

INTRODUCCIÓN:

El objetivo de este reto es instalar y configurar el servidor contenedor de servlets conocido como Tomcat en la red interna utilizada para los retos anteriores. Para este reto por lo tanto necesitaremos utilizar el siguiente software:

- Máquina virtual con Debian 10 como sistema operativo.
- Tomcat 9.

La mencionada máquina virtual será la máquina virtual "bastis" instalada para un reto anterior cuya configuración deberá ser la siguiente:

- Nombre: bastis.atenea.olimpo.god
- Alias: opencms.atenea.olimpo.god
- Dirección IP: 10.130.6.253
- OS: Debian 10

Tras instalar Tomcat se busca también desplegar una aplicación y limitar el acceso a la misma mediante los distintos métodos de autenticación y seguridad que ofrece el servidor de contenedores.

INSTALACIÓN DE TOMCAT:

PREREQUISITOS

Tomcat está desarrollado en Java por lo que necesitaremos instalar el JDK en nuestra máquina virtual; para ello:

```
# apt-get install default-jdk
```

Hecho esto, procederemos a crear un Nuevo usuario, ya que Tomcat no puede ser lanzado por un usuario administrador por motivos de seguridad. Crearemos por ello un usuario llamado tomcat cuyo grupo sea también tomcat.

```
# groupadd tomcat  
# useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
```

Además de crear el usuario, lo estamos configurando de modo que:

- No se pueda iniciar sesión con este usuario (-s /bin/false).
- Su carpeta de usuario sea el directorio de instalación de Tomcat (-d /opt/tomcat)
- Pertenezca al grupo que acabamos de crear (-g tomcat)

Hechos estos pasos podemos proceder a la instalación de Tomcat, pero configuraremos primero el nombre alternativo de la máquina en nuestro DNS.

```
/etc/bind/db.atenea.olimpo.god
```

```
[...Contenido anterior...]  
opencms IN A 10.130.6.253
```

```
/etc/bind/db.10.130.6
```

```
[...Contenido anterior...]  
253 IN PTR opencms.atenea.olimpo.god.
```

INSTALACIÓN

La versión estable actual de Tomcat es la versión 9.0.41, por lo que esta será la que utilizaremos. Para lo que necesitamos realizar la versión core será suficiente.

Descargaremos la distribución binaria del programa mediante curl; si no se dispone de este programa podemos instalarlo mediante el siguiente comando.

```
# apt-get install curl
```

Para obtener el link de descarga (y la última versión estable actual) podemos [consultar la web oficial](#).

En nuestro caso:

```
# curl -O https://ftp.cixug.es/apache/tomcat/tomcat-9/v9.0.41/bin/apache-tomcat-9.0.41.tar.gz
```

Realizaremos la instalación en "/opt/tomcat", por ello:

```
# mkdir /opt/tomcat
# tar xzvf apache-tomcat-9*.tar.gz -C /opt/tomcat --strip-components=1
```

Necesitaremos ahora dar permisos al grupo "TOMCAT" sobre el directorio de la aplicación y cambiar los permisos sobre múltiples directorios (además de asignarles el propietario correcto).

```
# cd /opt/tomcat
# chgrp -R tomcat /opt/tomcat
# chmod -R g+r conf
# chmod g+x conf
# chown -R tomcat webapps/ work/ temp/ logs/
```

CREAR SERVICIO PARA TOMCAT

Queremos que Tomcat se lance como un servicio en segundo plano, por lo que configuraremos el archivo del servicio systemd.

Crearemos el archivo "tomcat.service" en "/etc/systemd/system" con el siguiente contenido:

```
/etc/systemd/system/tomcat.service
/etc/systemd/system/tomcat.service
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh
```

```
User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

La línea que menciona JAVA_HOME deberá ser puesta acorde con la versión de java que estamos utilizando, si esta se desconoce podemos buscarla mediante el siguiente comando.

```
# update-java-alternatives -l
```

Una vez creado el archivo deberemos reiniciar el servicio encargado de controlar servicios para aplicar los cambios y ya podremos iniciar Tomcat como un servicio más de nuestro sistema, por lo que podemos habilitar su inicio automático.

```
# systemctl daemon-reload
# systemctl tomcat start
# systemctl tomcat enable
# systemctl tomcat status
# [...]
```

CONFIGURACIÓN DE FIREWALL Y ADMINISTRACIÓN WEB DE TOMCAT

Con nuestro setup actual tenemos activado el Firewall UFW habilitado por defecto, y este bloquea las peticiones en el puerto 8080 (Puerto por defecto de Tomcat), tendremos que configurar el firewall para habilitarlas, para ello:

```
# ufw allow 8080
```

Con esto ya tendremos acceso a nuestro servidor, pero tendremos que hacer algún paso extra para habilitar la consola de administración web de Tomcat; tendremos que crear un usuario con permisos sobre "manager-gui" y "admin-gui", esto podemos configurarlo en el archivo "tomcat-users.xml" accesible en la carpeta config del software.

Añadiremos las siguientes líneas tras el contenido ya existente:

/opt/tomcat/conf/tomcat-users.xml

```
<role rolename="manager-gui"/>
<role rolename="admin"/>
<user name="admin" password="admin" roles="manager-gui, admin-gui"/>
```

De forma predeterminada, las versiones más recientes de Tomcat restringen el acceso a las aplicaciones Manager y Host Manager a las conexiones que provienen del propio servidor.

Si se desea eliminar o modificar esta restricción se debe cambiar las restricciones de la dirección IP en los archivos context.xml apropiados.

Por ejemplo, para el administrador de aplicaciones, comentaremos la línea que define la restricción:

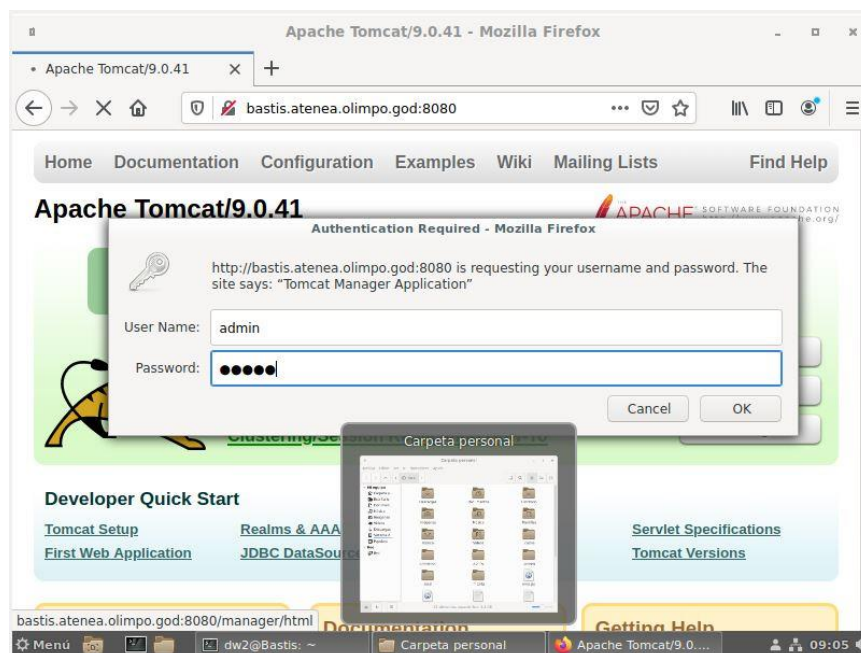
```
/opt/tomcat/webapps/manager/META-INF/context.xml
[...]  
<!-- <Valve className="org.apache.catalina.valves.RemoteAddrValve"  
      allow="127\.\d+\.\d+\.\d+/::1/0:0:0:0:0:0:1" /> -->  
[...]
```

Hecho esto tendremos que reiniciar el servicio para aplicar los cambios realizados.

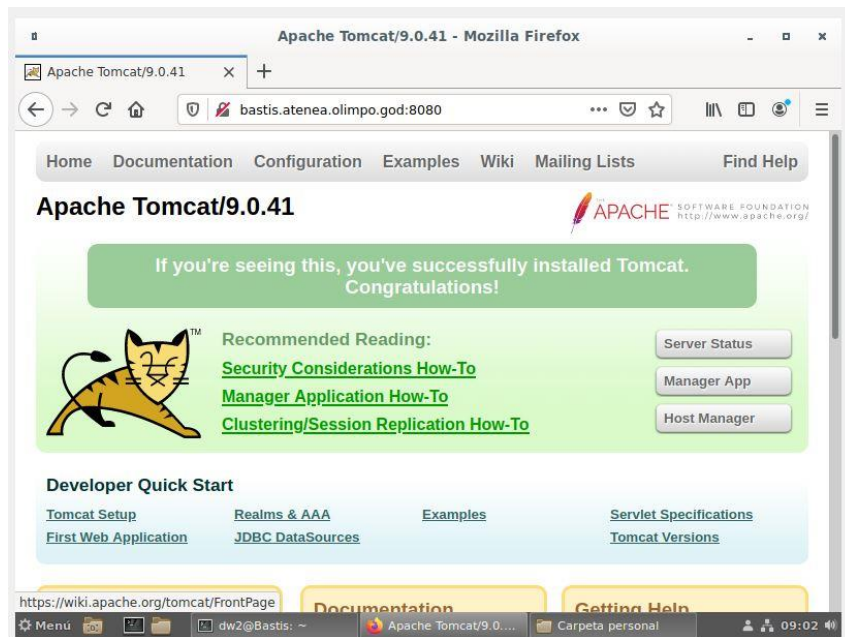
```
# systemctl tomcat restart
```

ACCESO WEB

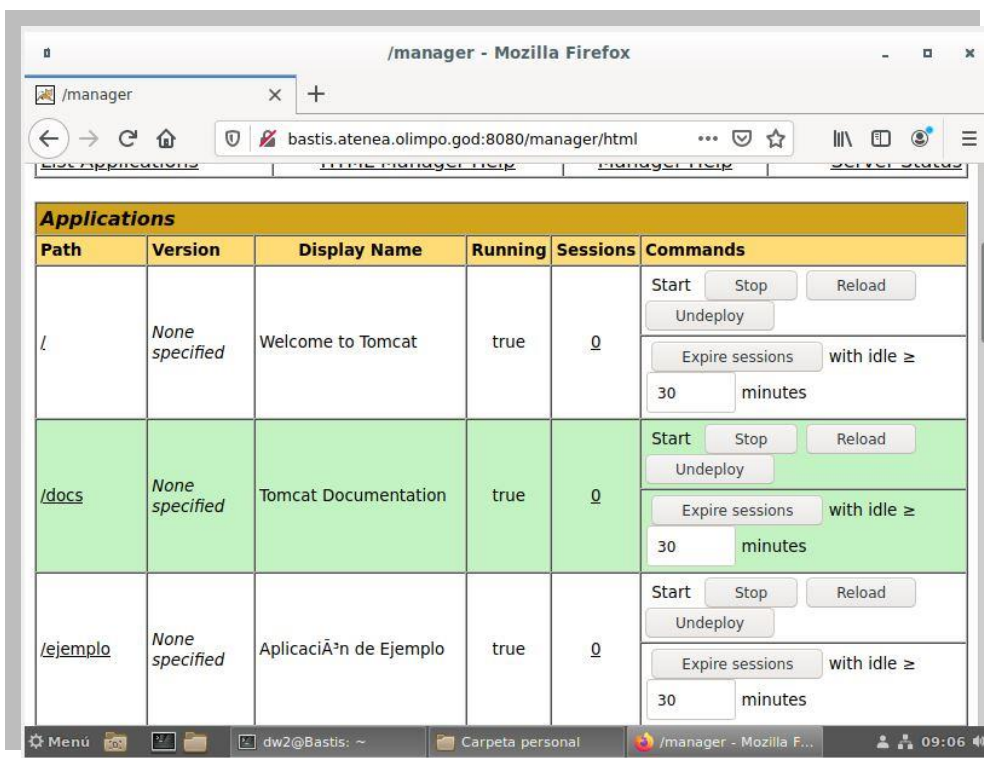
Con esto realizado podemos acceder tanto desde la propia máquina como desde el exterior tanto por IP como por nombre.



El usuario y contraseña solicitados serán los configurados en tomcat-users.xml.



Además, deberíamos poder acceder a “Manager App” gracias a los cambios en el contexto de la aplicación.



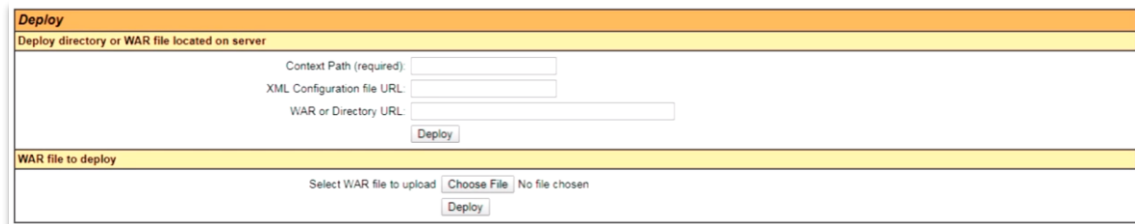
DESPLEGAR APLICACIÓN DE EJEMPLO:

Ahora ya tenemos Tomcat configurado y funcionando y podemos proceder a desplegar cualquier aplicación. En este caso desplegaremos una aplicación de ejemplo mediante su archivo WAR.

Hay muchas maneras de desplegar un WAR en Tomcat, este proceso puede hacerse mediante comandos, mediante interfaz gráfica y muchos IDE permiten el despliegue desde el propio programa.

La manera más sencilla teniendo acceso al servidor es colocar el WAR en la carpeta webapps de Tomcat, y si el servicio está activo, el sistema comenzará automáticamente a extraer el programa (si no está activo comenzará en el momento que se inicie el servicio).

Otra opción si no se dispone de acceso remoto al server, es acceder desde el panel de administración de Tomcat, y desde allí desplazarse al apartado "DEPLOY" y añadirlo desde la interfaz gráfica subiendo el archivo.



Hecha cualquiera de estas opciones la aplicación quedará desplegada y será también visible y administrable desde gestor de aplicaciones de Tomcat.

CONFIGURACIONES DE AUTENTICACIÓN Y ACCESO A RECURSOS:

Finalmente podemos proceder a configurar el acceso a nuestra aplicación mediante los distintos métodos de autenticación que ofrece Tomcat.

AUTENTICACIÓN "BASIC"

La autenticación "basic" es la más sencilla que ofrece Tomcat. Los roles y usuarios (con sus contraseñas) serán configurados en texto plano en el archivo tomcat-users.xml que hemos editado anteriormente para configurar el acceso al application manager.

Para los requisitos solicitados para el reto configuraremos el archivo de la siguiente manera:

```
/opt/tomcat/conf/tomcat-users.xml

[...]
```

```
<!-- Tomcat manager -->
<role rolename="manager-gui"/>
<role rolename="admin"/>
<user name="admin" password="admin" roles="manager-gui,admin-gui"/>

<!-- Definir los roles -->
<role rolename="teacher"/>
<role rolename="student"/>

<!-- Crear usuarios con rol "student" -->
<user username="usuario" password="usuario" roles="student"/>
<user username="usuario2" password="usuario2" roles="student"/>
<user username="usuario3" password="usuario3" roles="student"/>

<!-- Crear un usuario con 2 roles "teacher" y "student" -->
<user username="profesor" password="profesor" roles="teacher, student"/>
</tomcat-users>
```

Con esto tendremos configurado los roles "teacher" y "student", y estos roles aplicados para 4 usuarios distintos (además de lo configurado anteriormente para acceder al panel de administración).

Con esto hecho solo falta acceder al archivo web.xml para el que queramos configurar la autenticación de contenedor básica, y añadir el siguiente contenido.

```
/opt/tomcat/webapps/ejemplo/WEB-INF/web.xml

<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd" version="2.4">

  <display-name>Aplicación de Ejemplo</display-name>
  <description>
    Aplicación WEB de ejemplo
  </description>

  <servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>mypackage.Hello</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>

  <security-constraint>
    <display-name>Acceso solo para profesores</display-name>
    <web-resource-collection>
      <web-resource-name>Zona profesores</web-resource-name>
      <url-pattern>/privado/*</url-pattern>
    </web-resource-collection>

    <auth-constraint>
      <role-name>teacher</role-name>
    </auth-constraint>
  </security-constraint>

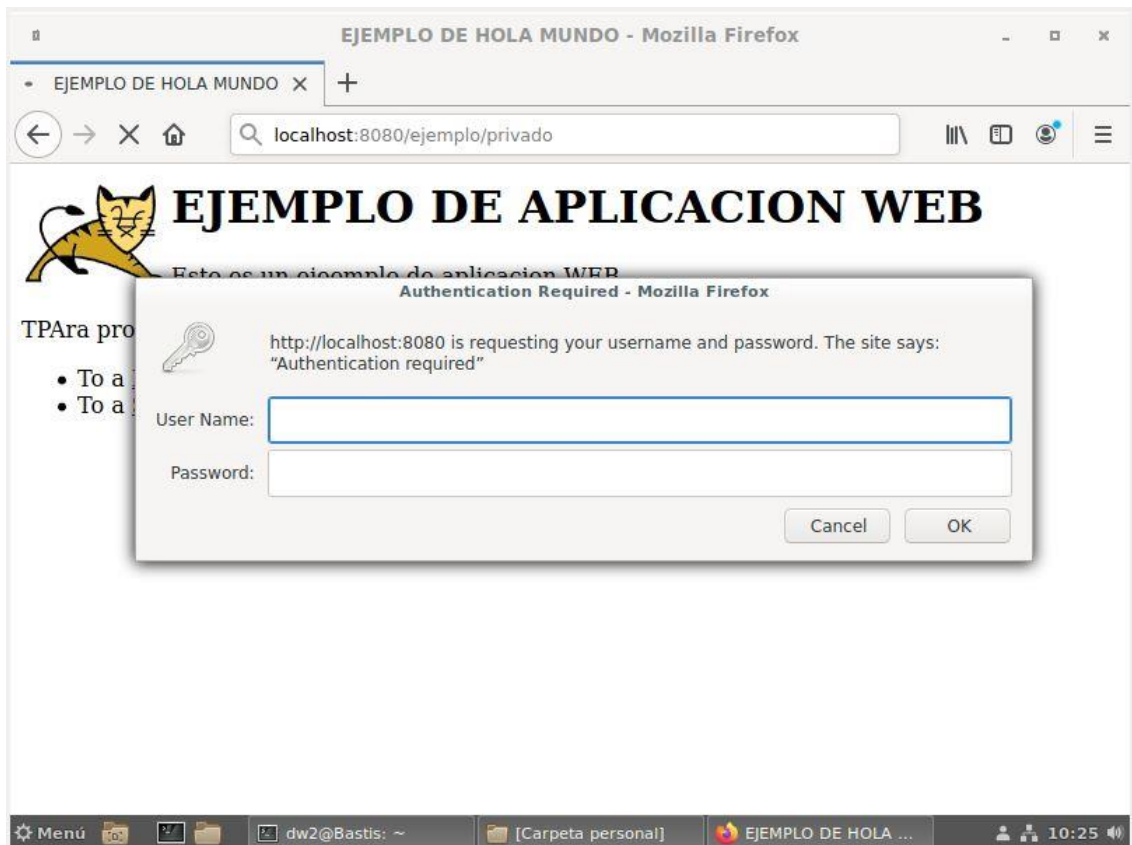
  <security-constraint>
    <display-name>Acceso para alumnos</display-name>
    <web-resource-collection>
      <web-resource-name>Zona compartida</web-resource-name>
      <url-pattern>/publico/*</url-pattern>
    </web-resource-collection>

    <auth-constraint>
      <role-name>student</role-name>
    </auth-constraint>
  </security-constraint>

  <login-config>
    <auth-method>BASIC</auth-method>
  </login-config>
</web-app>
```

Con esta configuración definimos dos zonas mediante las etiquetas de <security-constraint>, una publica para los usuarios "student" y otra privada para los usuarios "teacher". Especificamos los recursos para cada zona mediante la etiqueta <url-pattern> y el rol que tiene permitido el acceso mediante <auth-constraint>.

Finalmente, mediante las etiquetas <login-config> y <auth-method> decidiremos que método de autenticación se utilizará, en este caso, siendo la autenticación básica no necesitamos configurar ningún otro parámetro, y de esta manera Tomcat se encargará de mostrar un dialogo para el login de manera autónoma.



AUTENTICACIÓN "FORM"

Este método de autenticación es similar al método basic, por lo que los pasos que comparten estos métodos no serán nuevamente mostrados.

La mayor diferencia entre estos métodos de autenticación es que en este caso, nos encargaremos nosotros de mostrar el login de la aplicación, es decir, Se utiliza un formulario de login personalizado que no depende del navegador web. Este formulario debe cumplir algunas características para que se pueda comunicar correctamente con el subsistema de autenticación Tomcat.

Deberemos crear dos archivos jsp y añadirlos en el apartado público (web-content) de nuestra aplicación; estos pueden llamarse de la manera que deseemos, en nuestro caso serán "login.jsp" y "error.jsp" y tendrán el siguiente contenido.

/opt/tomcat/webapps/ejemplo/web/Login.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Insert title here</title>
  </head>
  <body>
    <form name='login' action='${pageContext.request.contextPath}/j_security_check' method='POST'>
      User:<br>
      <input type='text' name='j_username'><br> Pass:<br> <input type='password' name='j_password'><br>
      <input name="submit" type="submit" value="Acceder">
    </form>
  </body>
</html>
```

```

/opt/tomcat/webapps/ejemplo/web/error.jsp

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/Loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Error</title>
</head>
<body>
<form name='Login' action='${pageContext.request.contextPath}/j_security_check'
method='POST'>
    User:<br> <input type='text' name='j_username'>
    <br>Pass:<br> <input type='password' name='j_password'><br>
    <input name="submit" type="submit" value="Acceder">
</form>
<div align="center">
<p style="color:red"> User or Pass incorrect!</p>
</div>
</body>
</html>

```

Finalmente, para implementar este método simplemente tendremos que cambiar el `<auth-method>` configurado en el `web.xml` de la aplicación, especificando el `jsp` de login y el `jsp` ante posibles errores de autenticación.

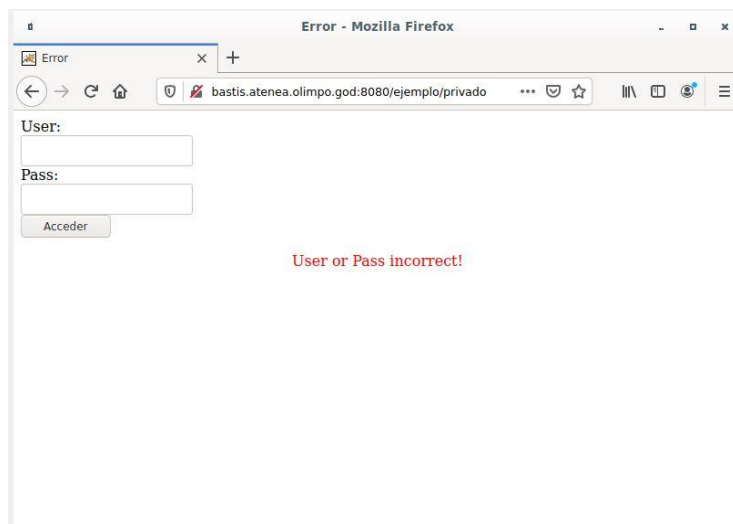
```

/opt/tomcat/webapps/ejemplo/web/WEB-INF/web.xml

[...]
<login-config>
    <auth-method>FORM</auth-method>
    <form-login-config>
        <form-login-page>/Login.jsp</form-login-page>
        <form-login-page>/error.jsp</form-login-page>
    </form-login-config>
</login-config>

```

Hecho esto, los usuarios, roles y contraseñas no habrán cambiado, pero el login será solicitado nuestros `jsp`.



AUTENTICACIÓN "FORM" CON ACCESO A BBDD (JDBC)

Finalmente configuraremos el acceso a la aplicación mediante roles, usuarios y contraseñas obtenidos desde una base de datos. En este caso tendremos que instalar y configurar una base de datos MariaDB.

Siendo que este es un proceso ya explicado no entraremos en profundidad en la instalación de este software:

En resumen, los pasos seguidos han sido:

1. Descargar mariadb-server
2. Configuraciones básicas mediante mysql-secure-installation
3. Instalar PHP
4. Instalar PhpMyAdmin.

Teniendo ya instalado un sistema gestor de bases de datos en nuestro sistema tendremos que descargar el driver correspondiente a dicho sistema. En nuestro caso el conector JDBC para [MariaDB](#).

Una vez descargado tendremos que almacenar el archivo .jar en la carpeta /opt/tomcat/lib.

Los roles y usuarios con este método de autenticación los obtendremos de dos tablas, por lo que primero las tendremos que crear.

Haremos este proceso desde PhpMyAdmin:

1. Crear una base de datos (en nuestro caso la llamaremos ejemplo)
2. Crear las 2 tablas necesarias

(descripción de la estructura de las tablas)

```
table users (  
    user_name varchar(45),  
    user_pass varchar(45)  
);  
  
table users_roles (  
    user_name varchar(45),  
    role_name(45)  
);
```

3. Introducir los roles y usuarios en las tablas creadas.

Para cambiar el origen de los usuarios y roles tendremos que en esta ocasión editar el fichero "server.xml"; donde, en este caso, configuraremos la autenticación mediante base de datos a nivel de aplicación (únicamente para el contexto de la aplicación "ejemplo").

Para ello:

/opt/tomcat/conf/server.xml

```
[...]  
<Context path="/ejemplo">  
    <Realm className="org.apache.catalina.realm.JDBCRealm"  
        driverName="org.mariadb.jdbc.Driver"  
        connectionName="dw2"  
        connectionPassword="dw2"  
        connectionURL="jdbc:mariadb://localhost/ejemplo"  
        userTable="users"  
        userNameCol="user_name"  
        userCredCol="user_pass"
```

```
        userRoleTable="users_roles"  
        roleNameCol="role_name"  
    />  
</Context>  
</Host>  
</Engine>  
</Service>  
</Server>
```

En el apartado <context> para nuestro contexto seleccionado tendremos que configurar el driver JDBC que hemos descargado, la URL de conexión (y credenciales de acceso) a la base de datos y las tablas correspondientes a roles y usuarios (además de sus columnas).

De esta manera el servidor implementará (únicamente para la aplicación de ejemplo) la autenticación mediante FORM explicada en el apartado anterior, pero obteniendo los usuarios directamente de la base de datos de MariaDB.