabajar con la variable CATALINA\_HOME y cómo valor se inserta la dirección dónde se encuentra instalado el Tomcat.

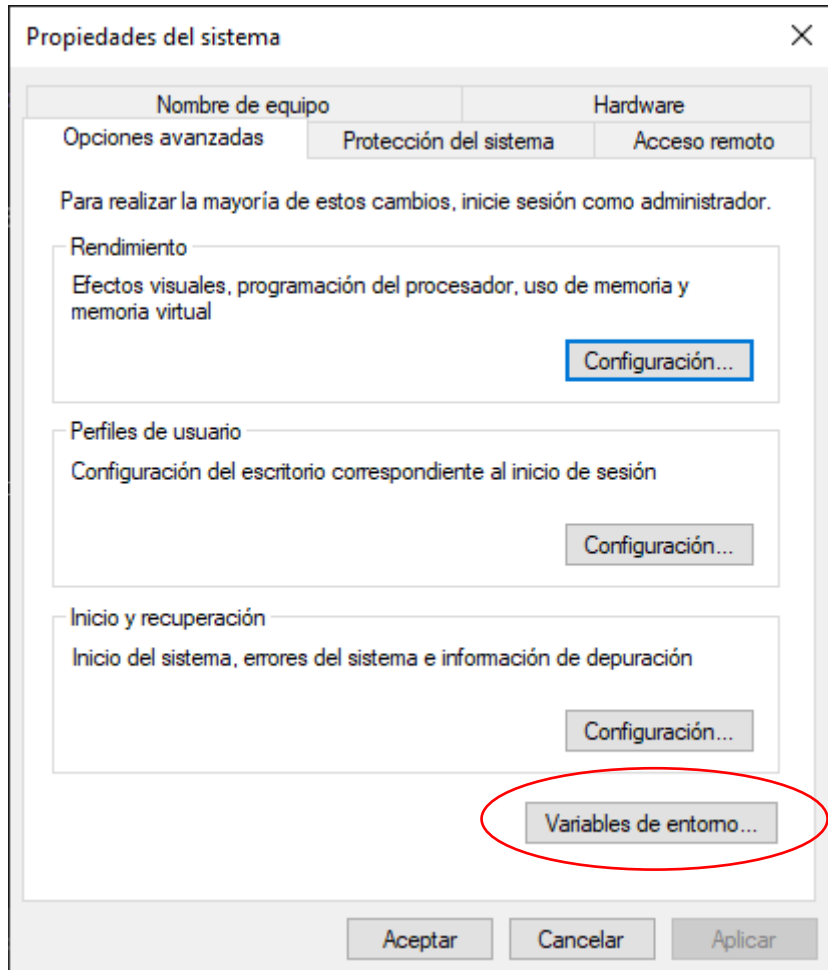
- Ingresar a la configuración avanzada:

En propiedades del equipo, en el panel derecho, está la opción de “configuración avanzada”.

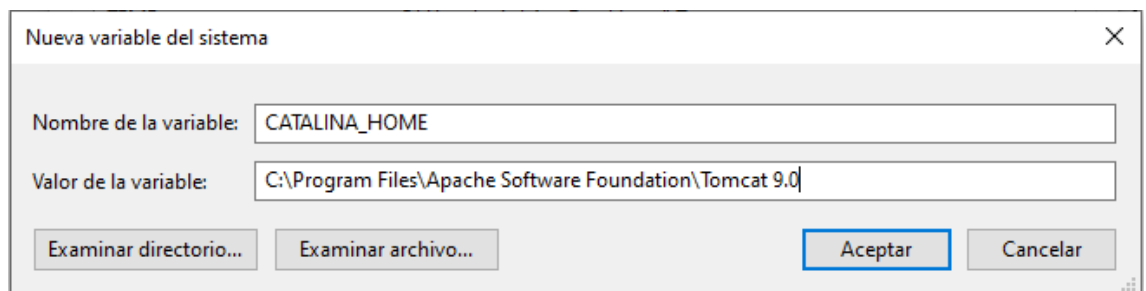
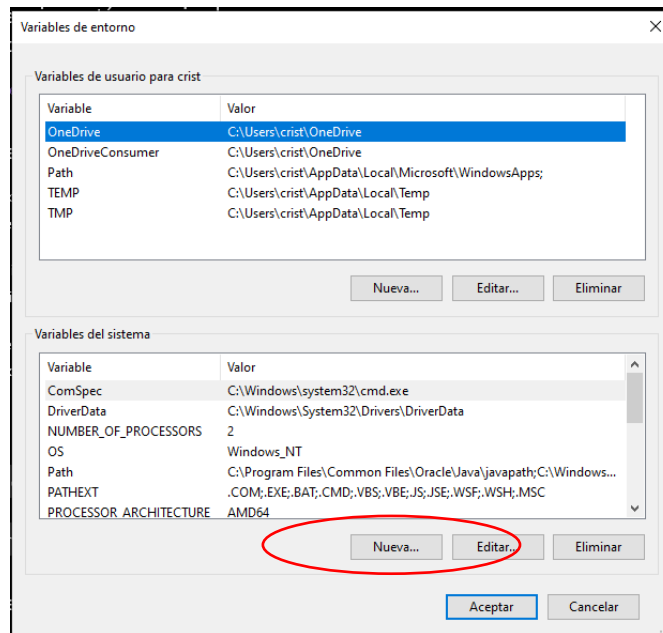


- Cambiar las variables:

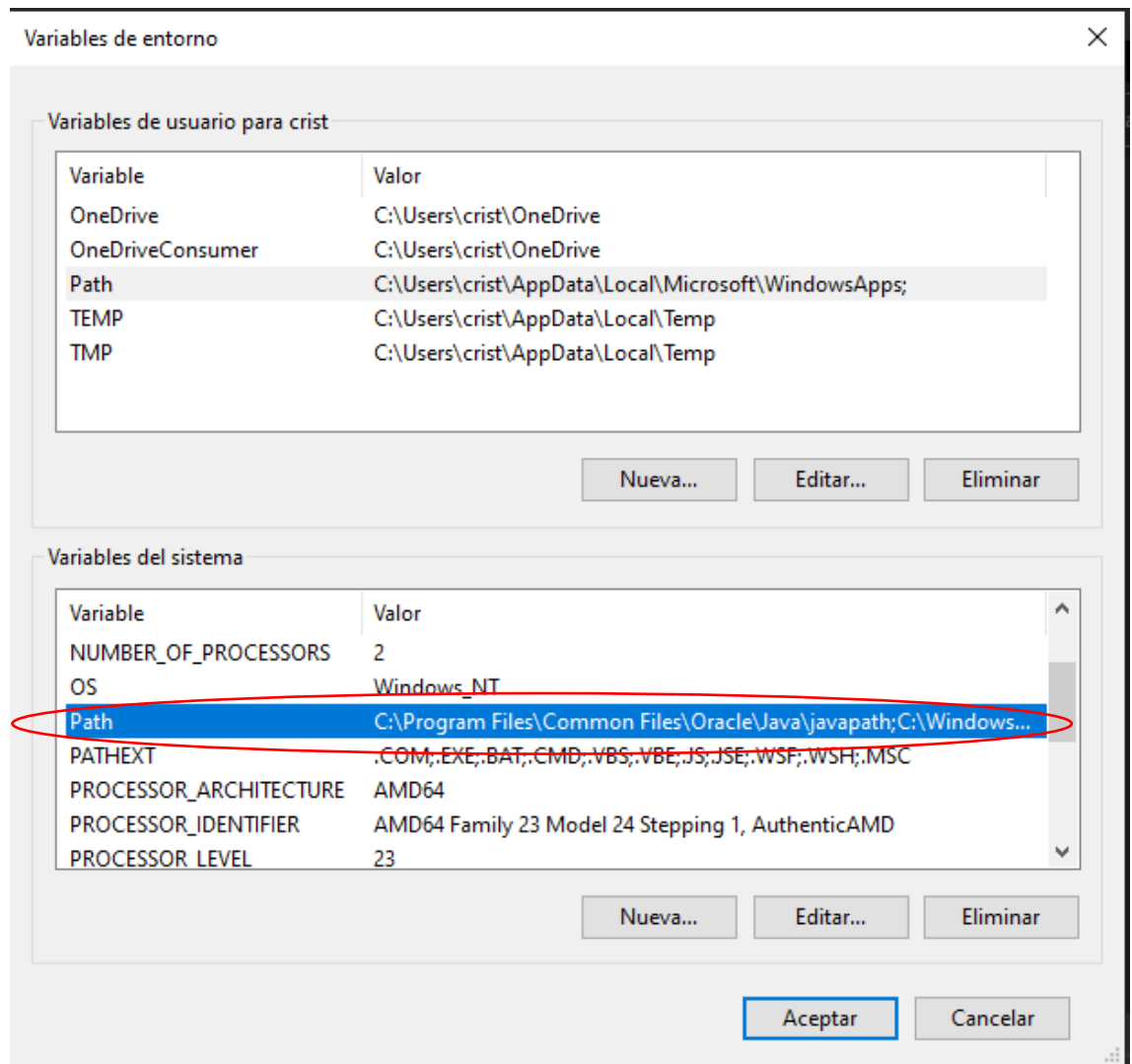
Aparecerá una ventana que se llama “Propiedades del Sistema”, el cual nos permitirá cambiar los valores del sistema.

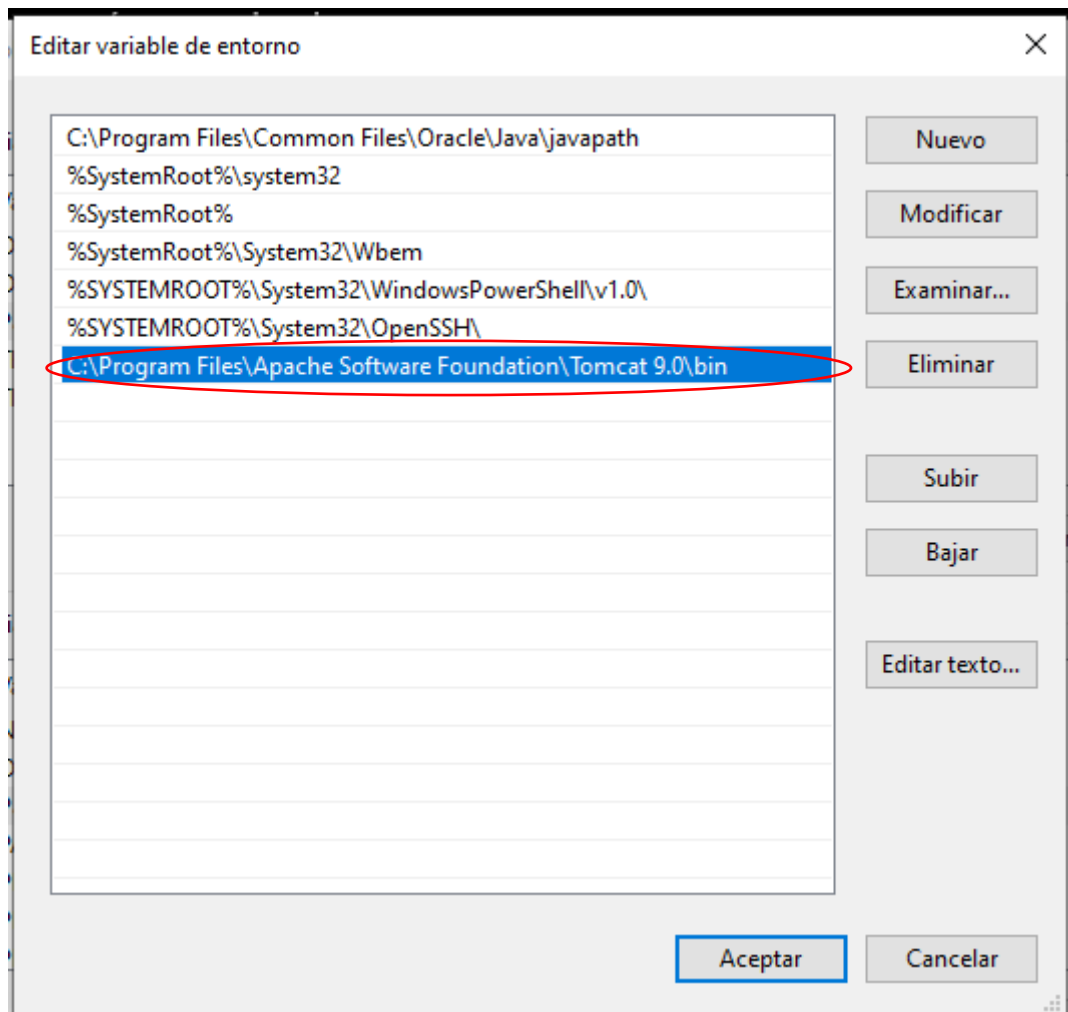


Haciendo click en “Nueva”, de la parte inferior se ingresa la nueva variable “CATALINA\_HOME” y se da de valor el directorio dónde se encuentra el Tomcat.



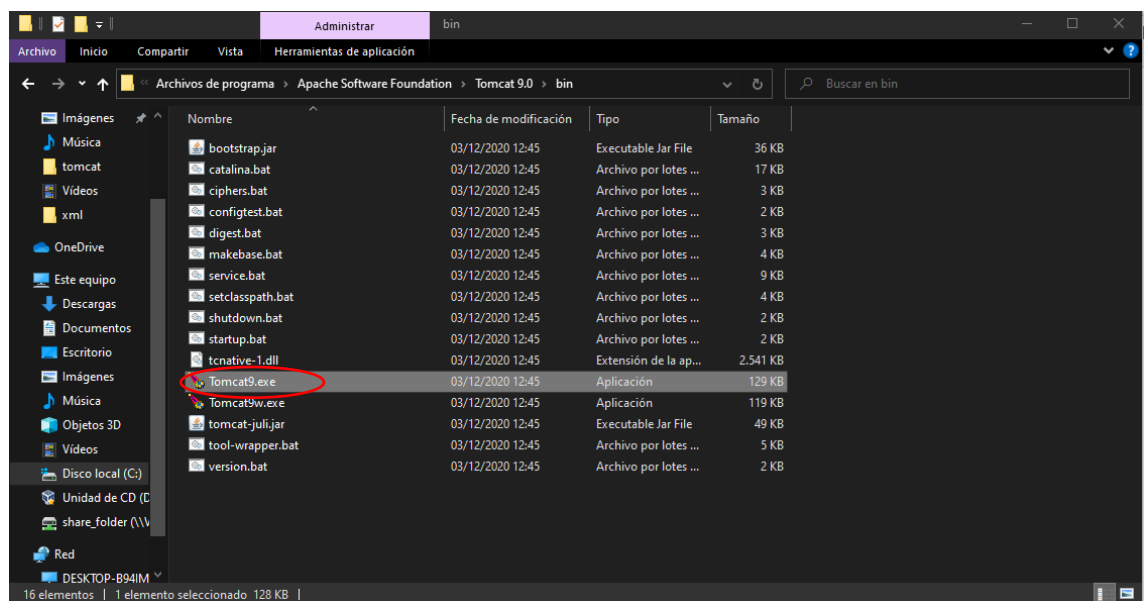
Y a la ruta del PATH se le agrega el /bin del directorio del Tomcat:





#### h. Comprobar funcionamiento:

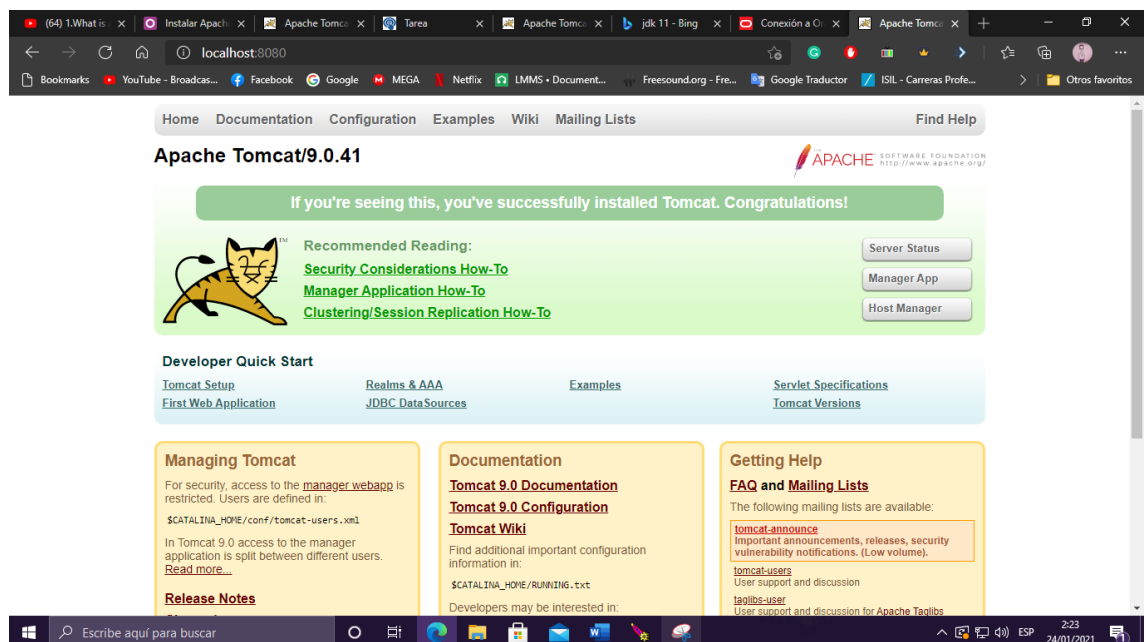
Nos dirigimos a la carpeta bin del directorio del Tomcat y luego ejecutamos cómo administrador la aplicación con su mismo nombre.



Comprobamos que todas las funcionalidades se activen en el terminal que se abre y el puerto dónde se encuentra el Tomcat para abrirlo:

```
C:\Program Files\Apache Software Foundation\Tomcat 9.0\bin\Tomcat9.exe
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
24-Jan-2021 02:20:14.042 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
-Djava.util.logging.config.file=C:\Program Files\Apache Software Foundation\Tomcat 9.0\conf\logging.properties
24-Jan-2021 02:20:14.042 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
--add-opens=java.base/java.lang=ALL-UNNAMED
24-Jan-2021 02:20:14.042 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
--add-opens=java.base/java.io=ALL-UNNAMED
24-Jan-2021 02:20:14.042 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
--add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
24-Jan-2021 02:20:14.042 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
exit
24-Jan-2021 02:20:14.042 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
abort
24-Jan-2021 02:20:14.042 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
-Xms128m
24-Jan-2021 02:20:14.042 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument:
-Xmx256m
24-Jan-2021 02:20:14.480 INFO [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent Cargada la biblioteca
nativa APR de Apache Tomcat [1.2.25] con la versi|n APR [1.7.0].
24-Jan-2021 02:20:14.480 INFO [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent Capacidades APR: IPv6
[true], enviar fichero [true], aceptar filtros [false], aleatorio [true].
24-Jan-2021 02:20:14.480 INFO [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent APR/OpenSSL configura
tion: useAprConnector [false], useOpenSSL [true]
24-Jan-2021 02:20:14.854 INFO [main] org.apache.catalina.core.AprLifecycleListener.initializeSSL OpenSSL inicializado c
orrectamente [OpenSSL 1.1.1g 21 Apr 2020]
24-Jan-2021 02:20:16.606 INFO [main] org.apache.coyote.AbstractProtocol.init Inicializando el manejador de protocolo ["
http-nio-8080"]
24-Jan-2021 02:20:18.349 INFO [main] org.apache.catalina.startup.Catalina.load Server initialization in [7645] millise
conds
```

Abrimos el Tomcat en el navegador:



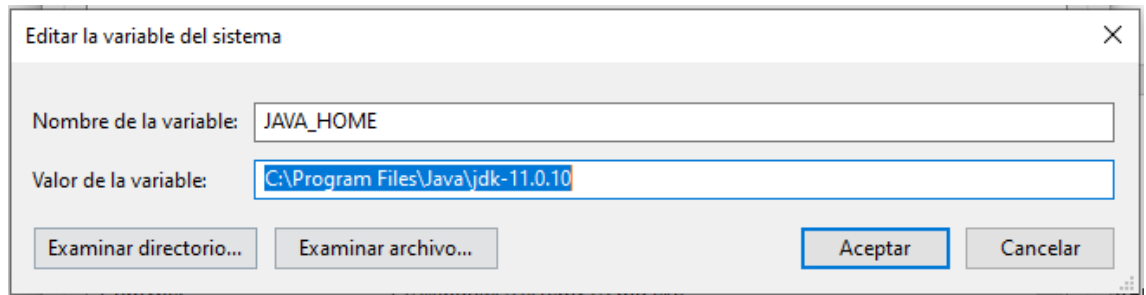
Para poder desactivar el servidor, en el terminal se presiona la combinación de teclas “Ctrl+C” y se apagará automáticamente el servidor.

Posteriormente comprobaremos su funcionamiento desde el XAMPP.

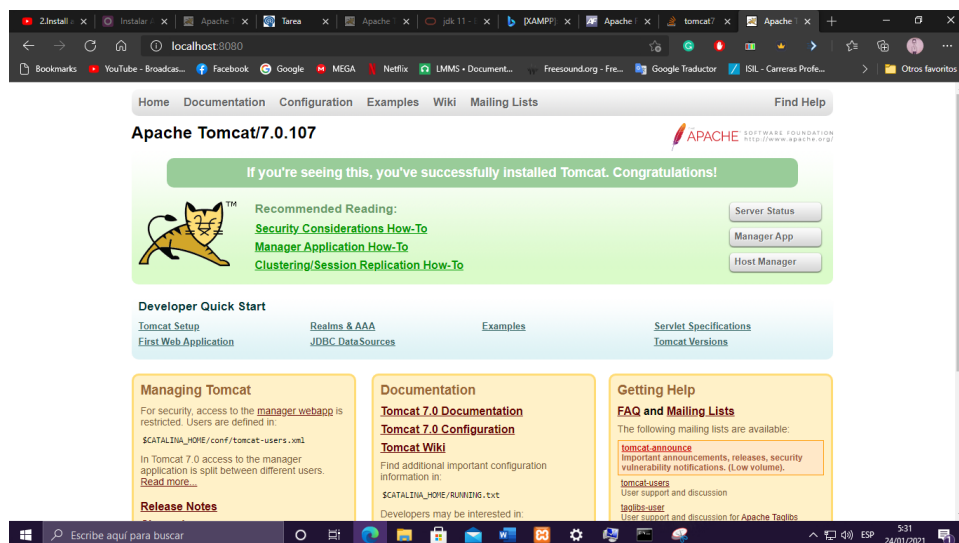
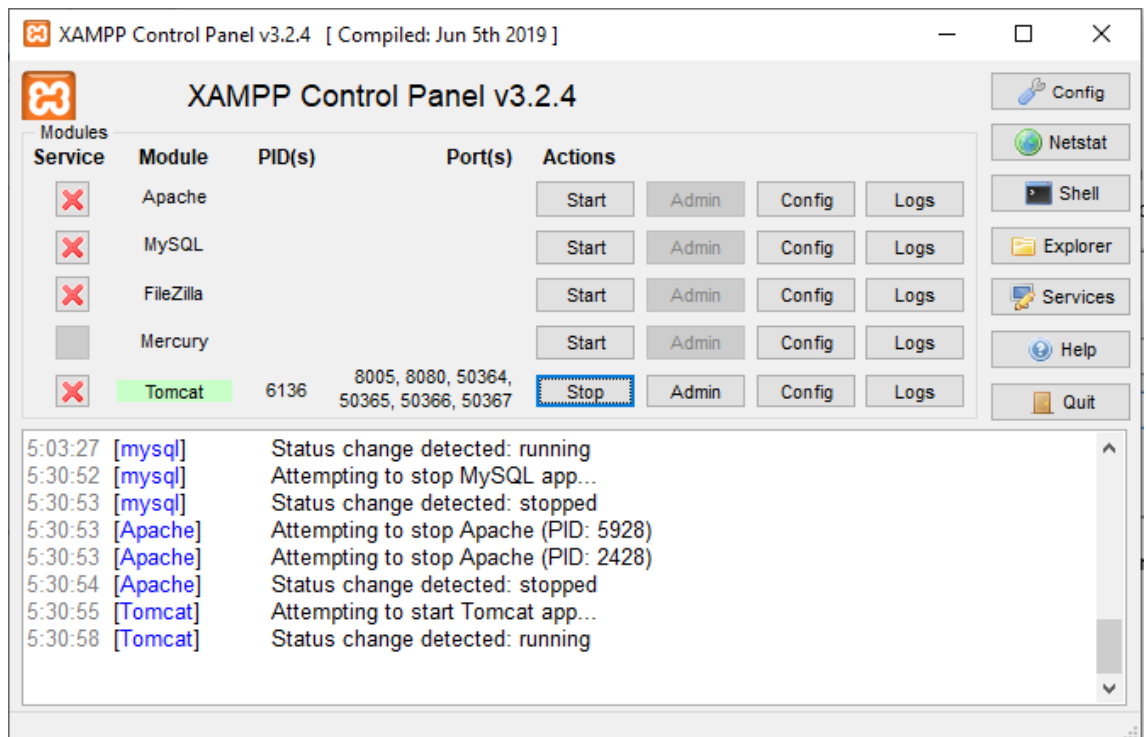
- **Funcionamiento en XAMPP**

XAMPP tiene su propia versión de XAMPP (la 7 en específico) que también se puede activar con estos previos pasos.

Primero, necesitamos agregar una nueva variable de entorno llamada JAVA\_HOME, el cuál redireccionará directamente al JDK previamente aplicado.



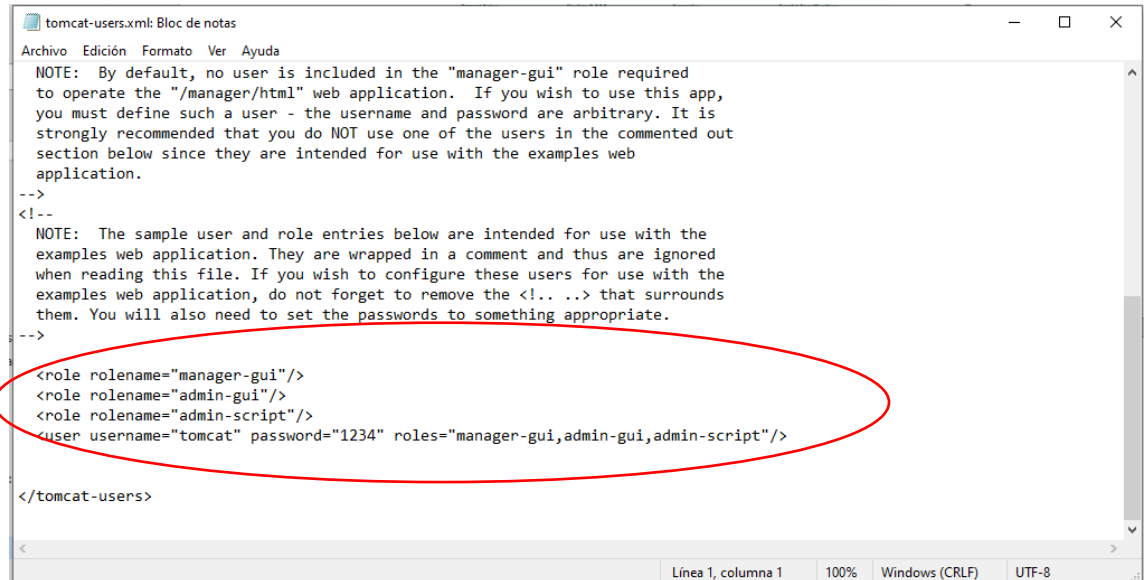
Luego de esto, podemos abrir el XAMPP y ya podrá empezar el Tomcat:





Adicionalmente podemos agregar los permisos de administrador para los distintos controles en Tomcat, tales como *manager-gui*, *admin-gui* y *admin-script*. Al ser una prueba, a un solo usuario le añadiremos todos estos roles, pero es recomendable que un solo usuario no posea todos los permisos para uso profesional.

Se puede hacer este cambio en la pestaña de Config, perteneciente al Tomcat. Elegimos la opción de “tomcat-users” y dentro del archivo descomentamos el sector de roles de esta manera (las comentarios están encapsulados por `<!-- -->`).



```
tomcat-users.xml: Bloc de notas
Archivo Edición Formato Ver Ayuda
NOTE: By default, no user is included in the "manager-gui" role required
to operate the "/manager/html" web application. If you wish to use this app,
you must define such a user - the username and password are arbitrary. It is
strongly recommended that you do NOT use one of the users in the commented out
section below since they are intended for use with the examples web
application.
-->
<!--
NOTE: The sample user and role entries below are intended for use with the
examples web application. They are wrapped in a comment and thus are ignored
when reading this file. If you wish to configure these users for use with the
examples web application, do not forget to remove the <!-- --> that surrounds
them. You will also need to set the passwords to something appropriate.
-->
<role rolename="manager-gui"/>
<role rolename="admin-gui"/>
<role rolename="admin-script"/>
<user username="tomcat" password="1234" roles="manager-gui,admin-gui,admin-script"/>

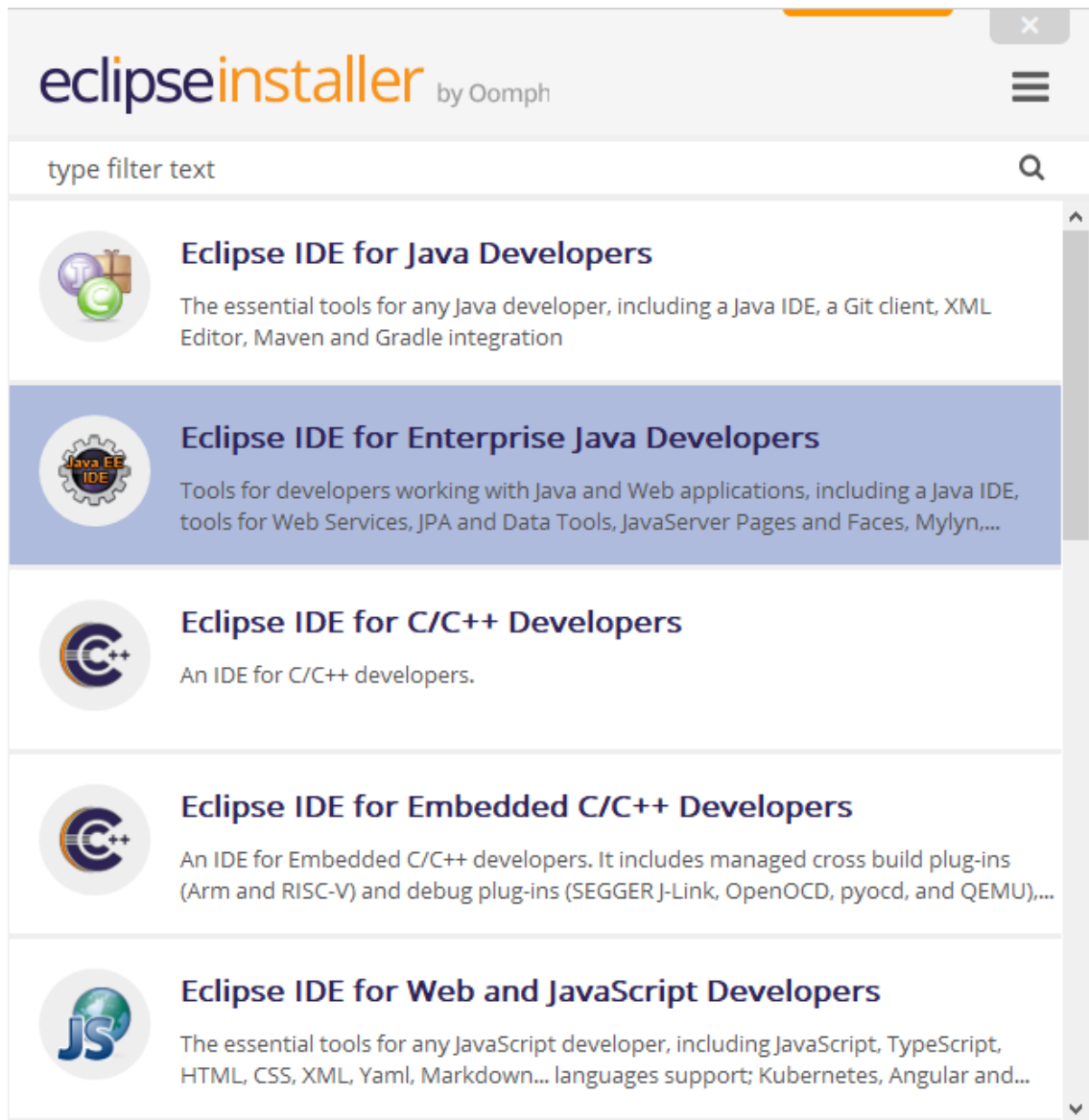
</tomcat-users>
```

## 2. Instalación de Eclipse y empleo del IDE junto a Tomcat:

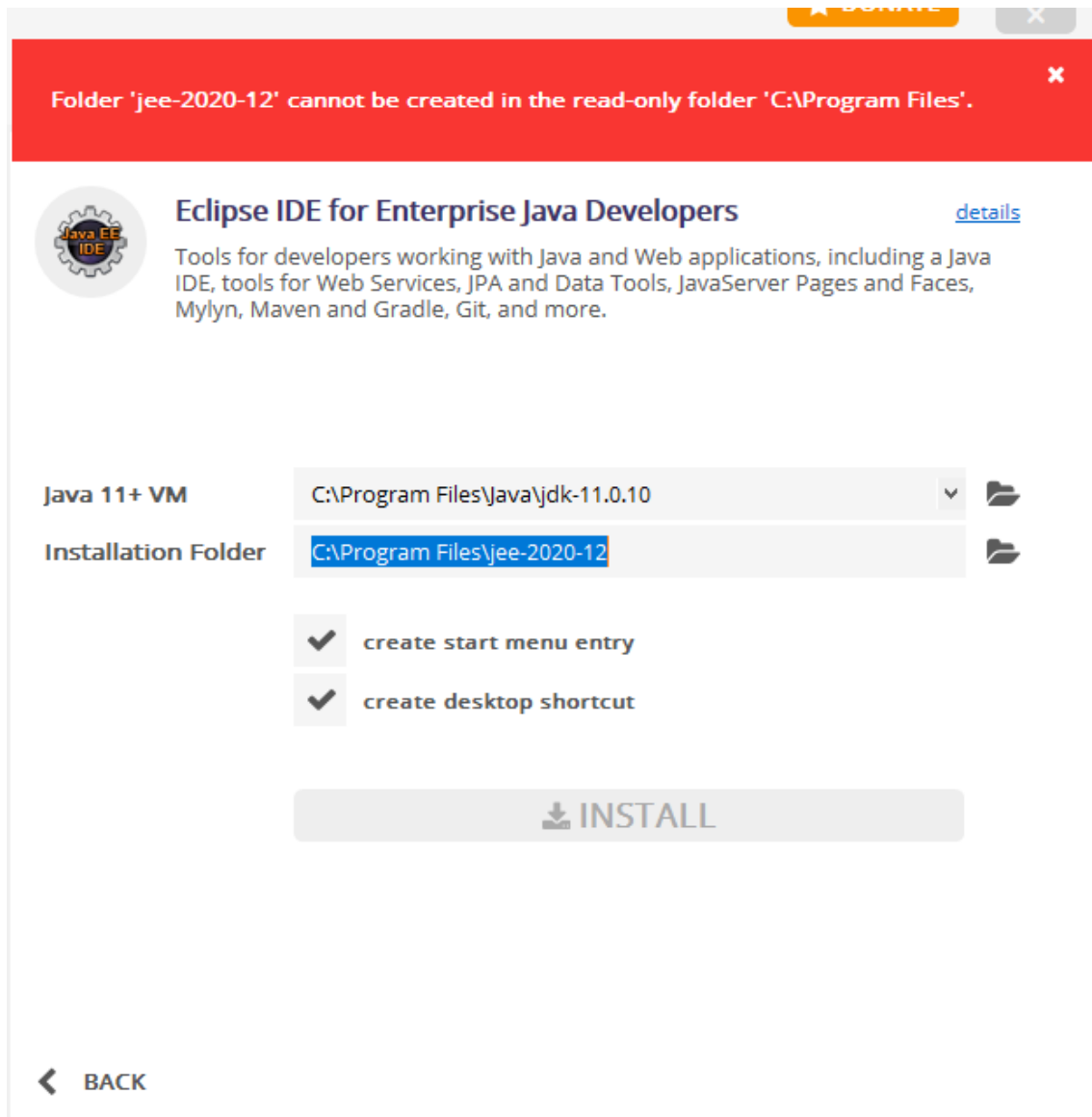
### a. Instalación de Eclipse IDE:

Podemos descargar Eclipse de su página web e instalarlo.

En la pestaña principal de la instalación, tomamos la opción de “Eclipse IDE for Enterprise Java Developers”.



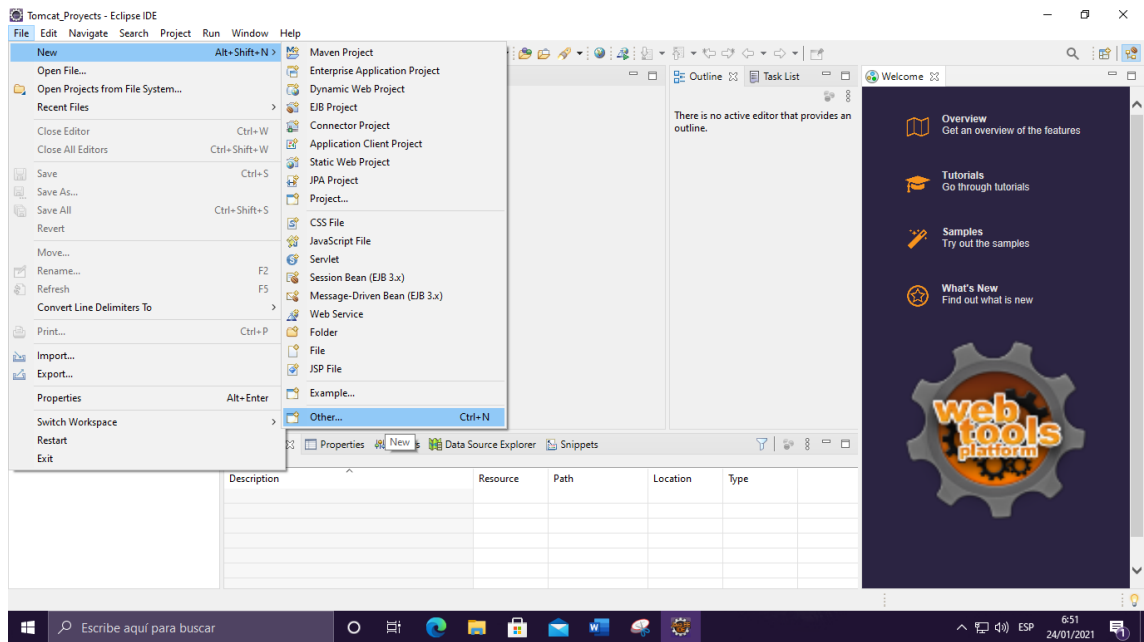
Luego de ello, preguntará por la ruta donde se encuentra el JDK y la ubicación de la carpeta de instalación.



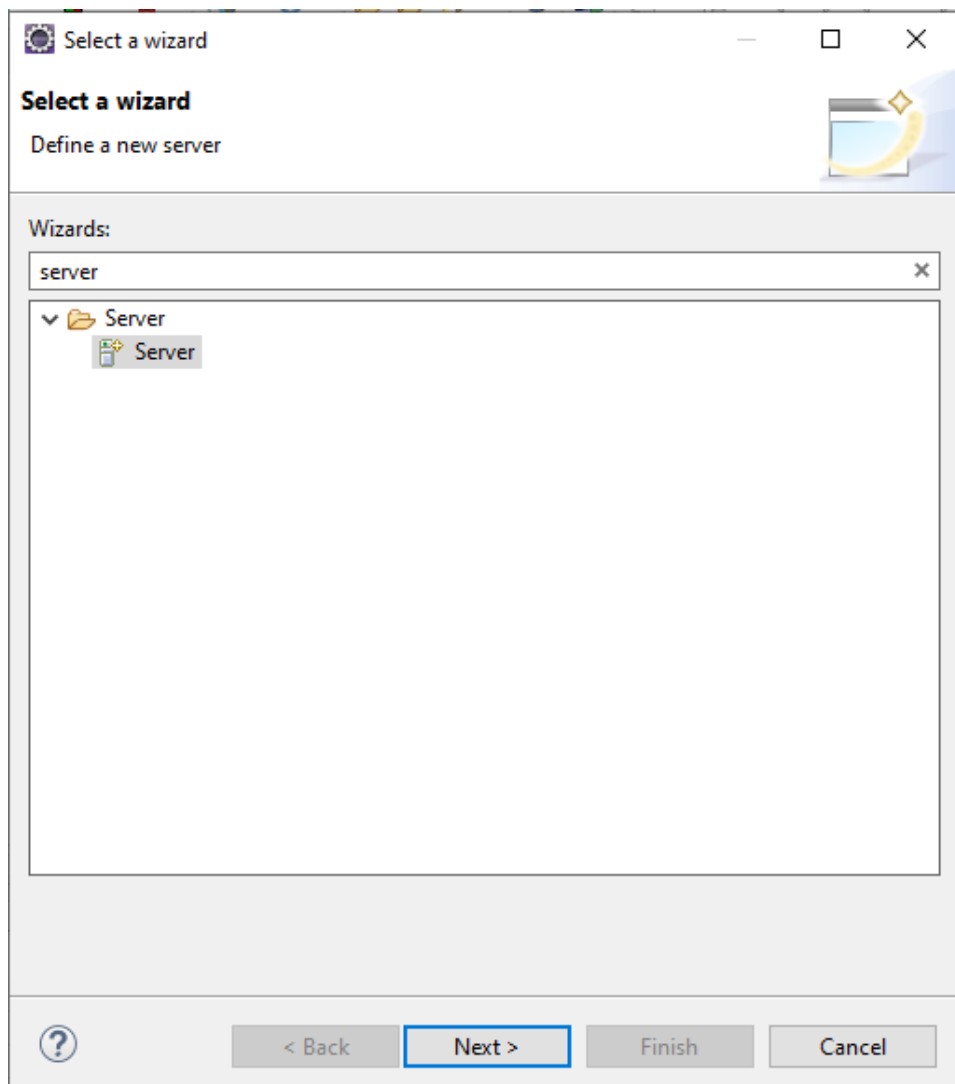
Paso siguiente, se procede a instalar el IDE y luego se configurará una carpeta para el Workspace, la cual puede ubicarse en cualquier lugar que uno elija.

#### b. Configuración de Eclipse junto a Tomcat:

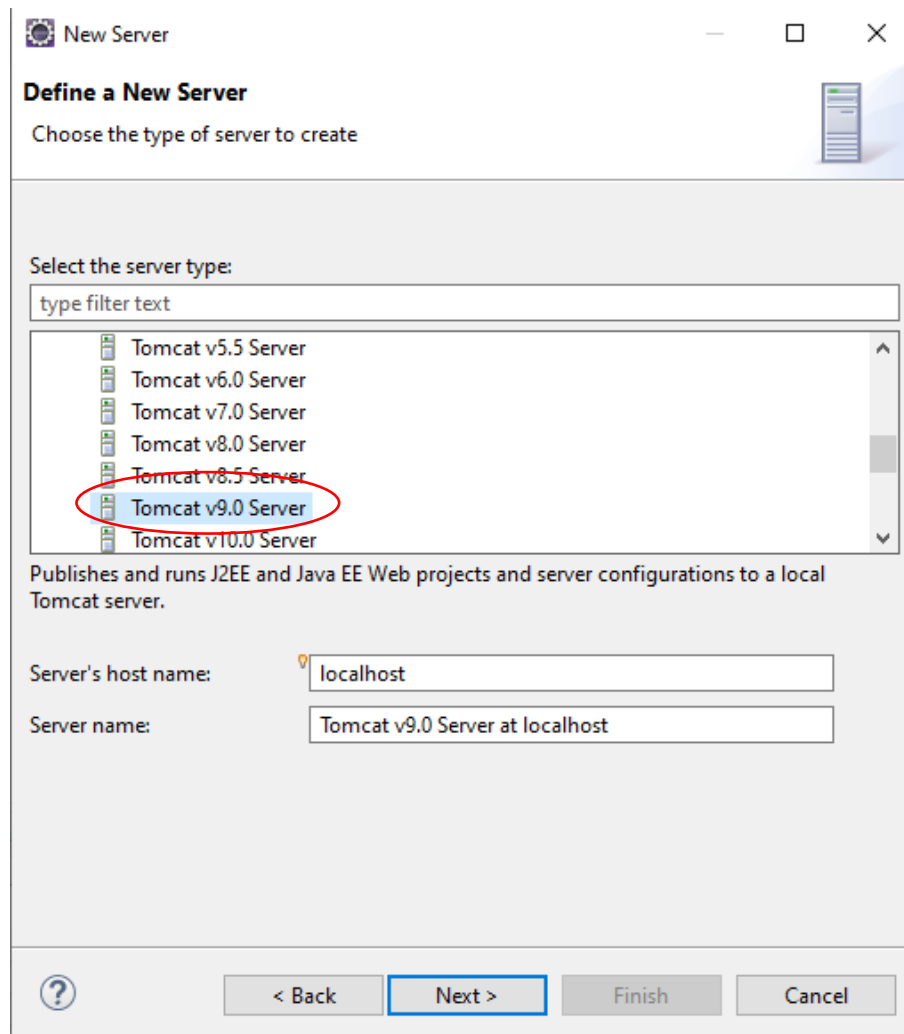
El primer paso es ir a “Archivo” y abrir uno nuevo. Luego de ello, se abrirá una lista de opciones al lado, la cual pulsaremos la última opción de “Others”.



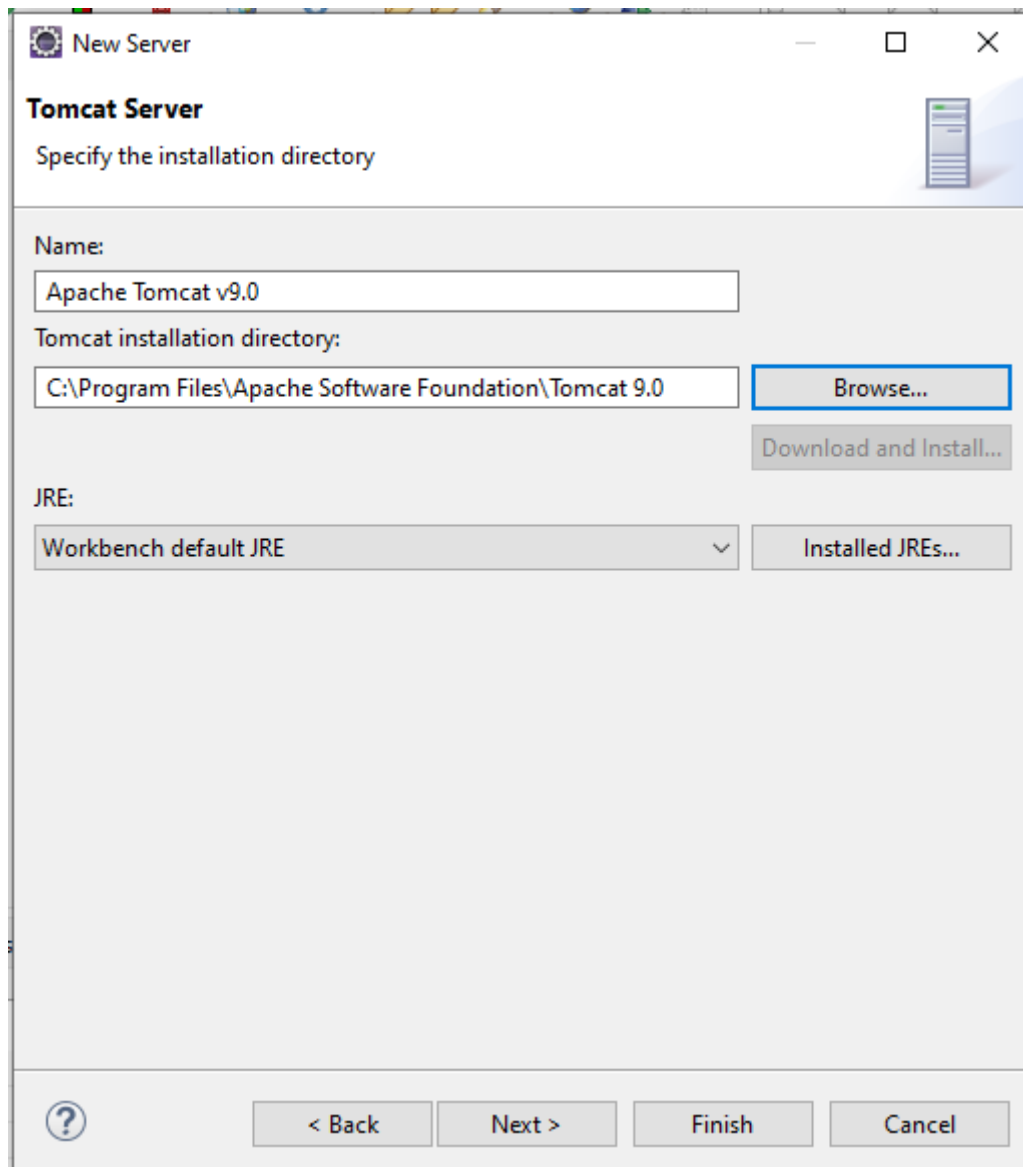
Se abrirá una ventana emergente, la cuál nos dejará buscar por nombre. Ahí mismo, buscamos el nombre “Server” y le damos a “Next”.



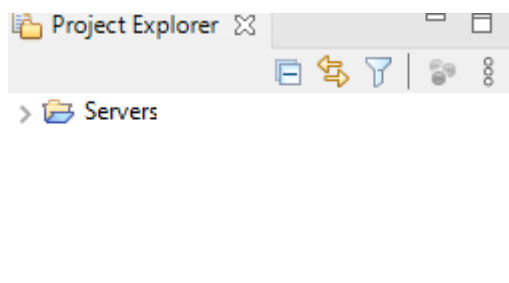
Paso seguido, nos mostrará las opciones de servidores que hay disponible. En este caso, escogeremos la carpeta de Apache y buscaremos la opción de Tomcat Server 9.0, que es la versión instalada y la cuál elegiré por encima de la versión 7.0 del XAMPP (porque es la más actual). Luego de ello, le damos click a “Next”.



La siguiente ventana preguntará por la ubicación del Tomcat. En este caso estoy usando la carpeta raíz donde se instaló el Tomcat 9.0. Le damos “Next” y por último preguntará por los proyectos disponibles y que se pueden usar en el servidor, al no haber ninguno, solo le damos a continuar y así tendremos el servidor ya configurado.

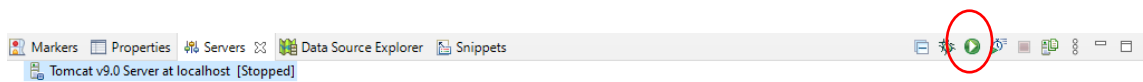


Si todo ha sido correcto y sin ninguna notificación de error, nos aparecerá una carpeta llamada "Server".

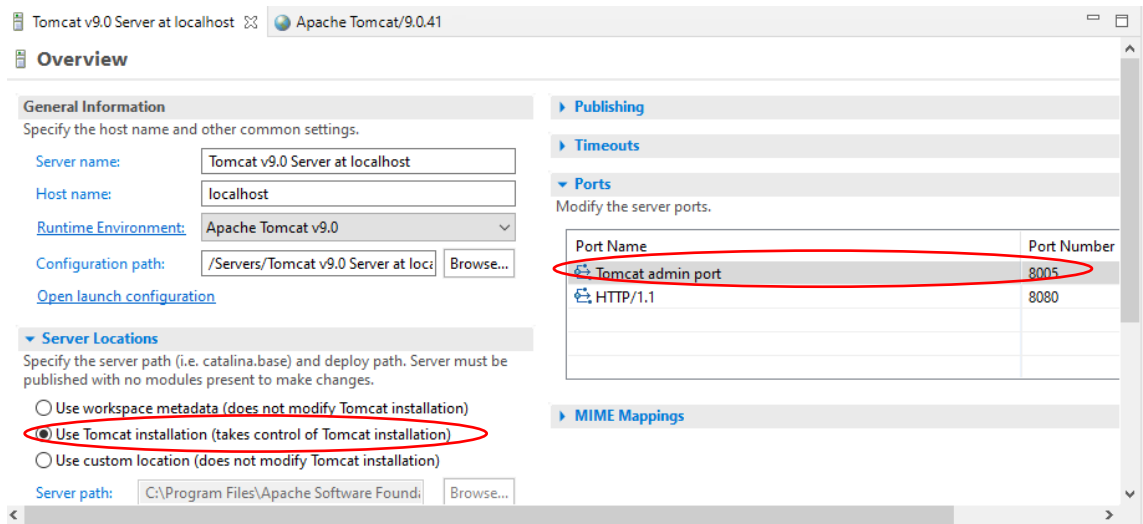


c. Comprobación del funcionamiento de Tomcat dentro de Eclipse:

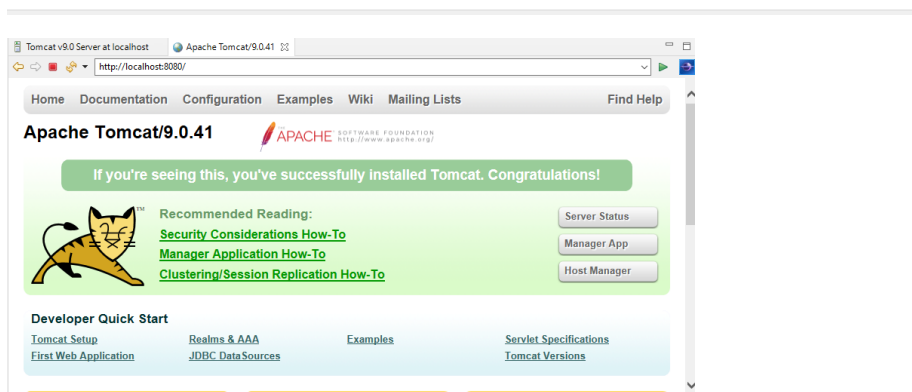
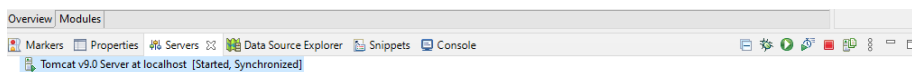
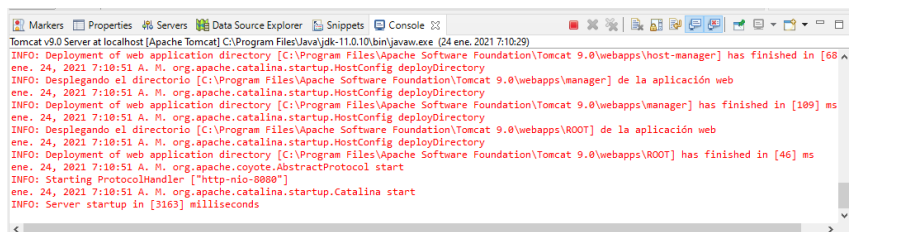
Primero, debemos irnos a la vista de Servidor y desde ahí podremos encender el servidor Tomcat creado.



En la ventana, hacemos doble click en el único renglón y nos abrirá un archivo de configuración al cual tendremos que cambiarle dos puntos: habilitamos la opción de usar la instalación del Tomcat y, además, al “Tomcat admin port” le agregaremos un puerto por defecto el cual es el 8005.



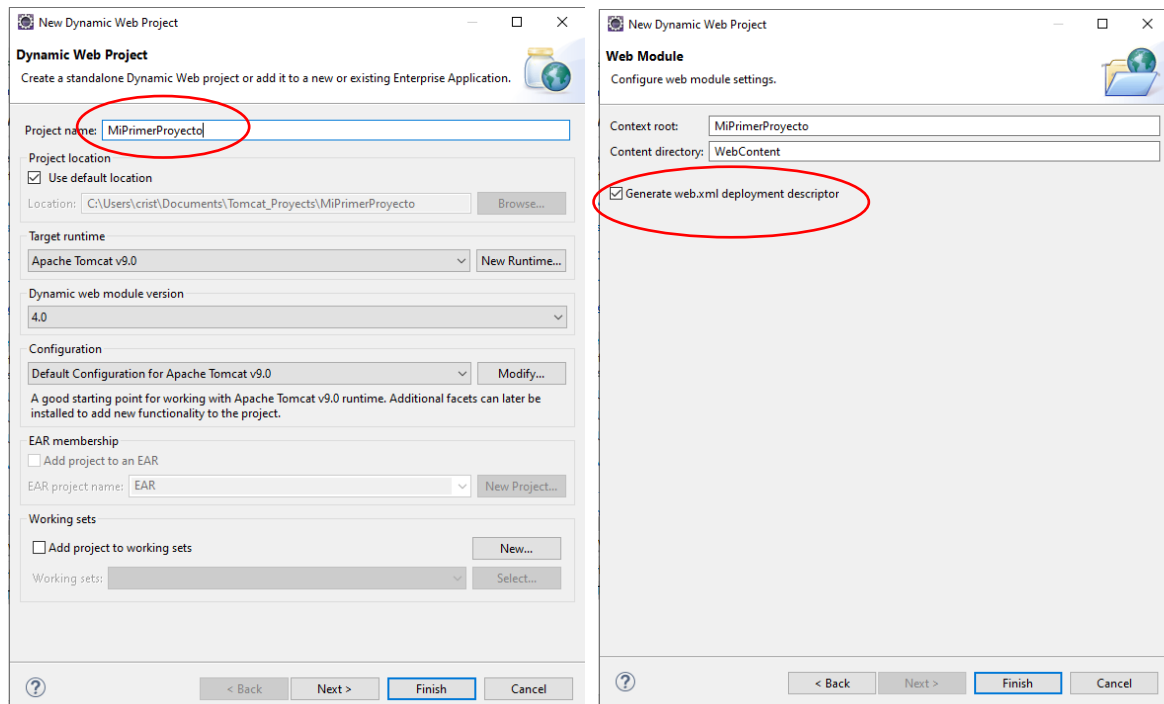
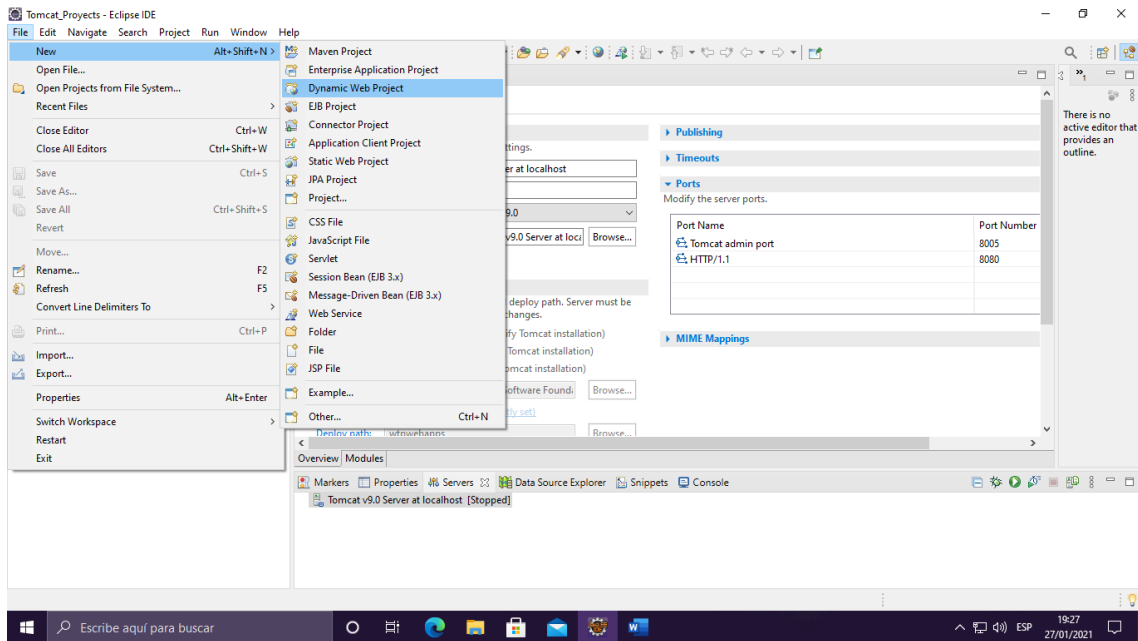
Con esto configurado, podremos arrancar el Servidor Tomcat y verificar su estado con el debugger web incorporado en el propio Eclipse:



### 3. Despliegue de Servlets:

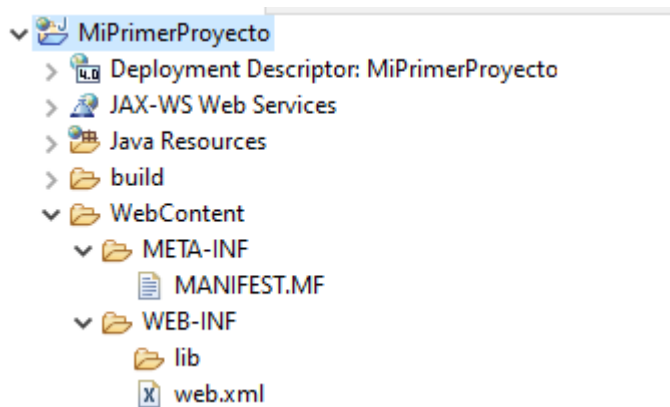
#### a. Crear el proyecto:

Crearemos el proyecto nuevo en File, luego marcaremos la opción de Dynamic Web Project. Se llamará “MiPrimerProyecto” y se le asignará la opción de crear un descriptor de despliegue de la aplicación.





b. Estructura del proyecto:



- **Deployment Descriptor:** Acceso al descriptor de despliegue guardado en el fichero **WEB-INF\web.xml**.
- Carpeta **WebContent**, dónde colocaremos todos los recursos Web, cómo ficheros HTML, JSPs, imágenes, etc. Los ficheros no colocados en este directorio, o en algún subdirectorio del mismo, no estará disponibles cuando la aplicación se ejecute en el servidor.
- Carpeta **WebContent\META-INF**. Contiene el fichero **MANIFEST.MF**, utilizado para mapear clases de ficheros JAR existentes en otros proyectos pertenecientes al mismo Enterprise Application Project.
- Carpeta **Java Resources**, dónde colocaremos nuestros Servlets y nuestros ficheros **.java**.

c. Añadir un nuevo archivo:

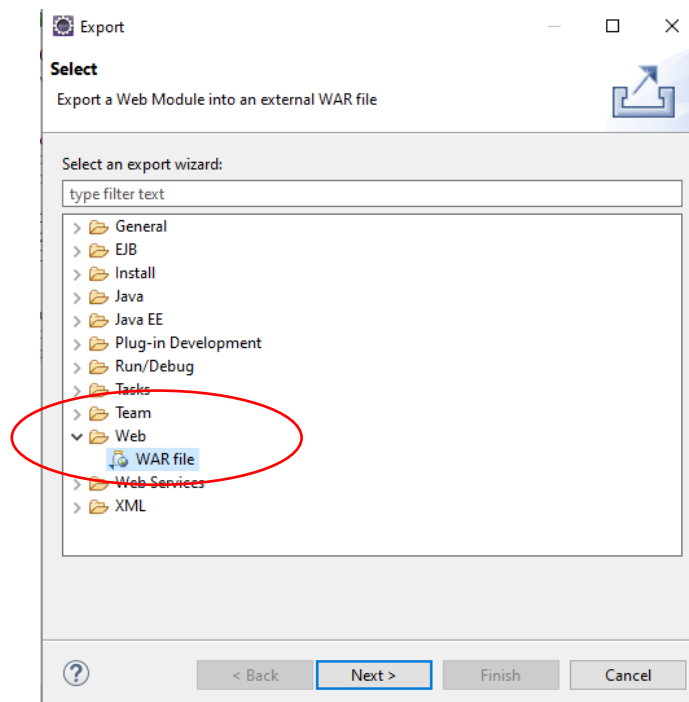
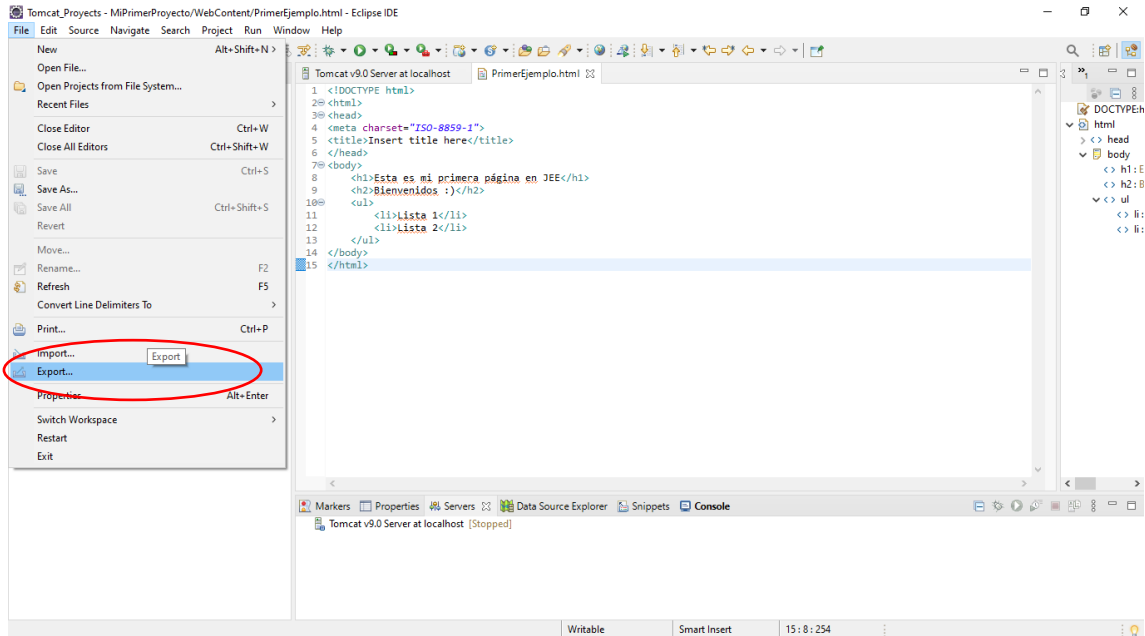
Pasaremos a añadir una hoja llamada *PrimerEjemplo.html* y la editaremos con contenido básico HTML.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <h1>Esta es mi primera página en JEE</h1>
9 <h2>Bienvenidos :)</h2>
10 <ul>
11 <li>Lista 1</li>
12 <li>Lista 2</li>
13 </ul>
14 </body>
15 </html>
```

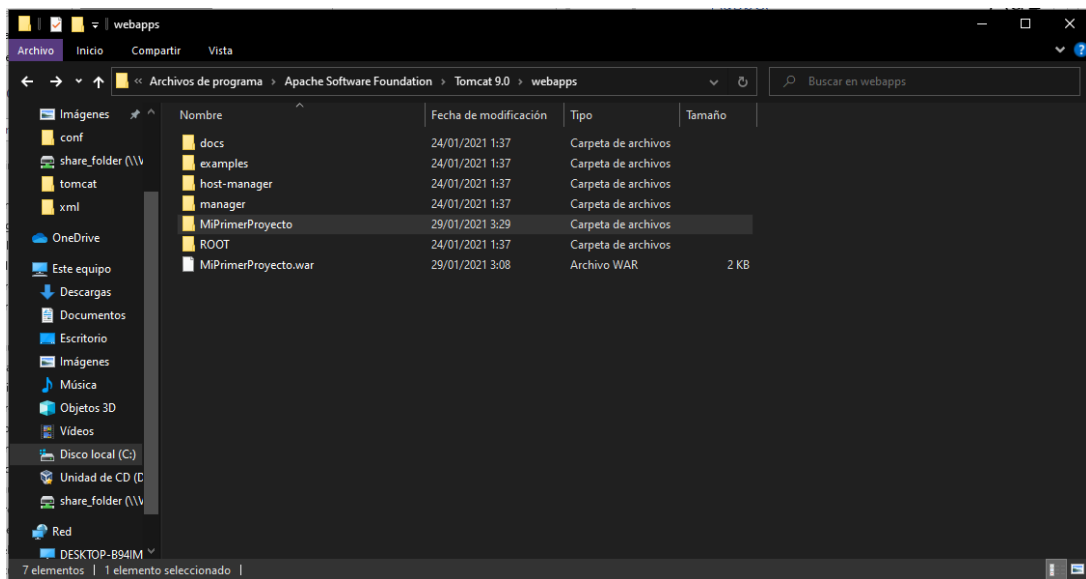
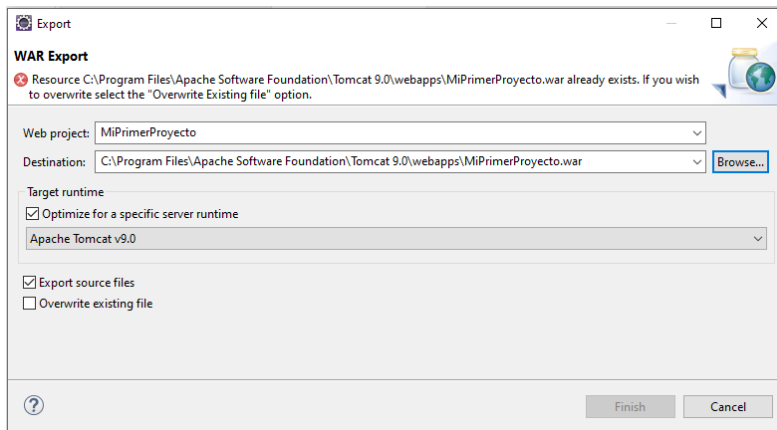
#### d. Exportar y mostrar en servidor:

Primero vamos a mostrar el archivo directamente por el servidor de Tomcat, luego lo haremos por el servidor que está en Eclipse.

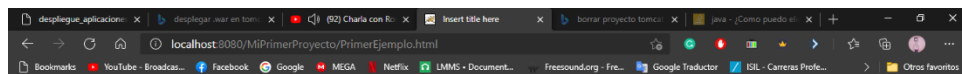
Primero, exportamos el proyecto creado cómo un archivo **WAR** (Web archives):



Luego de ello, le daremos a next y guardaremos el archivo en la carpeta del Tomcat llamada **webapps**. Automáticamente, la carpeta de Tomcat creará una carpeta en específico con el nombre de nuestro proyecto.



Le damos a finalizar y encendemos el Tomcat.



Esta es mi primera página en JEE

Bienvenidos :)

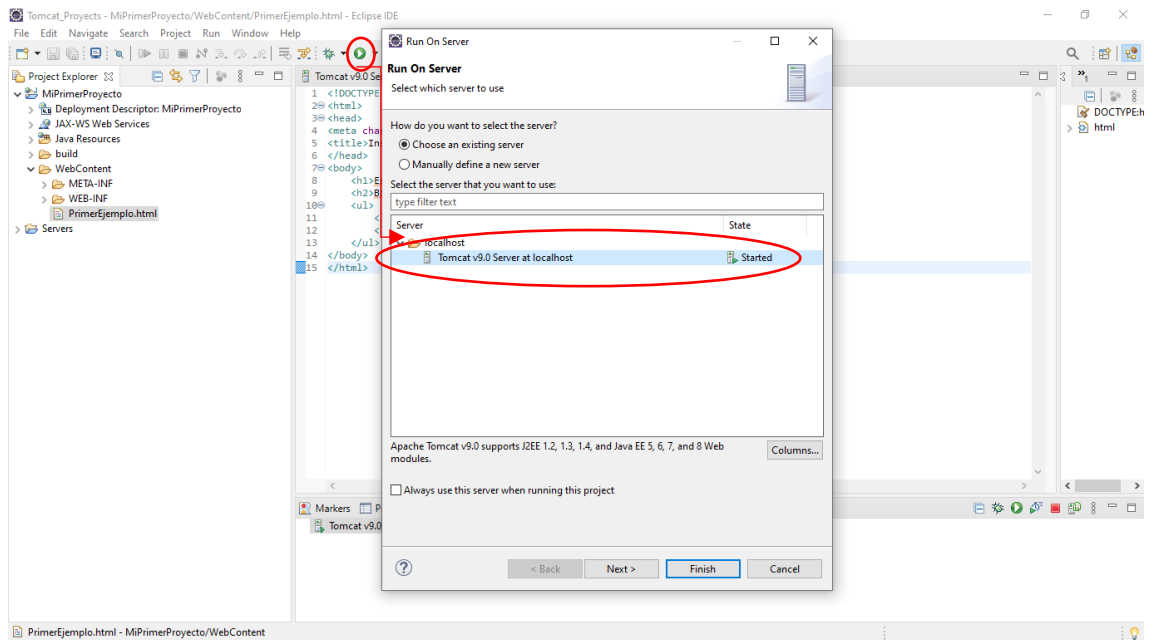
- Lista 1
- Lista 2

Probando la app desplegado, aplicamos la carpeta raíz y la página creada en el ejemplo y se puede visualizar.

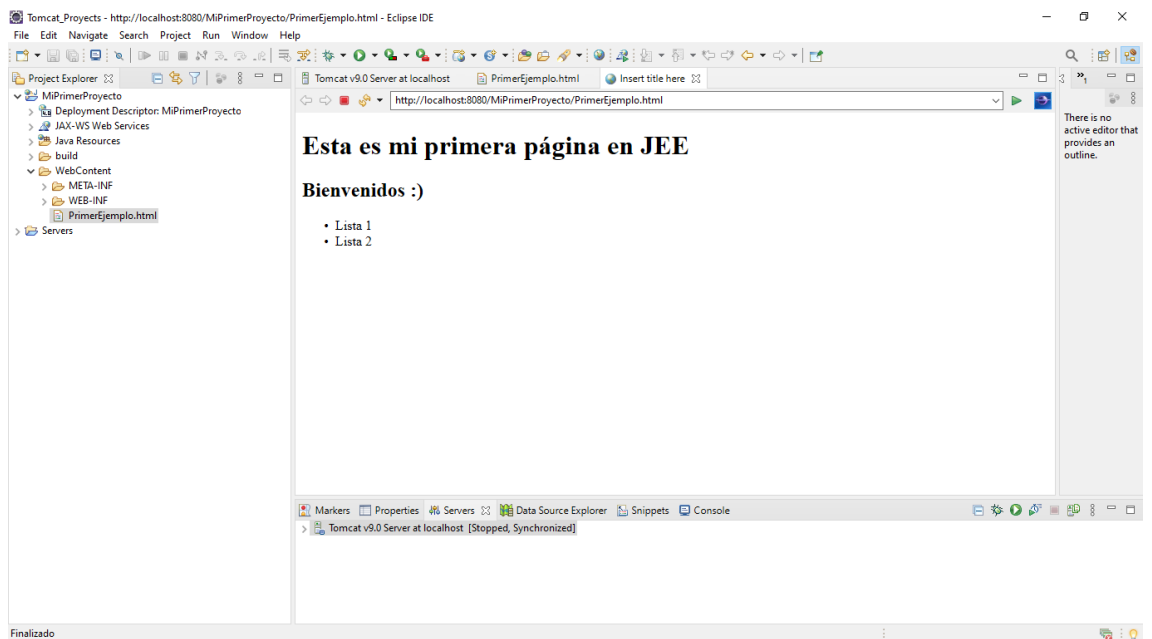
- Desplegar en Eclipse:

Primero, debemos apagar el Tomcat para darle el puerto **:8080** al server creado en Eclipse.

Posteriormente encendemos el server en Eclipse y luego procederemos a presionar el botón de iniciar y elegiremos el servidor que ya tenemos instalado:

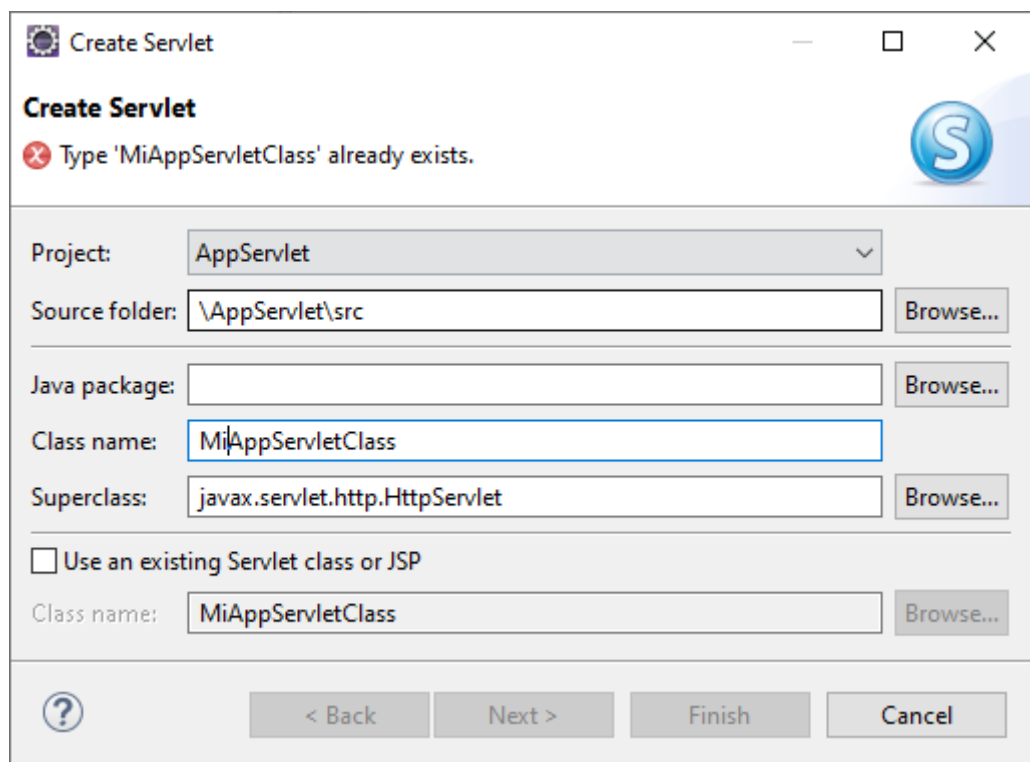
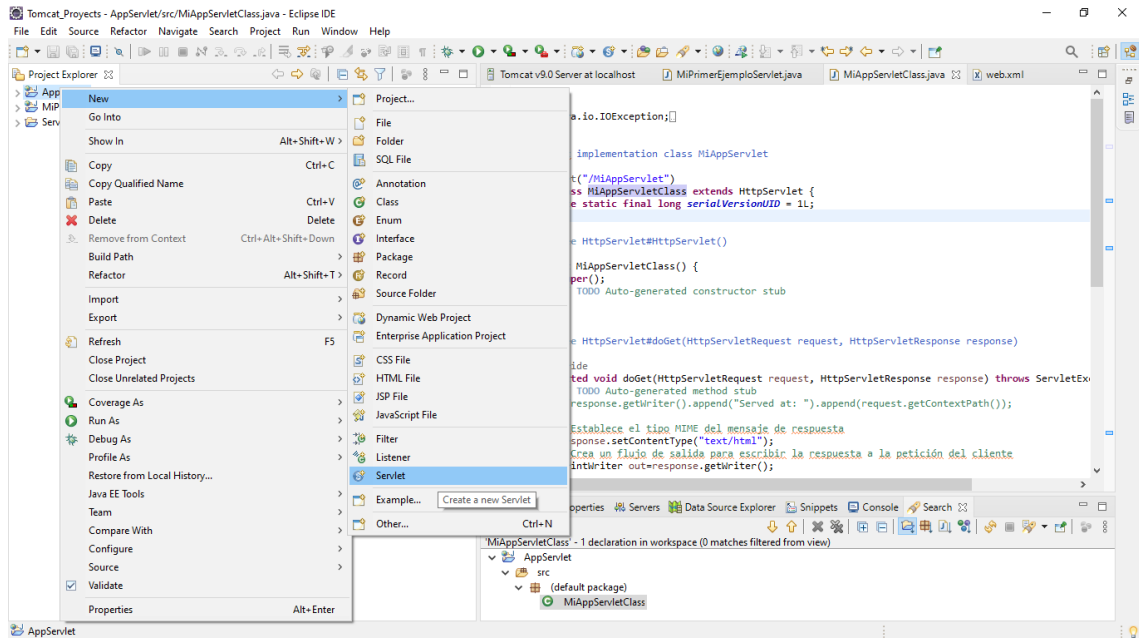


Se reiniciará el servidor y luego se levantará la página en el navegador incorporado en Eclipse:



#### e. Despliegue de un servlet:

Crearemos un nuevo proyecto llamado “AppServlet” y luego, procederemos a crear un servlet. Solo debemos darle click derecho a la carpeta **JavaResources\src** y buscar la opción de crear un servlet, colocándole un nombre, que en este caso será “PrimerEjemploServlet”. No se tendrá que agregar nada más, ya que se agregará un archivo por defecto.



Luego de esto, dentro de la función *doGet* se le agregan los siguientes parámetros y, además, se importan ciertas librerías:

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    //response.getWriter().append("Served at:
").append(request.getContextPath());

    //Establece el tipo MIME del mensaje de respuesta
    response.setContentType("text/html");
    //Crea un flujo de salida para escribir la respuesta a la petición del
cliente
    PrintWriter out=response.getWriter();

    //Escribe el mensaje de respuesta en una página html
    try {
        out.println("<html>");
        out.println("<head><title>Ejemplo Servlet</title></head>");
        out.println("<body>");
        out.println("<h1>Este es mi ejemplo de Servlet</h1>");
        //Escribe el título dentro del h1
        //Muestra información de la petición del cliente
        out.println("<p>Request URI: " + request.getRequestURI() +
"</p>");
        out.println("<p>Protocolo: " + request.getProtocol() + "</p>");
        out.println("<p>Dirección remota: " + request.getRemoteAddr() +
"</p>");
        //Genera un número aleatorio para cada petición
        out.println("<p>Número Aleatorio: <strong>" + Math.random()
+"</strong></p>");
        out.println("</body></html>");
    } finally {
        out.close(); //Cierra el flujo de salida
    }

}
```

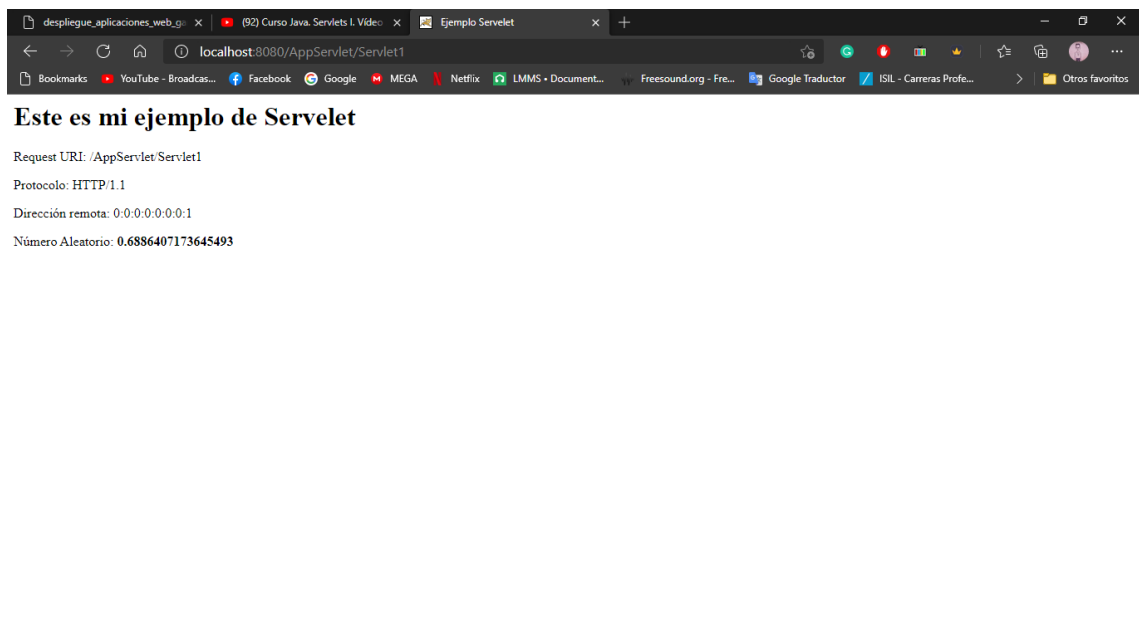
Y en el fichero **web.xml** se tienen que agregar las directivas para poder lanzar la aplicación desde las clases del servlet. El fichero se encuentra en la carpeta **WebContent/WEB-INF/lib**:

```
<!-- Indicamos que la clase MiAppServletClass se corresponde
con el nombre de servlet MiAppServlet -->

<servlet>
    <servlet-name>MiAppServlet</servlet-name>
    <servlet-class>MiAppServletClass</servlet-class>
</servlet>
<!-- Indicamos que el nombre de servlet MiAppServlet corresponde a la url
/Servlet1 -->
<servlet-mapping>
    <servlet-name>MiAppServlet</servlet-name>
    <url-pattern>/Servlet1</url-pattern>
</servlet-mapping>
```

Recordemos que para cada servlet nuevo tendremos que agregar todas etiquetas con su nombre correspondiente y su contenido referido a la clase creada.

Y luego de haber exportado el proyecto a la carpeta contenedora **webapps**, procederemos a desplegar el nuevo fichero con la siguiente dirección **“//localhost:8080/AppServlet/Servlet1”**. Estos son los resultados:



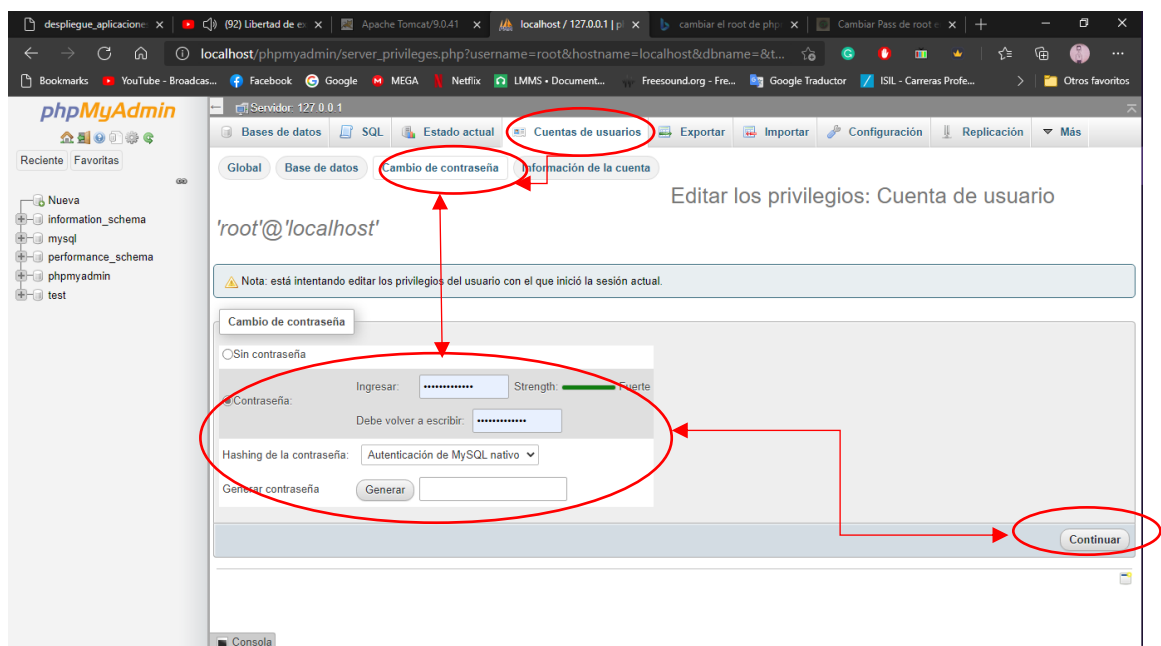
## 4. Servlet con acceso a base de datos con JDBC

### a. Instalación y habilitación del MySQL y phpMyAdmin:

Ya que previamente hemos instalado el XAMPP, ya tenemos habilitada el entorno para trabajar con el MySQL, al mismo tiempo que el phpMyAdmin. Así que procederemos a cambiar las configuraciones del root que ingresa al entorno, porque tendremos que insertar esos datos de login para la conexión con Java.

Password: cr1ss4lley12#

Primero, cambiaremos la contraseña en el phpMyAdmin, dónde en la ventana principal se encuentra en las ventanas superiores la opción de cambiar privilegios. Entramos en esa pestaña y luego aparecerá otra pestaña llamada “Cambiar Contraseña”. Por último, podemos ingresar la contraseña que queramos.

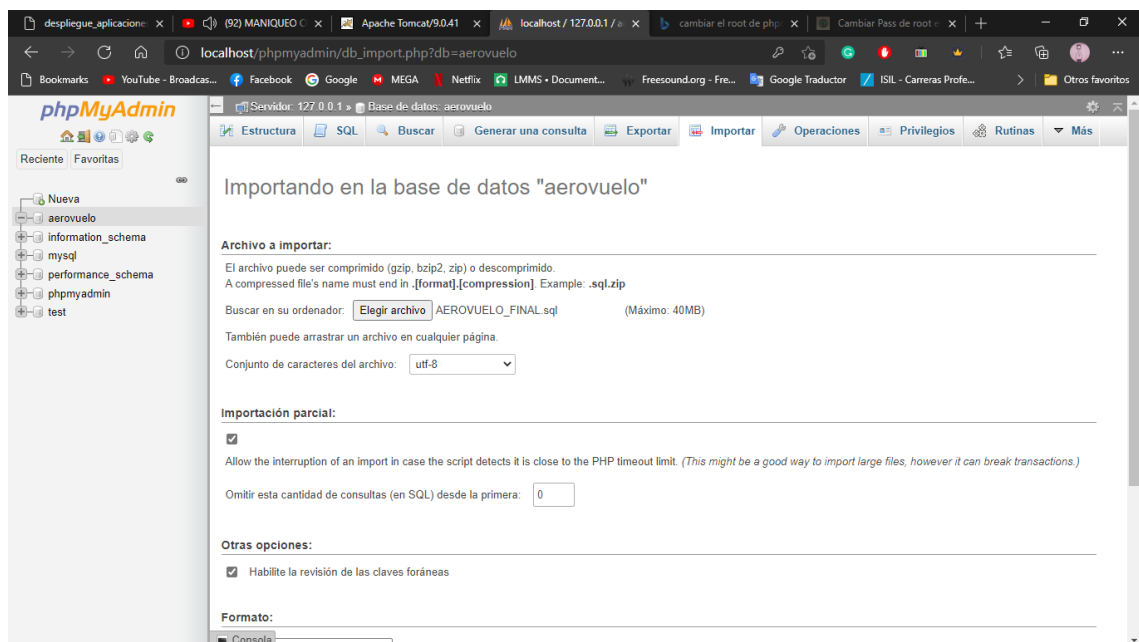


Luego, abriremos dentro de la carpeta de XAMPP ubicada en **C:\** la carpeta de phpMyAdmin, para ingresar al archivo **config.inc.php**. Se procede a cambiar en la primera parte de tipo de autenticación e información la variable de contraseña con la misma que pusimos en el IDE de phpMyAdmin.

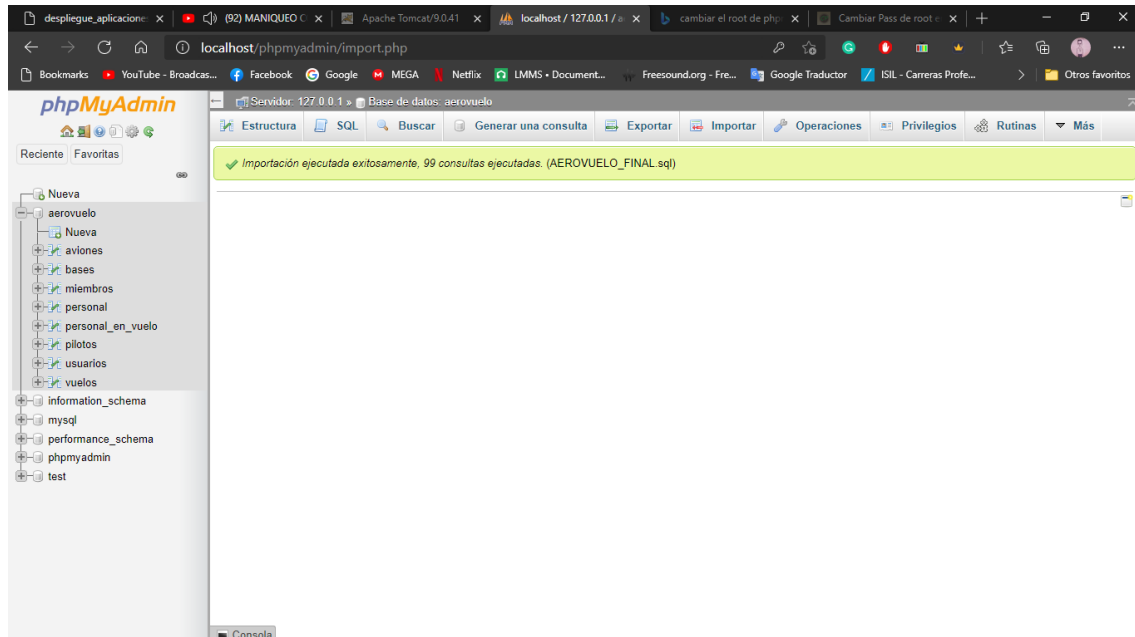


```
*C:\xampp\phpMyAdmin\config.inc.php - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
config.inc.php
10  */
11  $i = 0;
12
13  /*
14  * First server
15  */
16  $i++;
17
18  /* Authentication type and info */
19  $cfg['Servers'][$i]['auth_type'] = 'config';
20  $cfg['Servers'][$i]['user'] = 'root';
21  $cfg['Servers'][$i]['password'] = 'cr1ss41ley12#';
22  $cfg['Servers'][$i]['extension'] = 'mysqli';
23  $cfg['Servers'][$i]['AllowNoPassword'] = true;
24  $cfg['Lang'] = '';
25
26  /* Bind to the localhost ipv4 address and tcp */
27  $cfg['Servers'][$i]['host'] = '127.0.0.1';
28  $cfg['Servers'][$i]['connect_type'] = 'tcp';
29
30  /* User for advanced features */
31  $cfg['Servers'][$i]['controluser'] = 'pma';
32  $cfg['Servers'][$i]['controlpass'] = '';
33
34  /* Advanced phpMyAdmin features */
35  $cfg['Servers'][$i]['pmadb'] = 'phpmyadmin';
36  $cfg['Servers'][$i]['bookmarktable'] = 'pma_bookmark';
37  $cfg['Servers'][$i]['relation'] = 'pma_relation';
38  $cfg['Servers'][$i]['table_info'] = 'pma_table_info';
39  $cfg['Servers'][$i]['table_coords'] = 'pma_table_coords';
40  $cfg['Servers'][$i]['pdf_pages'] = 'pma_pdf_pages';
41  $cfg['Servers'][$i]['column_info'] = 'pma_column_info';
42  $cfg['Servers'][$i]['history'] = 'pma_history';
PHP Hypertext Preprocessor file length: 2,064 lines: 62 Ln: 21 Col: 49 Pos: 430 Unix (LF) UTF-8 INS
```

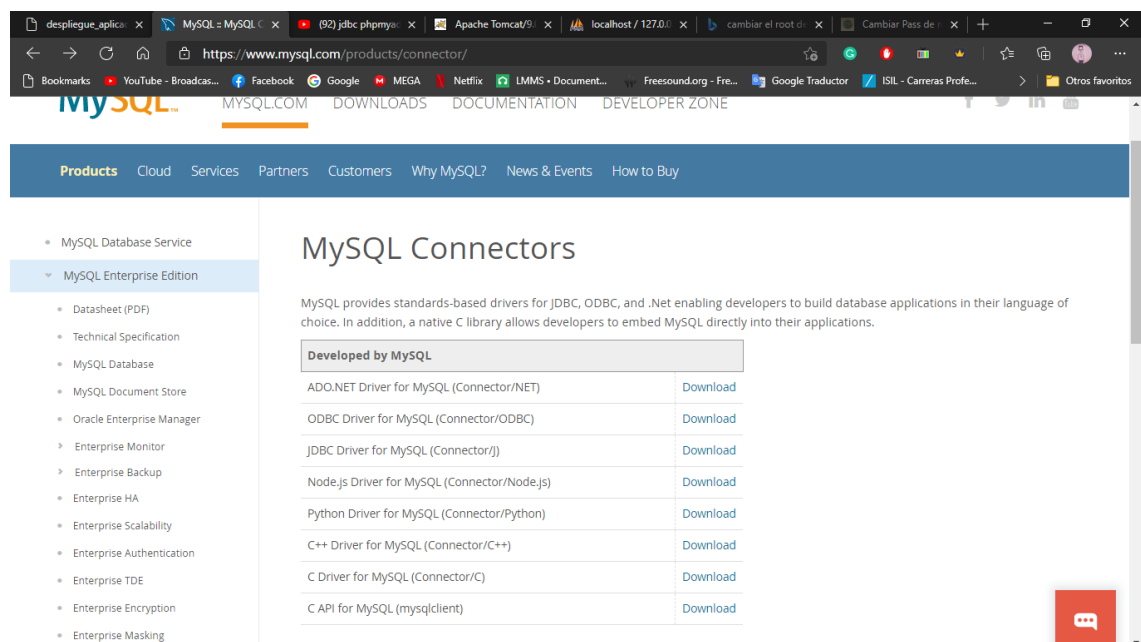
Posteriormente, creamos un schema “aerovuelo” e importaremos la base de datos que creamos en el proyecto AEROVUELO.



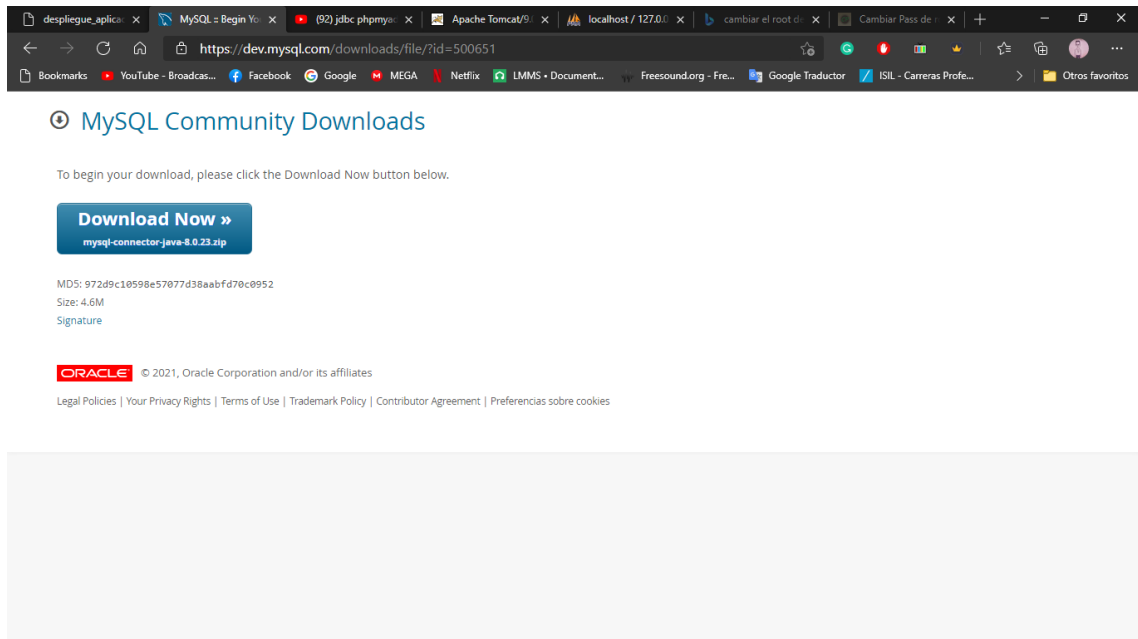
Y estos son los resultados:



Luego de esto, debemos descargar el JDBC desde la página , y luego tenemos que vincularlo con el phpMyAdmin.



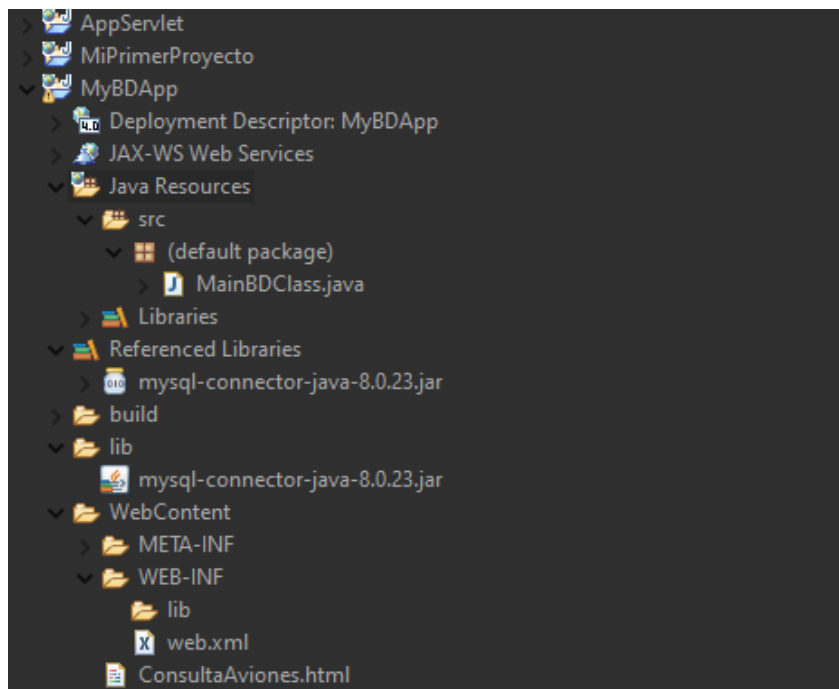
Con la carpeta ya descomprimida, tenemos que referenciarla en el nuevo proyecto que vamos a crear. Para ello primero debemos crear un nuevo proyecto para luego vincularlo con el conector JDBC.



Posteriormente, debemos agregar el archivo **.jar** descomprimido en la carpeta **C:\Program Files\Apache Software Foundation\Tomcat 9.0\lib**, así conseguiremos que la librería se habilite para hacer la consulta correspondiente a la base de datos.

#### b. Implementación y consulta a la base de datos

Ya en Eclipse, creamos un Servlet y en la carpeta **WEB-CONTENT** el archivo **ConsultaAviones.html**.



Este es el código para el .html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Consulta Aviones</title>
<style>
    ul, li{
        list-style:none;
    }
    p{
        font-weight:bold;
        font-size:1.2em;
    }
</style>
</head>
<body>
<h2>Aquí podrás consultar por los aviones que existen</h2>
<form method="GET" action="//localhost:8080/MyBDApp/consulta">
<h4>Elige un avión:</h4>
<ul>
    <li><p><input type="checkbox" name="avion" value="BOEING 747"/> Boeing
747</p></li>
    <li><p><input type="checkbox" name="avion" value="BOEING 777"/> Boeing
777</p></li>
    <li><p><input type="checkbox" name="avion" value="BOEING 737"/> Boeing
737</p></li>
    <li><p><input type="checkbox" name="avion" value="BOEING 787"/> Boeing
787</p></li>
    <li><p><input type="checkbox" name="avion" value="BOEING 757"/> Boeing
757</p></li>
    <li><p><input type="checkbox" name="avion" value="BOEING 767"/> Boeing
767</p></li>
    <li><p><input type="checkbox" name="avion" value="BOEING 727"/> Boeing
727</p></li>
    <li><p><input type="checkbox" name="avion" value="A 320"/> A
320</p></li>
    <li><p><input type="checkbox" name="avion" value="A 380"/> A
380</p></li>
    <li><p><input type="submit" value="Buscar"/></p></li>
</ul>
</form>
</body>
</html>
```

En el action del formulario estamos redireccionando al servlet dónde se harán las consultas.

Este es el código del servlet:

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

/**
 * Servlet implementation class MainBDClass
 */

public class MainBDClass extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();

        //Se crea la variable para la conexión
        Connection conn = null;
        //Se crea variable para la consulta
        Statement stmt = null;

        try {
            //Paso 1: Cargar el driver JDBC
            Class.forName("com.mysql.cj.jdbc.Driver");

            //Paso 2: Conectarse a la base de Datos utilizando la
clase Connection
            String userName="root";
            String password="cr1ss4lley12#";
            String url="jdbc:mysql://localhost:3306/aerovuelo";
            conn = DriverManager.getConnection(url, userName,
password);

            //Paso 3: Crear sentencias SQL utilizando objetos tipo
Statement

            stmt = conn.createStatement();

            //Paso 4: Ejecutar las sentencias SQL a través de los
objetos Statement
            String sqlStr= "SELECT * FROM AVIONES WHERE A_TIPO = " +
            "'" + request.getParameter("avion") + "'";

            //Generar una página HTML como resultado de la consulta

            out.println("<html><head><title>Resultado de la
consulta</title></head><body>");
```

```

        out.println("<h3>Estos son los resultados de tu
busqueda</h3>");
        out.println("<p> Tu consulta es: " + sqlStr + "</p>");
        ResultSet rset = stmt.executeQuery(sqlStr);

        //Paso 5: Procesar el conjunto de registros resultante
        utilizando ResultSet
        int count = 0;
        int dispo = 0;
        while(rset.next()) {
            out.println("<p> ID: " +
rset.getString("idAVIONES") + "</p>");
            out.println("<p> Modelo: " +
rset.getString("A_TIPO") + "</p>");
            dispo =
Integer.parseInt(rset.getString("A_DISPONIBLE"));
            if(dispo==0) {
                out.println("<p>Avión disponible</p>");
            }
            else {
                out.println("<p>Avión no disponible</p>");
            }
            out.println("<p>Id de la base del avion: " +
rset.getString("A_idBASES") + "</p>");
            count++;
        }
        out.println("<p>----- " + count + " registros encontrados
-----</p>");
        out.println("</body></html>");
    }catch (Exception e) {
        e.printStackTrace();
    }finally {
        out.close(); //Cerramos el flujo de escritura
        try {
            //Cerramos el resto de recursos
            if(stmt != null) {stmt.close();}
            if(conn != null) {conn.close();}
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}

}

}

```

Luego de haber especificado la ruta para la base de datos, especificar los datos de usuario para el phpMyAdmin y hacer la consulta correspondiente, tenemos que cambiar el archivo **web.xml**.

```

<servlet>
    <servlet-name>ConsultaAviones</servlet-name>
    <servlet-class>MainBDClass</servlet-class>
</servlet>

```

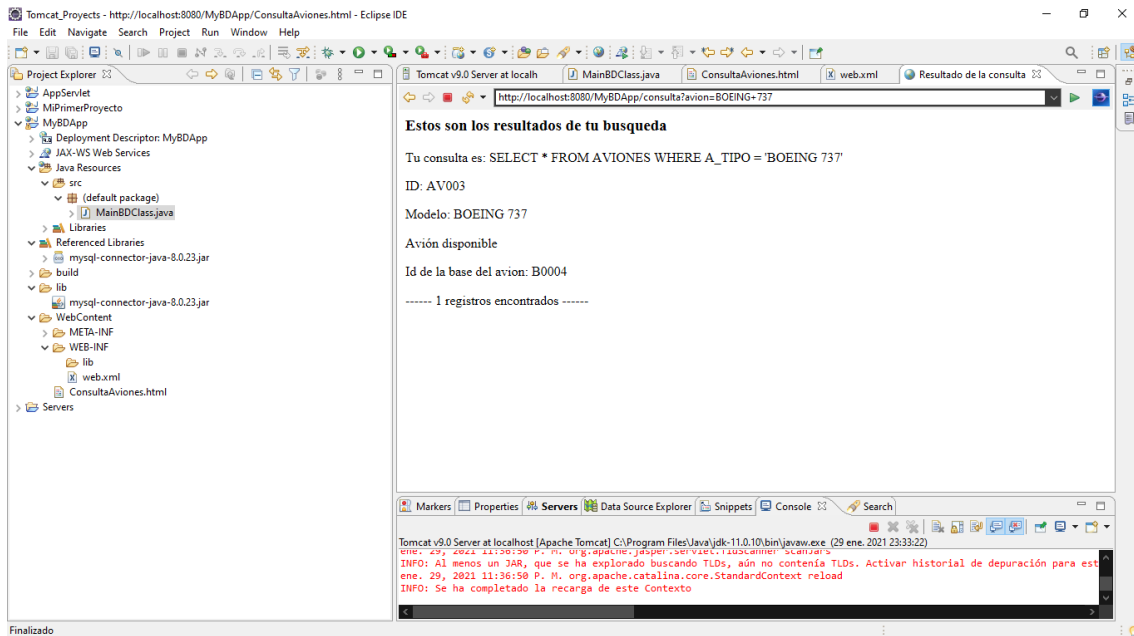
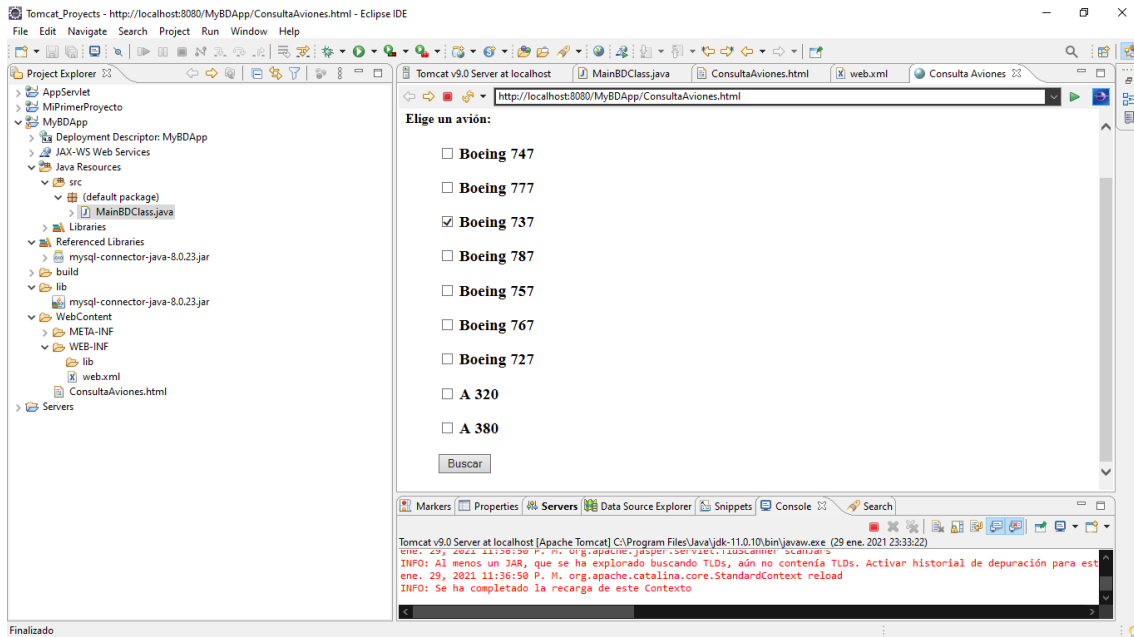
```

<servlet-mapping>
    <servlet-name>ConsultaAviones</servlet-name>
    <url-pattern>/consulta</url-pattern>
</servlet-mapping>

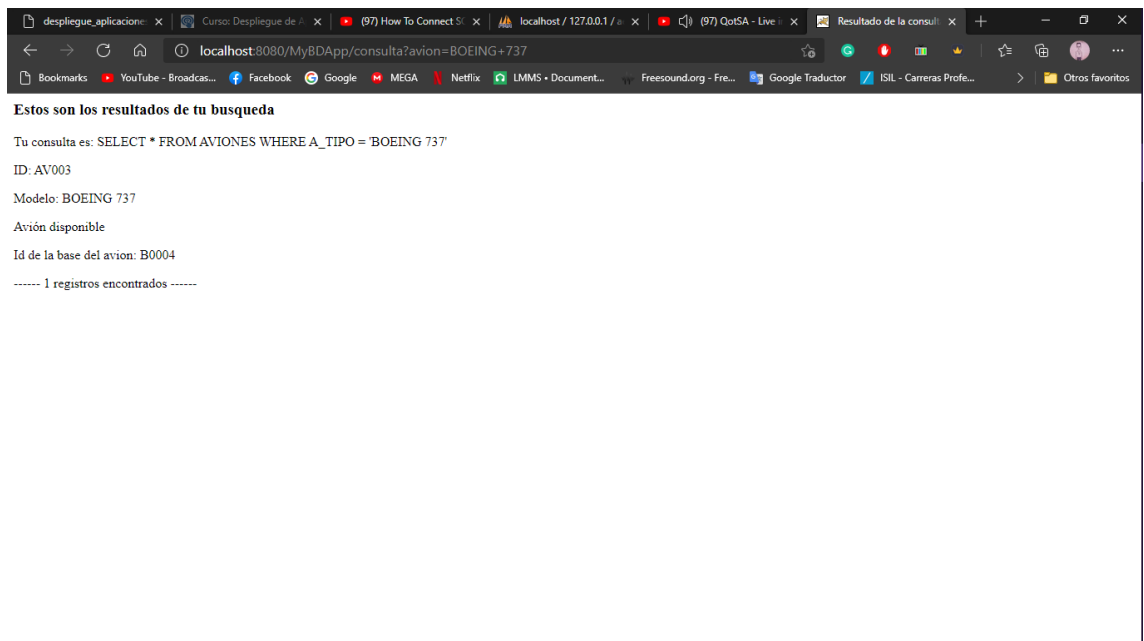
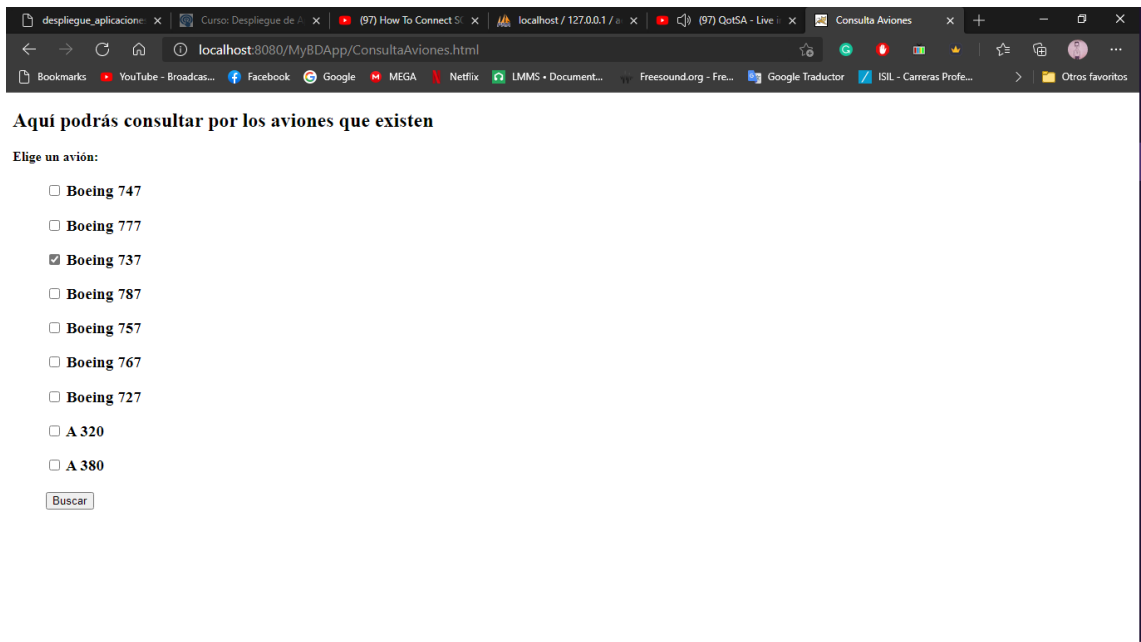
```

c. Resultados:

Y estos son los resultados:



## Resultados en la Web:



## 5. Sesiones con Servlets:

### a. Crear proyecto y establecer por cookies:

Creamos un proyecto llamado **Service**. Añadimos un Servlet llamado **"SesionesServlet.java"** y le insertamos el siguiente código:

```
import java.io.*;
import javax.servlet.http.*;
import java.util.Date;
```



```

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    //Recoge la sesión actual si existe, en otro caso crea una nueva
    HttpSession session = request.getSession();
    Integer contadorAccesos;
    synchronized(session) {
        contadorAccesos =
(Integer)session.getAttribute("contadorAccesos");
        if (contadorAccesos == null) {
            contadorAccesos = 0;
        } else {
            contadorAccesos = contadorAccesos + 1;
        }
        session.setAttribute("contadorAccesos", contadorAccesos);
    }

    //Escribe el mensaje de respuesta en una página html
    try {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<meta http-equiv='Content-Type'
content='text/html; charset=UTF-8' />");
        out.println("<title>Servlet de la prueba de
sesión</title></head><body>");
        out.println("<h2>Accesos: " + contadorAccesos + " en esta
sesión. </h2>");
        out.println("<p>(Identificador de sesión: " +
session.getId() + ">");
        out.println("<p>(Fecha de creación de la sesión: " +
new Date(session.getCreationTime()) + ">");
        out.println("<p>(Fecha último acceso " +
new Date(session.getLastAccessedTime()) + ">");
        out.println("<p>(Máximo tiempo inactivo de la sesión: " +
session.getMaxInactiveInterval() + " seconds)>");
        out.println("<p><a href='" + request.getRequestURI() +
">Refrescar</a>>");
        out.println("<p><a href='" +
response.encodeURL(request.getRequestURI())
+ ">Refrescar con reescritura de URLs</a>>");
        out.println("</body></html>");
    } finally {
        out.close(); //Cerrar el flujo de salida
    }
}

```

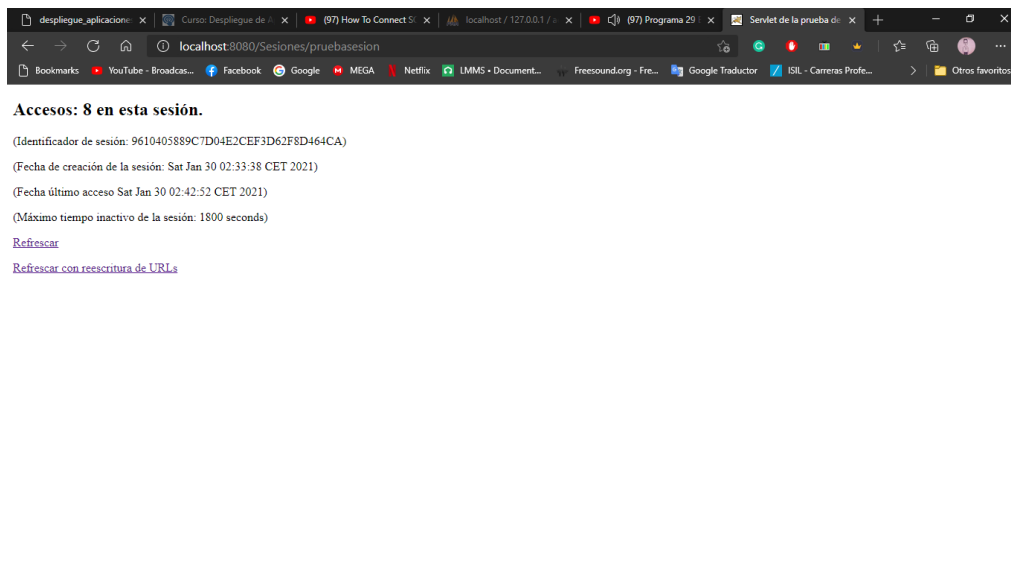
Y estos son los parámetros para el **web.xml**:

```
<servlet>
  <servlet-name>ServletPruebaSession</servlet-name>
  <servlet-class>MainClassSession</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>ServletPruebaSession</servlet-name>
  <url-pattern>/pruebasesion</url-pattern>
</servlet-mapping>
```

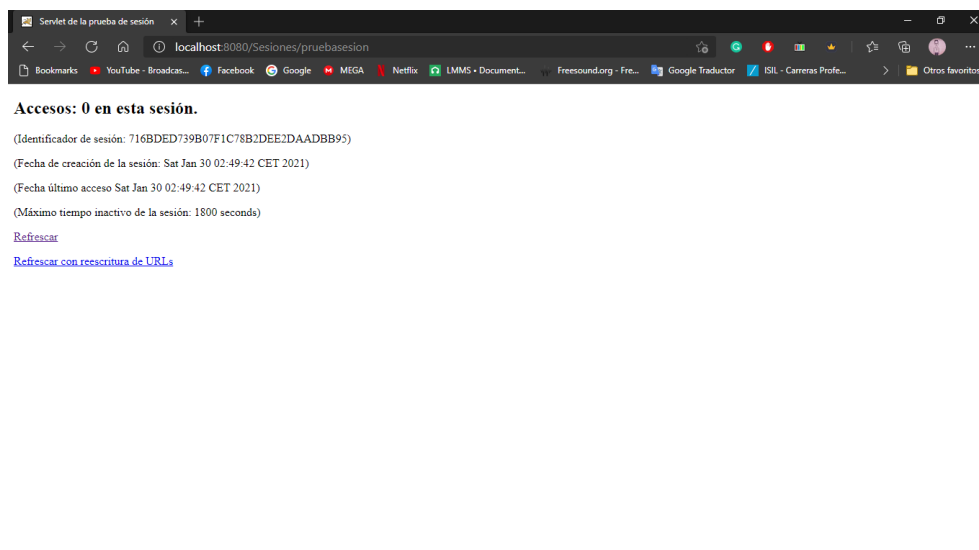
## b. Despliegue y resultados:

Luego de terminar el código, procedemos a exportar proyecto en un **.war**. Luego de esto desplegamos el resultado en el navegador:



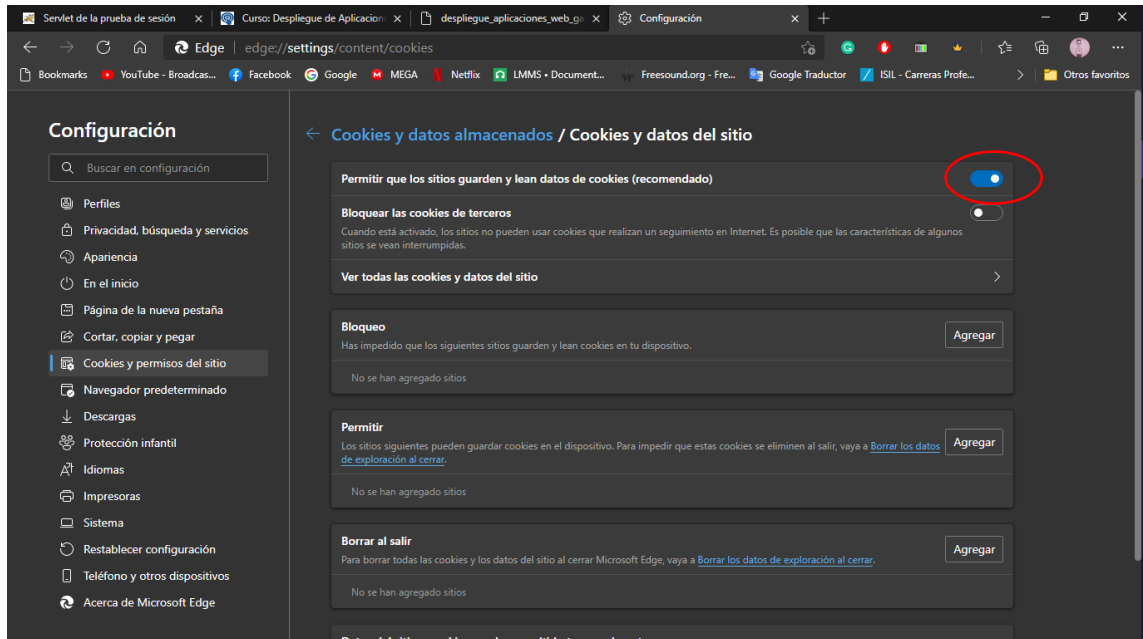
Estos resultados corresponden a actualizar la página, ya que recolecta las sesiones a través de las cookies.

Al reiniciar el navegador, obtenemos que la sesión ha expirado y vuelve a contar de nuevo desde 0. También cambia el identificador, porque es una nueva:

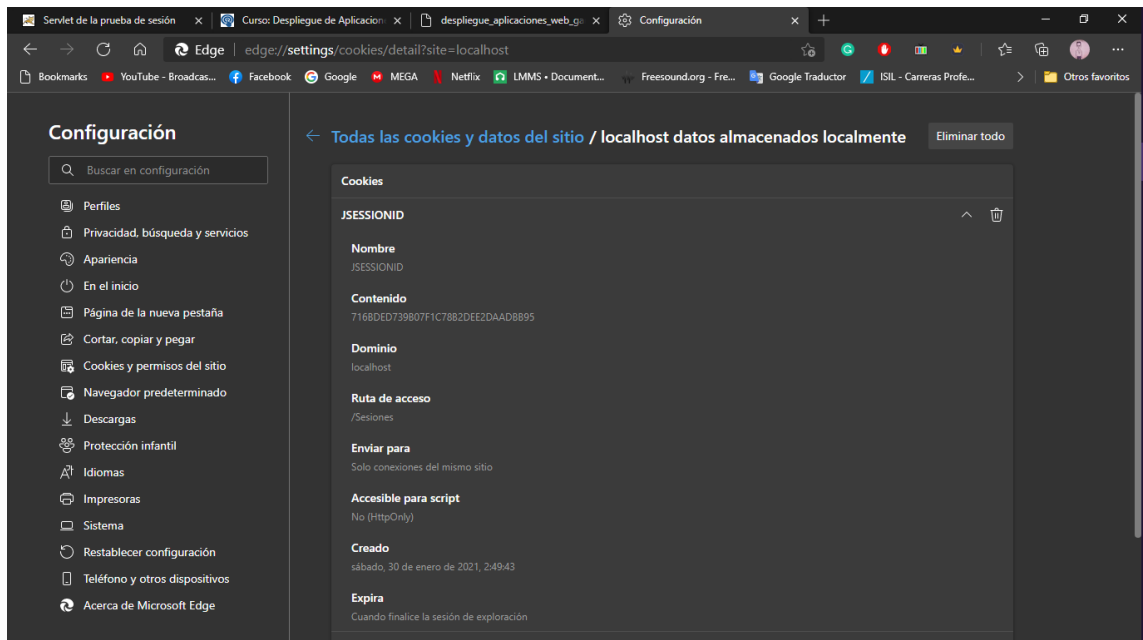


### c. Despliegue sin uso de cookies (usando el ID):

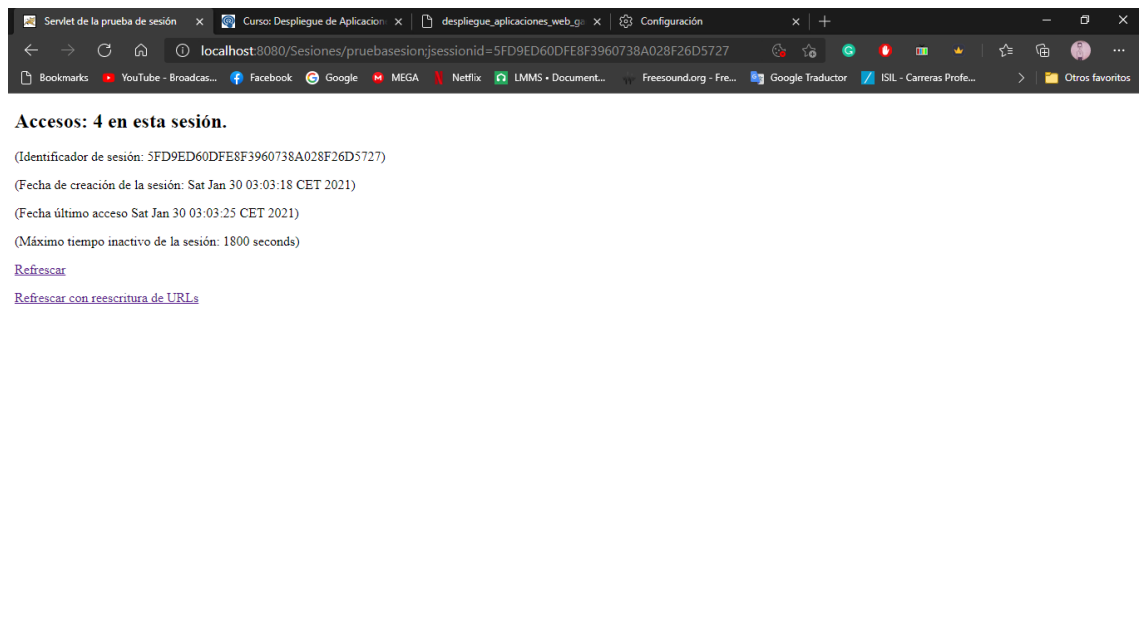
Automáticamente el API de Java taba con las cookies para empezar y mantener una sesión. Ahora intentaremos hacerlo sin ello. Tenemos que deshabilitar la opción de usar cookies en sitios de terceros para poder usar la otra funcionalidad:



Estos son los datos que almacena el navegador de nuestro proyecto:



Y ahora comprobaremos las sesiones de usuario via URL:



Bien se puede apreciar en la barra para ingresar la URL, figura el ID que usa Java, esto quiere decir que nuestro navegador no está permitiendo cookies y la única manera de registrar sesiones es a través de las sesiones de usuario.

## 6. Pool de conexiones con JNDI y Servlets:

### a. Creación de proyecto y definición de recurso JNDI:

El primer paso es crear un nuevo proyecto el cuál llamaremos **PoolConexiones**. Esta aplicación nos va a pedir un usuario y una contraseña para ingresar a la aplicación y en el caso de ser correcto el login, se creará una variable de sesión para almacenar el usuario.

Al ya tener una tabla llamada "Usuarios" en la base de datos, tenemos que ahora definir el recurso JNDI. Este se puede cambiar a través de la carpeta **C:\Program Files\Apache Software Foundation\Tomcat 9.0\conf**, ahí encontraremos el archivo llamado **context.xml**, al cuál le agregaremos la siguiente etiqueta:

```
<Resource name="jdbc/mysql_aerovueLo" auth="Container"
type="javax.sql.DataSource"
maxActive="100" maxIdle="30" maxWait="10000" removeAbandoned="true"
username="root" password="cr1ss4lley12#"
driverClassName="com.mysql.cj.jdbc.Driver"
url="jdbc:mysql://localhost:3306/aerovueLo"/>
```

Con esto habremos habilitado ya el recurso JNDI (Java Directory and Naming Interface), que lo que hace es reutilizar las sesiones creadas por el usuario, así las mantiene y no tiene que crear otras por cada página que contiene el proyecto. Se considera que se tiene un *pool* de conexiones abiertas, dónde este recurso busca una que esté libre y nos da esa misma, apuntando la que tenemos nosotros y deja de estar libre. Tras esto, se cierra la conexión,

pero a pesar de la petición de cierre, el pool deja abierta la conexión que nos reservó y la vuelve a marcar como libre para la siguiente petición.

Parámetros que se pueden definir:

- **Resource name:** Establece el nombre del recurso disponible para las aplicaciones.
- **Type:** establece el tipo del recurso disponible para las aplicaciones.
- **url:** La cadena de conexión a la base de datos.
- **Password:** La contraseña del usuario para acceder a la base de datos.
- **maxActive:** El número máximo de conexiones en el pool de conexiones.
- **maxIdle:** El número máximo de conexiones inactivas a retener en el pool de conexiones.
- **maxWait:** EL tiempo máximo a esperar para obtener una conexión, en milisegundos.

#### b. Codificación:

Ahora pasaremos al proyecto a definir el archivo **index.html**, en la carpeta WebContent, que será el punto de entrada a la aplicación.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Insert title here</title>
6 <style>
46 </head>
47 <body>
48 <fieldset>
49 <legend>Login</legend>
50 <form>
51 <label for="usuario"><p class="p1">Usuario: </p></label>
52 <input type="text" name="usuario"/>
53 <label for="password"><p class="p1">Contraseña: </p></label>
54 <input type="password" name="password"/>
55 <div class="bot">
56 <input type="reset" value="Limpiar"/>
57 <input type="submit" value="Enviar"/>
58 </div>
59 </form>
60 </fieldset>
61 </html>
```

Ahora crearemos un servlet llamado **LoginServlet.java** con el siguiente código:

```
import java.io.*;
import java.util.logging.*;
import javax.naming.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import javax.sql.*;
```

```

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    private DataSource pool; //Pool de conexiones a la base de datos
    public LoginServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    @Override
    public void init(ServletConfig config) throws ServletException {
        try {
            // Crea un contexto para poder luego buscar el recurso
            DataSource
                InitialContext ctx = new InitialContext();
            //Busca el recurso DataSource en el contexto
            pool =
            (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_aerovuelo");
            if(pool == null)
                throw new ServletException("Datasource desconocida
'mysql_aerovuelo'");
        } catch(NamingException e){
            Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null,
e);
        }
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        Connection conn=null;
        Statement stmt=null;
        try {

            out.println("<html><head><title>Login</title></head><body>");
            out.println("<h2>Login</h2>");

```

```

pool
    conn = pool.getConnection(); //Obtiene una conexión del

stmt = conn.createStatement();

//recupera los parámetros de la petición usuario y
password
String usuario = request.getParameter("usuario");
String password = request.getParameter("password");
boolean noUsuario = usuario != null &&
((usuario=usuario.trim()).length()>0);
boolean noPwd = password != null &&
((password=password.trim()).length()>0);

//Valida los parámetros de la petición request
if(!noUsuario) {
    out.println("<h3>Debes introducir tu
usuario</h3>");
} else if(!noPwd) {
    out.println("<h3>Debes introducir tu
contraseña</h3>");
} else {
    //Verifica que existe algún usuario con ese login y
password
String sqlStr;
sqlStr ="SELECT * FROM usuarios WHERE U_USERNAME =
'" + usuario.toLowerCase() +
                                "' AND U_PASSWORD = '" + password
+ "'";

    ResultSet rset = stmt.executeQuery(sqlStr);

    if(!rset.next()) { // Si está vacío el resulset
        out.println("<h3>Nombre o contraseña
incorrecta</h3>");
        out.println("<p><a href='index.html'>Vuelve a
la página principal</a></p>");
    } else {

        //Crea una nueva sesión y guarda el usuario
como variable de sesión

        //Primero, invalida la sesión si existe
HttpSession session =
request.getSession(false);

        if(session != null) {
            session.invalidate();
        }
        session = request.getSession(true);
        synchronized (session) {
            session.setAttribute("usuario",
usuario);
        }

        while(rset.next()) {
            out.println("<p> Hola, " +
rset.getString("U_NAME") + " !</p>");
        }
        out.println("<p><a href='hazalgo'>Hacer
Algo</a></p>");
    }
}

```

```

        }
        out.println("</body></html>");

    } catch (SQLException e){
        out.println("<h3>Servicio no
Disponible</h3></body></html>");

        Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null,
e);
    } finally {
        out.close();
        try {
            if (stmt != null) stmt.close();
            if (conn != null) conn.close(); //Devuelve la
conexión al pool
        } catch (SQLException ex) {

            Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
}

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

}

```

Paso seguido, creamos el Servlet **HazAlgo.java**:

```

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

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class HazAlgo
 */
@WebServlet("/HazAlgo")
public class HazAlgo extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */

```



```

public HazAlgo() {
    super();
    // TODO Auto-generated constructor stub
}

/**
 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();

    try {
        out.println("<html><head><title>Haz
algo</title></head><body>");
        out.println("<h2>Intenta hacer algo</h2>");

        //Recupera el nombre de usuario
        String usuario;
        HttpSession session = request.getSession(false);
        if (session == null) {
            out.println("<p>No has iniciado sesión</p>");
        } else {
            synchronized (session) {
                usuario = (String)
session.getAttribute("usuario");
            }
            out.println("<table>");
            out.println("<tr>");
            out.println("<td>Usuario</td><td>" + usuario +
"</td>");

            out.println("<tr>");
            out.println("</table>");

            out.println("<p><a href='logout'>Logout</p>");
        }
        out.println("</body></html>");
    } finally {
        out.close();
    }
}

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

```

Y, por último, el Servlet **Logout.html**:

```
import java.io.*;
import javax.servlet.http.*;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class LogoutServlet
 */
@WebServlet("/LogoutServlet")
public class LogoutServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public LogoutServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
    HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        try {

            out.println("<html><head><title>Logout</title></head><body>");
            out.println("<h2>Logout</h2>");
            HttpSession session = request.getSession();
            if(session == null) {
                out.println("<h3>No has iniciado sesión</h3>");
            } else {
                session.invalidate();
                out.println("<p>Adios</p>");
                out.println("<p><a
href='index.html'>Login</a></p>");
            }
            out.println("</body></html>");
        } finally {
            out.close();
        }
    }
}
```

```

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

```

Y para finalizar, agregamos la información de los Servlets en el archivo **web.xml**.

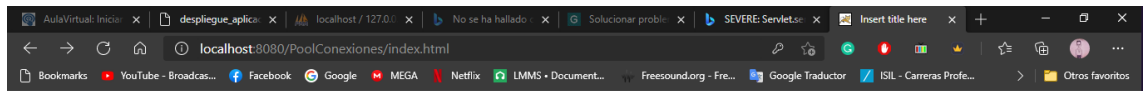
```

<servlet>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>LoginServlet</servlet-class>
</servlet>
<servlet>
    <servlet-name>HazAlgoServlet</servlet-name>
    <servlet-class>HazAlgo</servlet-class>
</servlet>
<servlet>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>LogoutServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/login</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>HazAlgoServlet</servlet-name>
    <url-pattern>/hazalgo</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/logout</url-pattern>
</servlet-mapping>
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
</welcome-file-list>

```

### c. Comprobación de resultados:

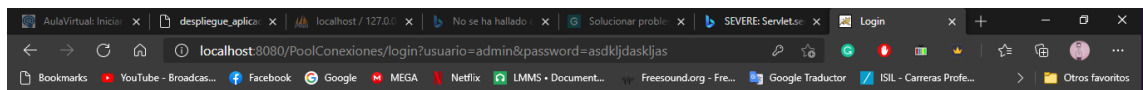
Primero comprobamos el resultado erróneo:



**Login**

Usuario:

Contraseña:

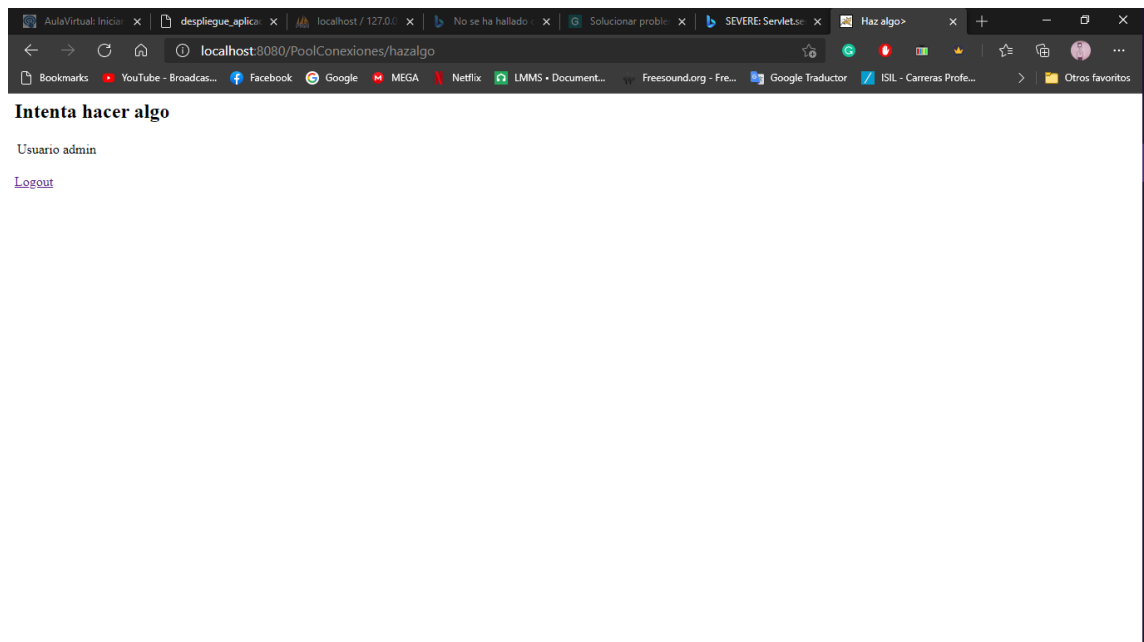
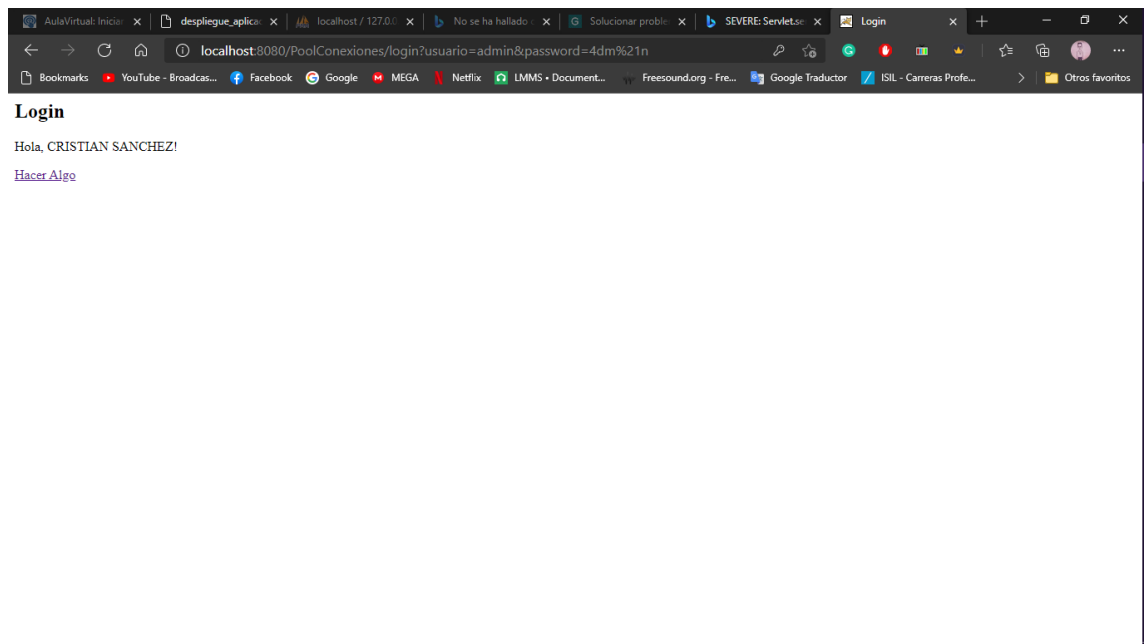


## Login

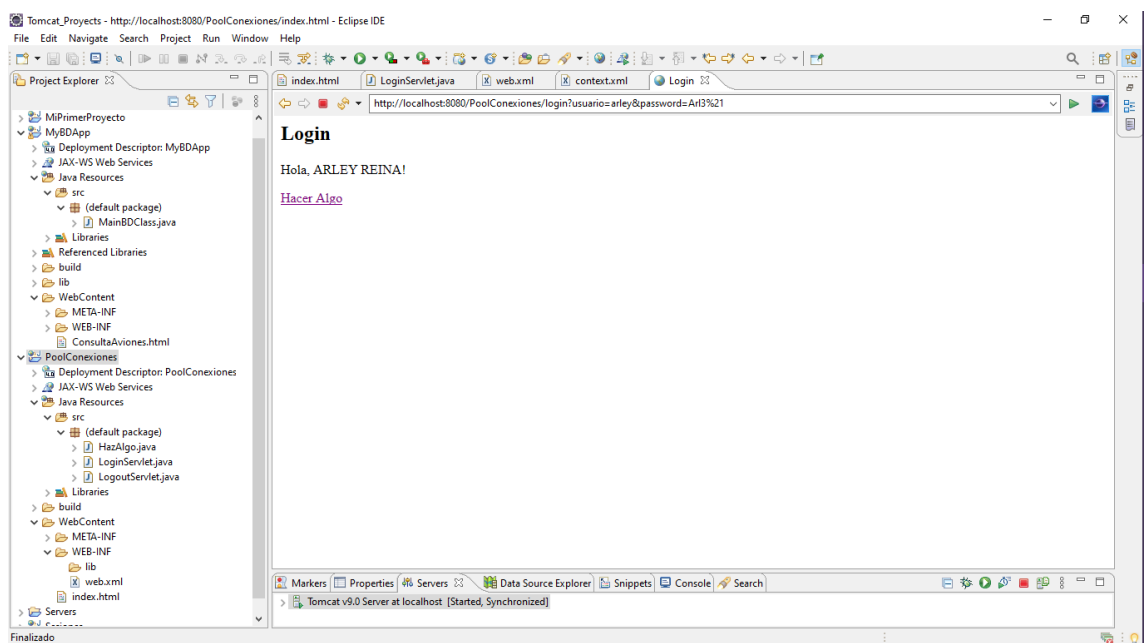
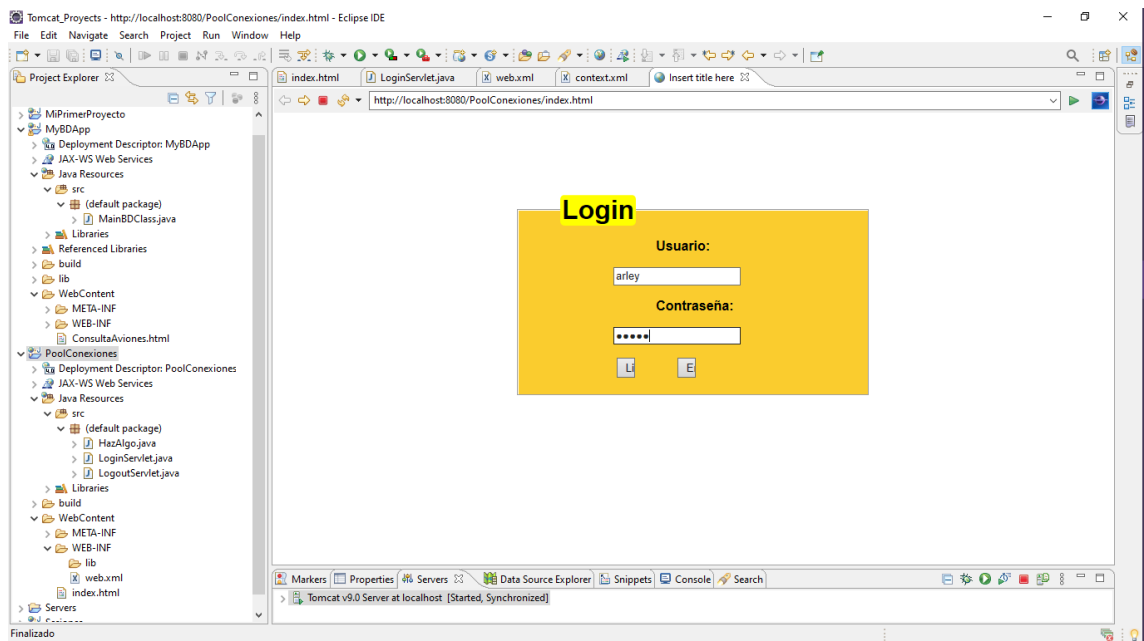
Nombre o contraseña incorrecta

[Vuelve a la página principal](#)

Y ahora el correcto:



Y el mismo resultado obtenemos en Eclipse:

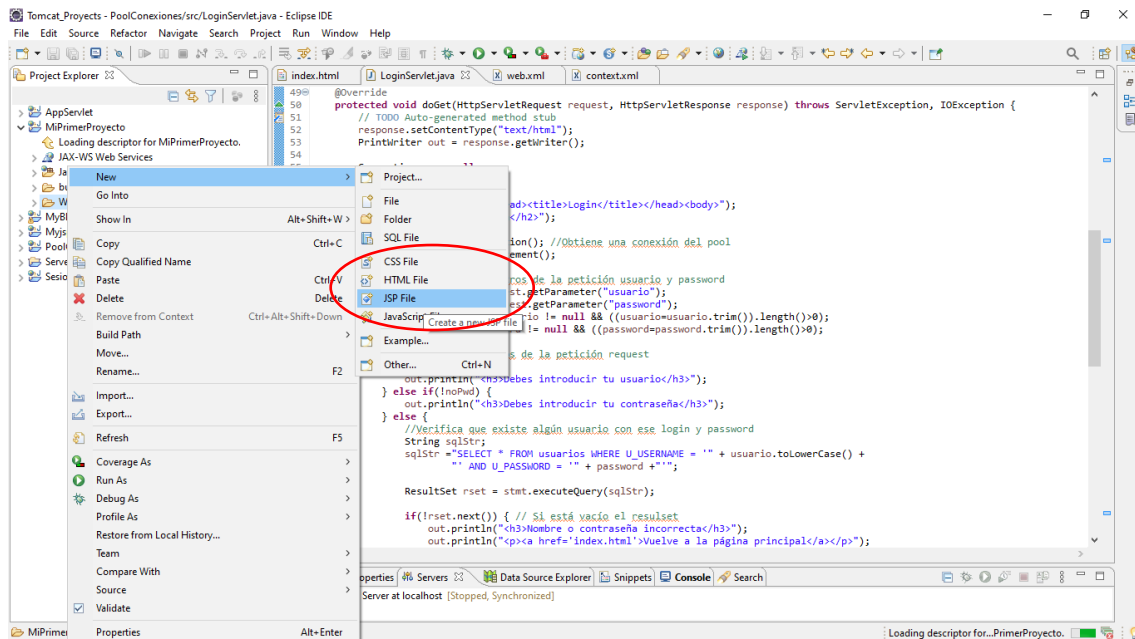


## 7. Creación de Páginas con JSP:

Creamos un proyecto nuevo llamado “MyjspPage”, es aquí donde crearemos la página JSP. Haremos 3 ejemplos con las funcionalidades más importantes de JSP.

### a. Primer ejemplo con expresiones básicas:

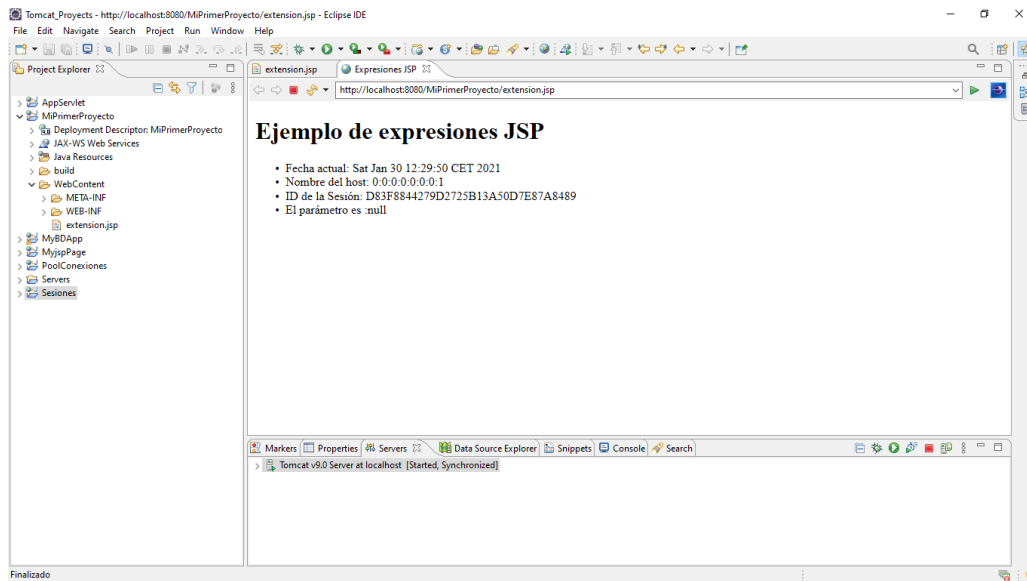
La primera página se llamará **expresiones.jsp** que tendremos que crearla en el directorio donde se crean las páginas html, o sea, en **Web-Content**.



Y este es el contenido de la página:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Expresiones JSP</title>
</head>
<body>
    <h1>Ejemplo de expresiones JSP</h1>
    <ul>
        <li>Fecha actual: <%=new java.util.Date()%></li>
        <li>Nombre del host: <%=request.getRemoteHost() %></li>
        <li>ID de la Sesión: <%=session.getId() %></li>
        <li>El parámetro es :<%=request.getParameter("nombre") %></li>
    </ul>
</body>
</html>
```

Y estos sus resultados:



b. Segundo ejemplo con contador de veces visitadas:

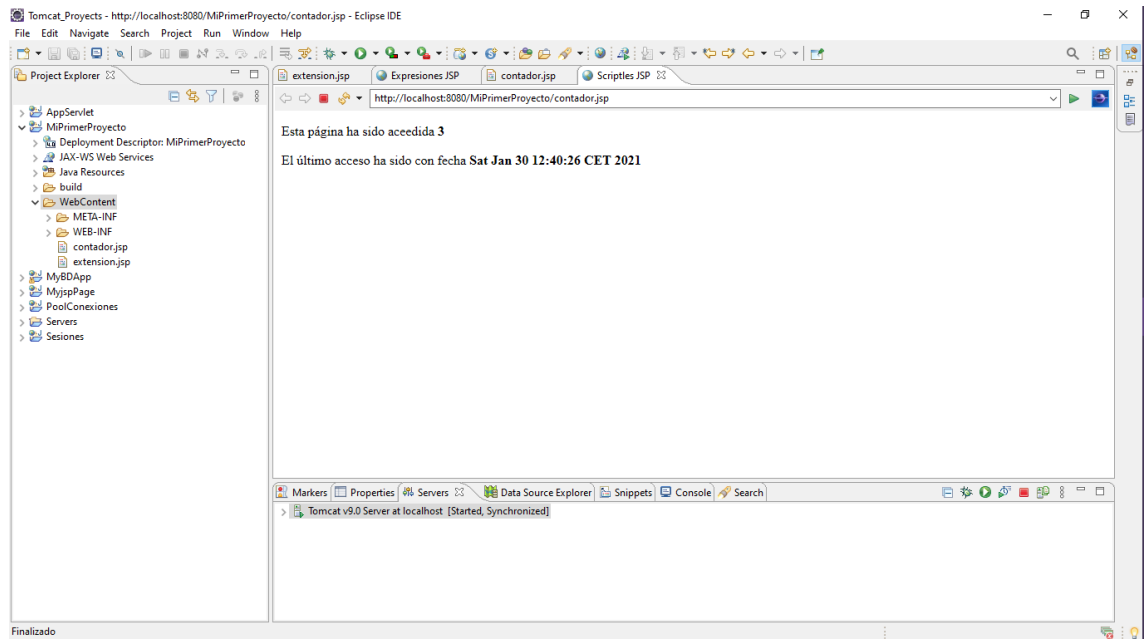
Para este segundo ejemplo, declararemos una variable que pueda contar cuantas veces hemos visitado la misma página.

Crearemos la página **contador.jsp**, con su correspondiente código:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Scriptles JSP</title>
</head>
<%@page import="java.util.*" %>
<!--Con esto podemos comentar el código JSP-->
<%!
    private int cont = 0;
    private Date fecha = new Date();
%>
<body>
    <p>Esta página ha sido aceedida <b><%=++cont %></b></p>
    <p>El último acceso ha sido con fecha <b><%=fecha%></b></p>
    <% fecha=new Date(); %>
</body>
</html>
```

Y estos son los resultados:





Cómo podemos ver, las variables declaradas siempre empiezan por “!” y para usarse, empiezan por “=”.

### c. Tercer ejemplo con conexión a la base de datos:

Por último, emularemos lo hecho en los Servlets anteriores, mostrando en pantalla los datos de una de las tablas a través de la página JSP.

Crearemos una página llamada consulta y mostraremos los resultados:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Consulta de Bases</title>
<style>
    body{
        font-family:"Gill Sans Extrabold", Helvetica, sans-serif;
    }
    table, tr, th, td{
        border: 2px solid black;
        border-collapse: collapse;
    }
    th{
        background-color: green;
        color: white;
    }
    h1{
        text-align:center;
    }
</style>
</head>
<body>
    <h1>Bases de Aviones y Personal</h1>
    <h3>Elige una Base: </h3>
    <form method="get">
```

```

        <p><input                                type="checkbox"                                name="base"
value="CARROLL"/>Carroll</p>
        <p><input type="checkbox" name="base" value="ROSE"/>Rose</p>
        <p><input type="checkbox" name="base" value="NEKOMA"/>Nekoma</p>
        <p><input                                type="checkbox"                                name="base"
value="MAUNSELL"/>Maunsell</p>
        <p><input type="checkbox" name="base" value="HETEL"/>Hetel</p>
        <p><input type="submit" value="consulta"/></p>
    </form>
    <%
        String[] bases = request.getParameterValues("base");
        if(bases != null) {
    %>
    <%@ page import = "java.sql.*" %>
    <%
        // Paso 1: Cargar el driver JDBC.
        Class.forName("com.mysql.cj.jdbc.Driver");
        //Paso 2: Conectarse a la Base Datos utilizando la clase
Connection
        String userName="root";
        String password="cr1ss4lley12#";
        String url="jdbc:mysql://localhost:3306/aerovuelo";
        Connection conn = DriverManager.getConnection(url, userName,
password);
        //Paso 3: Crear la sentencia SQL, utilizando objetos de tipo
Statement
        Statement stmt = conn.createStatement();
        String sqlStr = "SELECT * FROM bases WHERE ";
        for (int i=0; i < bases.length; i++){
            sqlStr= sqlStr + "B_NOMBRE = '" + bases[i] + "'";
            if(i != bases.length-1){
                sqlStr += "OR ";
            }
        }
        sqlStr += "ORDER BY B_NOMBRE";

        //para depuración
        System.out.println("La consulta sql es " + sqlStr);
        //Paso 4: Ejecutar las sentencias SQL a través de los objetos
Statement
        ResultSet rset = stmt.executeQuery(sqlStr);
    %>
    <hr/>
    <table>
        <tr>
            <th>ID de la base</th>
            <th>Nombre de la Base</th>
        </tr>
    <%
        //Paso 5: Procesar el conjunto de registros resultante utilizando
ResultSet
        while(rset.next()){
    %>
    <tr>
        <td><%= rset.getString("idBASES") %></td>
        <td><%= rset.getString("B_NOMBRE") %></td>
    </tr>
    <% } %>
    </table>

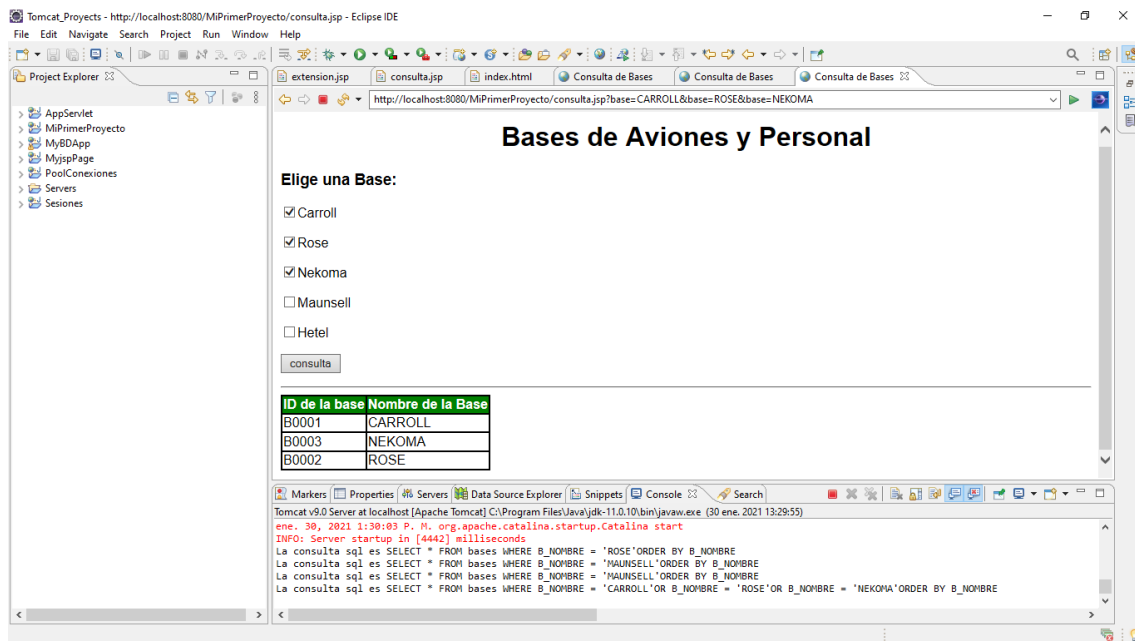
```

```

<%
    //Cierre de Recursos
    rset.close();
    stmt.close();
    conn.close();
}
%>
</body>
</html>

```

Y este es el resultado:



## 8. Conclusiones:

- El despliegue de aplicaciones a través de Tomcat ha sido un trabajo extenso, ya que no es sencillo usarlo en las primeras veces. La curva de aprendizaje es más sencilla cuándo ya se manejan conceptos básicos acerca del entorno o del lenguaje Java, de otro modo, gran parte de lo desarrollado tomará tiempo.
- Almacenar y administrar aplicaciones sí que me parece un punto positivo a rescatar en Tomcat, ya que su interfaz al ser sencilla y simple, no es necesario buscar mucho para buscar el proyecto. Lo negativo de esto es que todavía no hay una herramienta implementada para poder eliminar proyectos directamente desde el **Manager Applications**, lo que significa que, si hemos desplegado aplicaciones, tenemos que ir a la carpeta raíz para eliminarlos.
- Otro punto tedioso desde mi consideración es lo ortodoxo que es actualizar una aplicación cuándo se despliega en la carpeta raíz y se muestra por el navegador del ordenador: es un camino un poco extenso tener que eliminar todo el proyecto y luego corregir los errores para volver al exportar el proyecto al Tomcat. Tomando en cuenta que para se vean las actualizaciones de cualquier tipo se tiene que reiniciar, me parece que es algo muy tedioso, más para personas que recién conocen entornos de este tipo.

- Por último, creo que Tomcat tiene un potencial muy grande, porque puede manejar casi lo que cualquier contenedor de aplicaciones hace, con mejor fluidez en muchos casos (en comparación a Simfony, por ejemplo). Lo que me parece necesario es hacer más accesible su manejo y sus formas de usarse: se podrían desarrollar librerías más complementadas o unidas que hagan las funciones que el usuario tiene que buscar de manera externa (cómo la conexión a las bases de datos) y tener un tutorial más extenso. Aunque bien es entendible que sea una organización sin fines de lucro, pero deben darle mejor soporte en materia de guías al usuario y facilitar las herramientas adicionales desde la página.

## 9. Bibliografía:

- <https://tomcat.apache.org/>
- <https://www.apachefriends.org/es/index.html>
- [https://aulavirtual.murciaeduca.es/pluginfile.php/2967262/mod\\_resource/content/1/despliegue\\_aplicaciones\\_web\\_garceta.pdf](https://aulavirtual.murciaeduca.es/pluginfile.php/2967262/mod_resource/content/1/despliegue_aplicaciones_web_garceta.pdf)
- <https://gigastur.es/solucionar-problema-tomcat>
- <https://dev.mysql.com/downloads/connector/j/>
- <https://www.java.com/es/>
- <https://www.arquitecturajava.com/tomcat-context-xml-su-configuracion/>
- [¿Cómo resolver el problema java.lang.classnotfoundexception: com.mysql.jdbc.driver? - Stack Overflow en español](https://es.stackoverflow.com/questions/100000/como-resolver-el-problema-java-lang-classnotfoundexception-com-mysql-jdbc-driver)