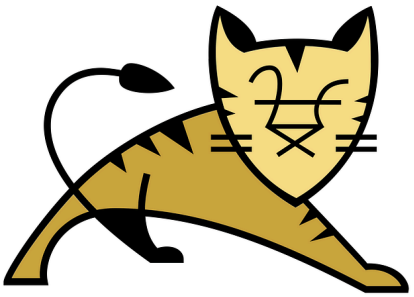
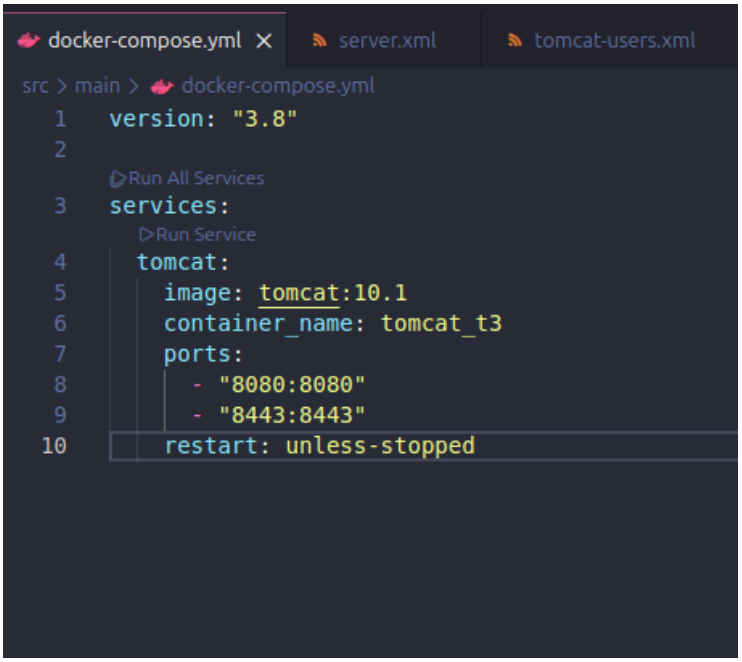


Apache Tomcat

Manuel Morente Díaz

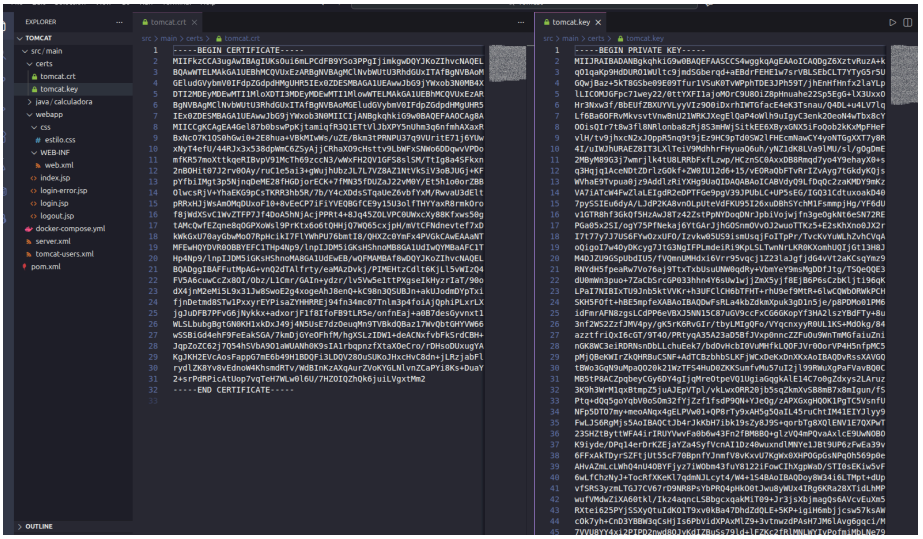
Paso 1:

Se crea el archivo docker-compose.yml utilizando la imagen tomcat:10.1 y se configuran los puertos 8080 y 8443 para iniciar el contenedor con el nombre tomcat_t3.



Paso 2:

Se crean los certificados de seguridad y se asegura su transferencia al contenedor (ruta /usr/local/tomcat/certs) a través de la configuración del archivo docker-compose.yml.



```
usua5pc17@A5PC17:~/Escritorio/DAW/Despliegue/Tomcat/src/main$ cd certs
usua5pc17@A5PC17:~/Escritorio/DAW/Despliegue/Tomcat/src/main/certs$ openssl req -x509 -newkey rsa:4096 -keyout tomcat.key -out tomcat.crt -days 365 -nodes
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:localhost
Email Address []:
usua5pc17@A5PC17:~/Escritorio/DAW/Despliegue/Tomcat/src/main/certs$
```

Paso 3:

Se realiza la instalación de la herramienta Maven en el sistema mediante sudo apt install maven y se verifica la versión instalada con el comando mvn --version.

```
Configurando libplexus-sec-dispatcher-java (2.0-3) ...
Configurando libwagon-file-java (3.5.3-1) ...
Configurando libcommons-io-java (2.11.0-2) ...
Configurando libmaven-resolver-java (1.6.3-1) ...
Configurando libmaven-shared-utils-java (3.3.4-1) ...
Configurando libguava-java (32.0.1-1) ...
Configurando liberror-prone-java (2.18.0-1) ...
Configurando libguice-java (4.2.3-2) ...
Configurando libmaven3-core-java (3.8.7-2) ...
Configurando maven (3.8.7-2) ...
update-alternatives: utilizando /usr/share/maven/bin/mvn para proveer /usr/bin/m
vn (mvn) en modo automático
usua5pc17@A5PC17:~/Escritorio/DAW/Despliegue/Tomcat$ mvn --version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 11.0.29, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: es_ES, platform encoding: UTF-8
OS name: "linux", version: "6.14.0-37-generic", arch: "amd64", family: "unix"
usua5pc17@A5PC17:~/Escritorio/DAW/Despliegue/Tomcat$
```

Paso 4:

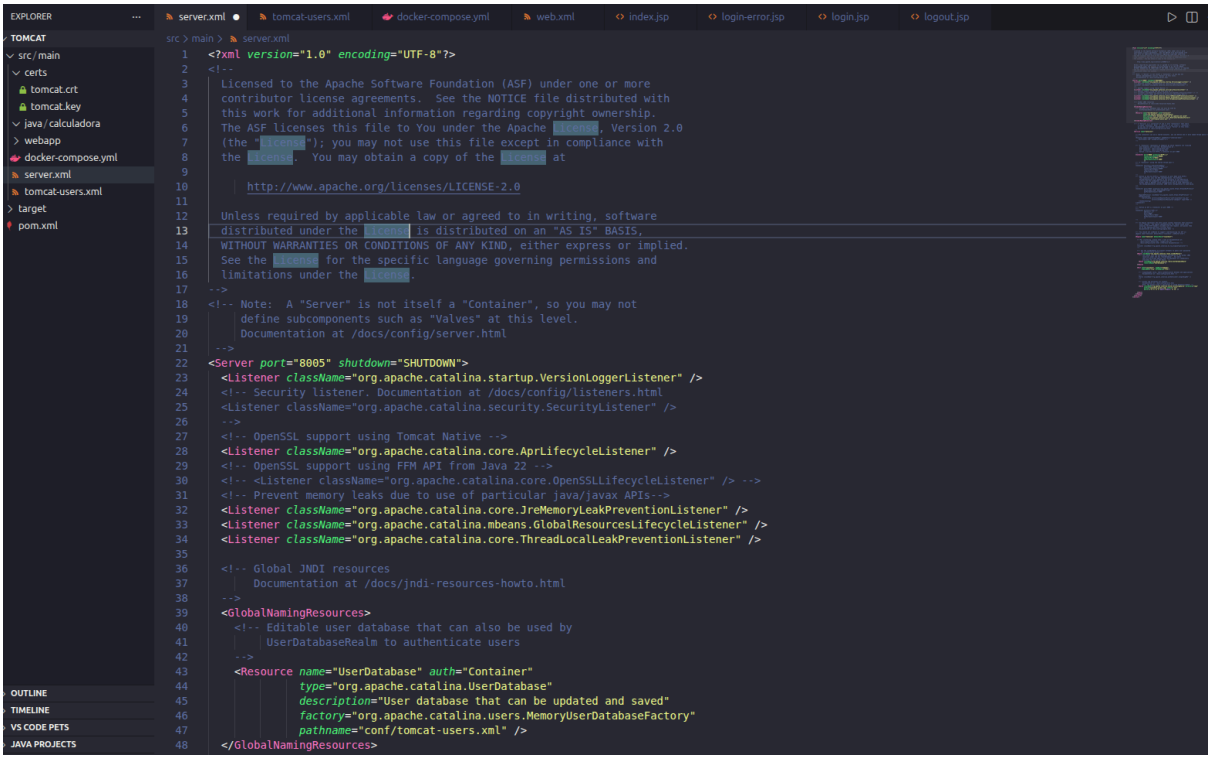
Se configura el archivo pom.xml definiendo las dependencias de jakarta.servlet-api y especificando el uso de Java 17 para la construcción del archivo .war del proyecto.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0">
    <groupId>calculadora</groupId> <!-- Identidad del proyecto -->
    <artifactId>calculadora</artifactId> <!-- artifactId: nombre del proyecto/artefacto -->
    <version>1.0.0</version>
    <packaging>war</packaging> <!-- packaging: indica que el resultado final será un WAR -->
    <properties>
        <!--
        Versión de Java con la que compilamos.
        Recomendación docente: usar 17 para evitar diferencias entre equipos.
        -->
        <maven.compiler.release>17</maven.compiler.release>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
    <!--
    API de Servlets Jakarta (para Tomcat 10).
    scope=provided significa:
    - Se necesita para COMPILAR
    - NO se empaqueta dentro del WAR
    - Porque Tomcat ya incluye esta librería en el servidor.
    -->
        <dependency>
            <groupId>jakarta.servlet</groupId>
            <artifactId>jakarta.servlet-api</artifactId>
            <version>6.0.0</version>
        <scope>provided</scope>
        </dependency>
    </dependencies>
    <build>
        <!--
        finalName controla el nombre del fichero generado, generará:
        target/calculadora.war
        (Esto es útil para Docker)
        -->
        <finalName>calculadora</finalName>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-war-plugin</artifactId>
                    <version>3.4.0</version>
                    <configuration>
                        <!--Si falta web.xml, NO falla el build.-->
                        <failOnMissingWebXml>false</failOnMissingWebXml>
                    </configuration>
                </plugin>
            </plugins>
    </build>
</project>
```

Paso 5:

Se extrae el archivo de configuración original del contenedor en ejecución hacia el directorio local del proyecto usando el comando docker cp tomcat_t3:/usr/local/tomcat/conf/server.xml ./server.xml.

```
usua5pc17@A5PC17:~/Escritorio/DAW/Despliegue/Tomcat/src/main$ sudo docker cp tomcat_t3:/usr/local/tomcat/conf/server.xml ./server.xml
Successfully copied 8.7kB to /home/usua5pc17/Escritorio/DAW/Despliegue/Tomcat/src/main/server.xml
usua5pc17@A5PC17:~/Escritorio/DAW/Despliegue/Tomcat/src/main$
```


EXPLORER
TOMCAT
src/main
certs
tomcat.crt
tomcat.key
java/calculadora
webapp
docker-compose.yml
server.xml
tomcat-users.xml
target
pom.xml
OUTLINE
TIMELINE
VS CODE PETS
JAVA PROJECTS
server.xml
tomcat-users.xml
docker-compose.yml
web.xml
index.jsp
login-error.jsp
login.jsp
logout.jsp
src > main > server.xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<!-- Note: A "Server" is not itself a "Container", so you may not
define subcomponents such as "Valves" at this level.
Documentation at /docs/config/server.html
-->
<Server port="8005" shutdown="SHUTDOWN">
<Listener className="org.apache.catalina.startup.VersionLoggerListener" />
<!-- Security listener. Documentation at /docs/config/listeners.html
<Listener className="org.apache.catalina.security.SecurityListener" />
-->
<!-- OpenSSL support using Tomcat Native -->
<Listener className="org.apache.catalina.core.AprLifecycleListener" />
<!-- OpenSSL support using FFM API from Java 22 -->
<!-- <Listener className="org.apache.catalina.core.OpenSSLLifecycleListener" /> -->
<!-- Prevent memory leaks due to use of particular java/javax APIs-->
<Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
<Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
<Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />
<!-- Global JNDI resources
Documentation at /docs/jndi-resources-howto.html
-->
<GlobalNamingResources>
<!-- Editable user database that can also be used by
UserDatabaseRealm to authenticate users
-->
<Resource name="UserDatabase" auth="Container"
type="org.apache.catalina.UserDatabase"
description="User database that can be updated and saved"
factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
pathname="conf/tomcat-users.xml" />
</GlobalNamingResources>

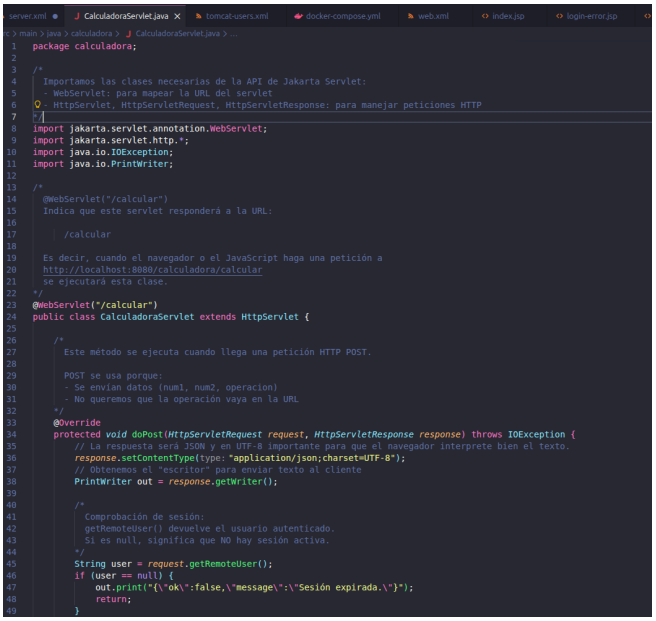
Paso 6:

Se edita el archivo server.xml local para buscar y descomentar el bloque del conector SSL, especificando las rutas internas del contenedor donde se encuentran los certificados generados.

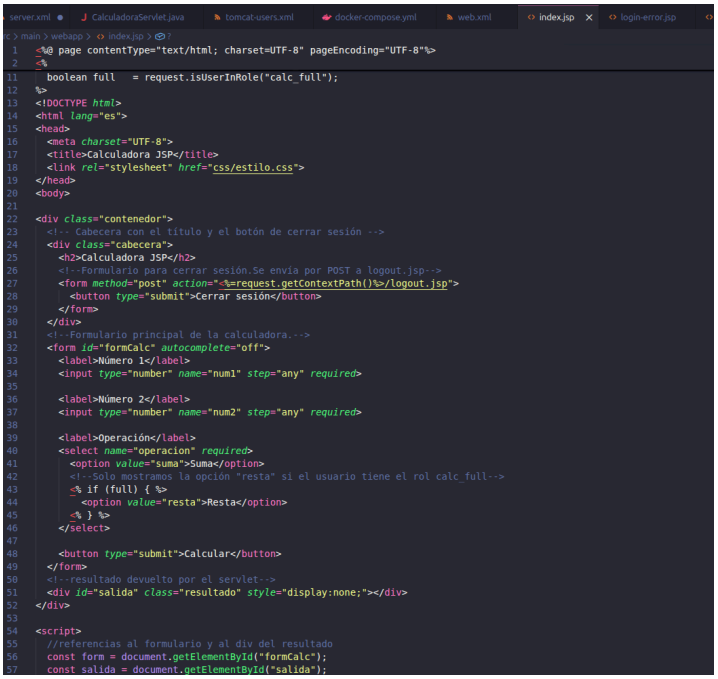
```
        >
        <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
        -->
        <SSLHostConfig>
            <Certificate certificateFile="/usr/local/tomcat/certs/tomcat.crt"
                         certificateKeyFile="/usr/local/tomcat/certs/tomcat.key"
                         type="RSA" />
        </SSLHostConfig>
        <!--
    </Connector>
```

Paso 7:

Se desarrolla la lógica de programación en un archivo Java y se diseña la interfaz de usuario en un archivo index.jsp, tomando como referencia los ejemplos proporcionados pero aplicando un diseño propio.



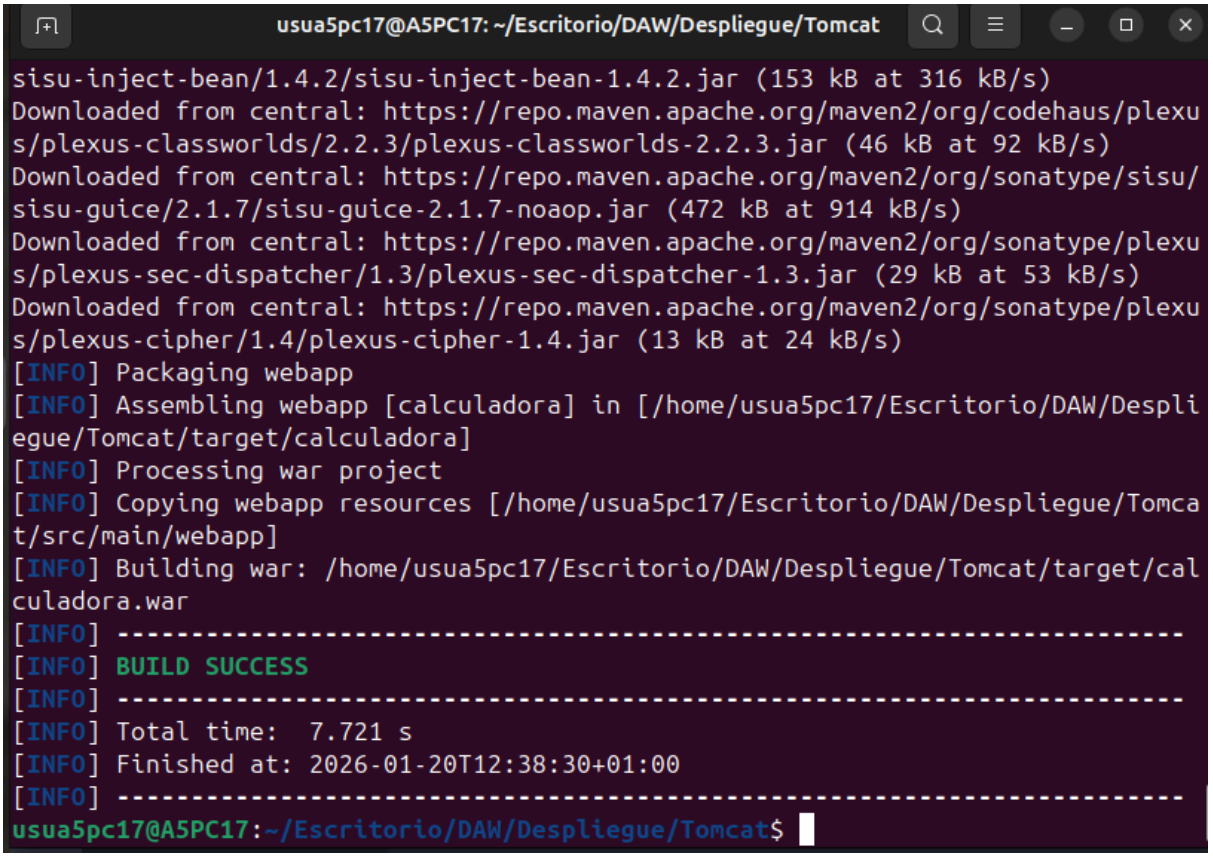
```java
package calculadora;

/*
 Importamos las clases necesarias de la API de Jakarta Servlet:
 - WebServlet: para mapear la URL del servlet
 - HttpServlet, HttpServletRequest, HttpServletResponse: para manejar peticiones HTTP
*/
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.*;
import java.io.IOException;
import java.io.PrintWriter;

/*
 @WebServlet("/calcular")
 Indica que este servlet responderá a la URL:

     /calcular

 Es decir, cuando el navegador o el JavaScript haga una petición a
 http://localhost:8080/calculadora/calcular
 se ejecutará esta clase.
*/
@WebServlet("/calcular")
public class CalculadoraServlet extends HttpServlet {

    /*
     Este método se ejecuta cuando llega una petición HTTP POST.

     POST se usa porque:
     - Se envían datos (num1, num2, operacion)
     - No queremos que la operación vaya en la URL
    */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {
        // La respuesta será JSON y en UTF-8 importante para que el navegador interprete bien el texto.
        response.setContentType(type: "application/json;charset=UTF-8");
        // Obtenemos el "escritor" para enviar texto al cliente
        PrintWriter out = response.getWriter();

        /*
         Comprobación de sesión:
         getRemoteUser() devuelve el usuario autenticado.
         Si es null, significa que NO hay sesión activa.
        */
        String user = request.getRemoteUser();
        if (user == null) {
            out.print("{\"ok\":false,\"message\":\"Sesión expirada.\"}");
            return;
        }
```

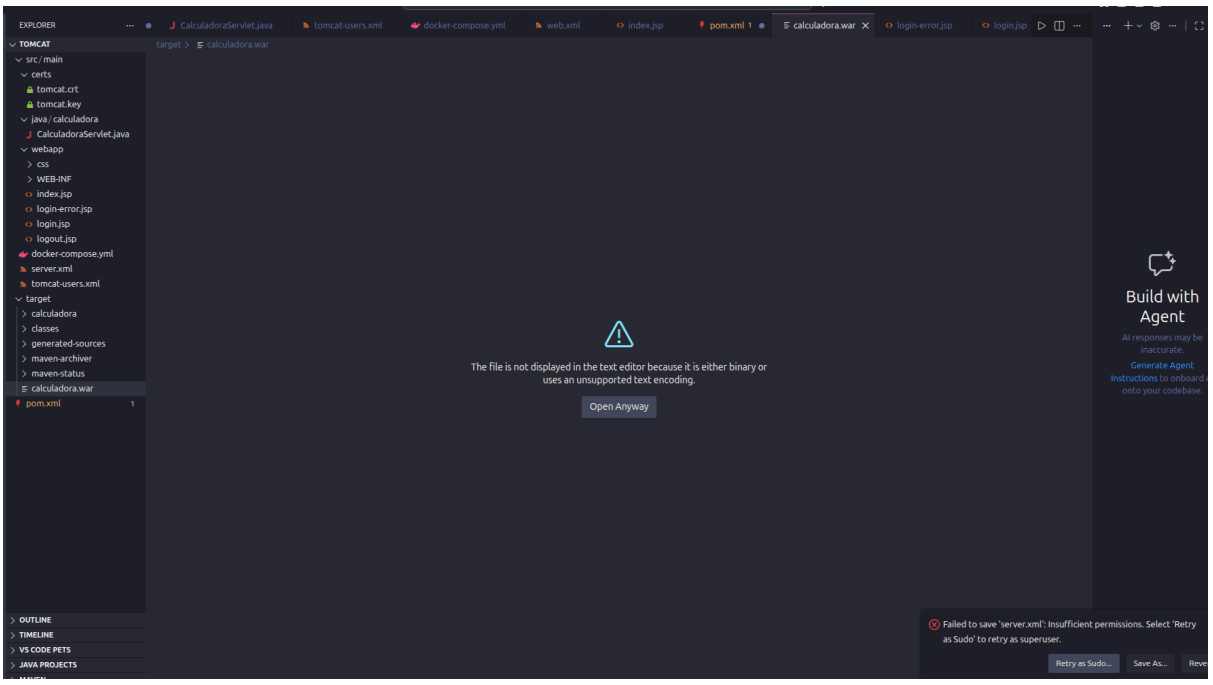


```jsp
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%
    boolean full   = request.isUserInRole("calc_full");
%>
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Calculadora JSP</title>
  <link rel="stylesheet" href="css/estilo.css">
</head>
<body>

<div class="contenedor">
  <!-- Cabecera con el título y el botón de cerrar sesión -->
  <div class="cabecera">
    <h2>Calculadora JSP</h2>
    <!--Formulario para cerrar sesión.Se envía por POST a logout.jsp-->
    <form method="post" action="<%=request.getContextPath()%>/logout.jsp">
      <button type="submit">Cerrar sesión</button>
    </form>
  </div>
  <!--Formulario principal de la calculadora.-->
  <form id="formCalc" autocomplete="off">
    <label>Número 1</label>
    <input type="number" name="num1" step="any" required>

    <label>Número 2</label>
    <input type="number" name="num2" step="any" required>

    <label>Operación</label>
    <select name="operacion" required>
      <option value="suma">Suma</option>
      <!--Solo mostramos la opción "resta" si el usuario tiene el rol calc_full-->
      <% if (full) { %>
        <option value="resta">Resta</option>
      <% } %>
    </select>

    <button type="submit">Calcular</button>
  </form>
  <!--resultado devuelto por el servlet-->
  <div id="salida" class="resultado" style="display:none;"></div>
</div>

<script>
  //referencias al formulario y al div del resultado
  const form = document.getElementById("formCalc");
  const salida = document.getElementById("salida");
```

Paso 8:

Se incorpora el archivo de configuración web.xml y se ejecuta el comando mvn clean package para compilar el proyecto y generar el archivoen la carpeta target/.


usua5pc17@A5PC17: ~/Escritorio/DAW/Despliegue/Tomcat
sisu-inject-bean/1.4.2/sisu-inject-bean-1.4.2.jar (153 kB at 316 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-classworlds/2.2.3/plexus-classworlds-2.2.3.jar (46 kB at 92 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/sisu/sisu-guice/2.1.7/sisu-guice-2.1.7-noaop.jar (472 kB at 914 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/plexus/plexus-sec-dispatcher/1.3/plexus-sec-dispatcher-1.3.jar (29 kB at 53 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/plexus/plexus-cipher/1.4/plexus-cipher-1.4.jar (13 kB at 24 kB/s)
[INFO] Packaging webapp
[INFO] Assembling webapp [calculadora] in [/home/usua5pc17/Escritorio/DAW/Despliegue/Tomcat/target/calculadora]
[INFO] Processing war project
[INFO] Copying webapp resources [/home/usua5pc17/Escritorio/DAW/Despliegue/Tomcat/src/main/webapp]
[INFO] Building war: /home/usua5pc17/Escritorio/DAW/Despliegue/Tomcat/target/calculadora.war
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 7.721 s
[INFO] Finished at: 2026-01-20T12:38:30+01:00
[INFO] --
usua5pc17@A5PC17:~/Escritorio/DAW/Despliegue/Tomcat$


EXPLORER
TOMCAT
src/main
certs
tomcat.crt
tomcat.key
java/calculadora
CalculadoraServlet.java
webapp
css
WEB-INF
index.jsp
login-error.jsp
login.jsp
logout.jsp
docker-compose.yml
server.xml
tomcat-users.xml
target
calculadora
classes
generated-sources
maven-archiver
maven-status
calculadora.war
pom.xml
target > calculadora.war
The file is not displayed in the text editor because it is either binary or uses an unsupported text encoding.
Open Anyway
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
OUTLINE
TIMELINE
VS CODE PETS
JAVA PROJECTS
Failed to save 'server.xml': Insufficient permissions. Select 'Retry as Sudo' to retry as superuser.
Retry as Sudo...
Save As...

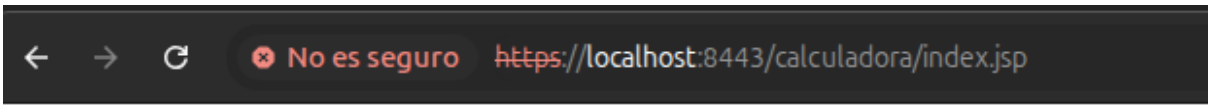
Paso 9:

Se configura el acceso mediante roles (calc_basic, calc_full) y usuarios (basic_user, full_user) en el archivo tomcat-users.xml, verificando la seguridad mediante las páginas de login, error y logout.

CalculadoraServlet.java | tomcat-users.xml | docker-compose.yml | web.xml | index.jsp | pom.xml 1

src > main > tomcat-users.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users>

<!-- Roles de la calculadora -->
<role rolename="calc_basic"/>
<role rolename="calc_full"/>

<!-- Usuario con calculadora básica -->
<user username="basic_user" password="123" roles="calc_basic"/>

<!-- Usuario con calculadora completa -->
<user username="full_user" password="321" roles="calc_full"/>

</tomcat-users>
```



Iniciar sesión

Usuario Contraseña Entrar

Usuarios de prueba:

- **basic_user** / 123
- **full_user** / 321

Iniciar sesión

Usuario Contraseña Entrar

Usuarios de prueba:

- **basic_user** / 123
- **full_user** / 321