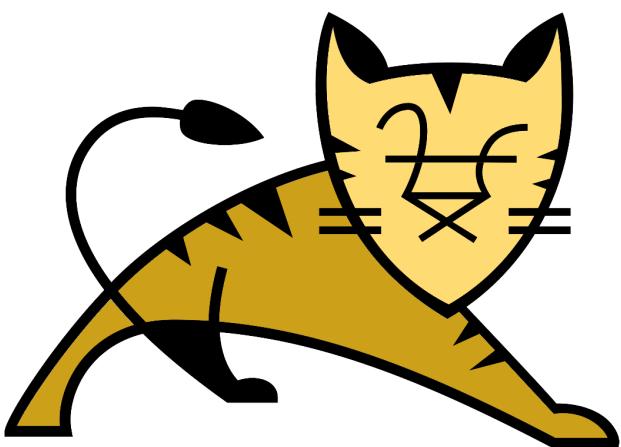


PRÁCTICA TEMA 3



Apache Tomcat

DANIEL AYBAR SOTO

PASO 0

```
DEPLIEGUE-ENTREGA-TOMCAT
  src\main
    certs
      tomcat.crt
      tomcat.key
    java\calculadora
      CalculadoraServlet.java
    webapp
      css
        estilo.css
    WEB-INF
      web.xml
      index.jsp
      login-error.jsp
      login.jsp
      logout.jsp
      docker-compose.yml
      server.xml
      tomcat-users.xml
      pom.xml
```

Para empezar, he creado los archivos según la estructura que aparece en la guía de la práctica.

Esto es sólo orientativo, ya que por ejemplo, en el siguiente punto tendré que reemplazar los certificados con unos generados automáticamente.

PASO 1

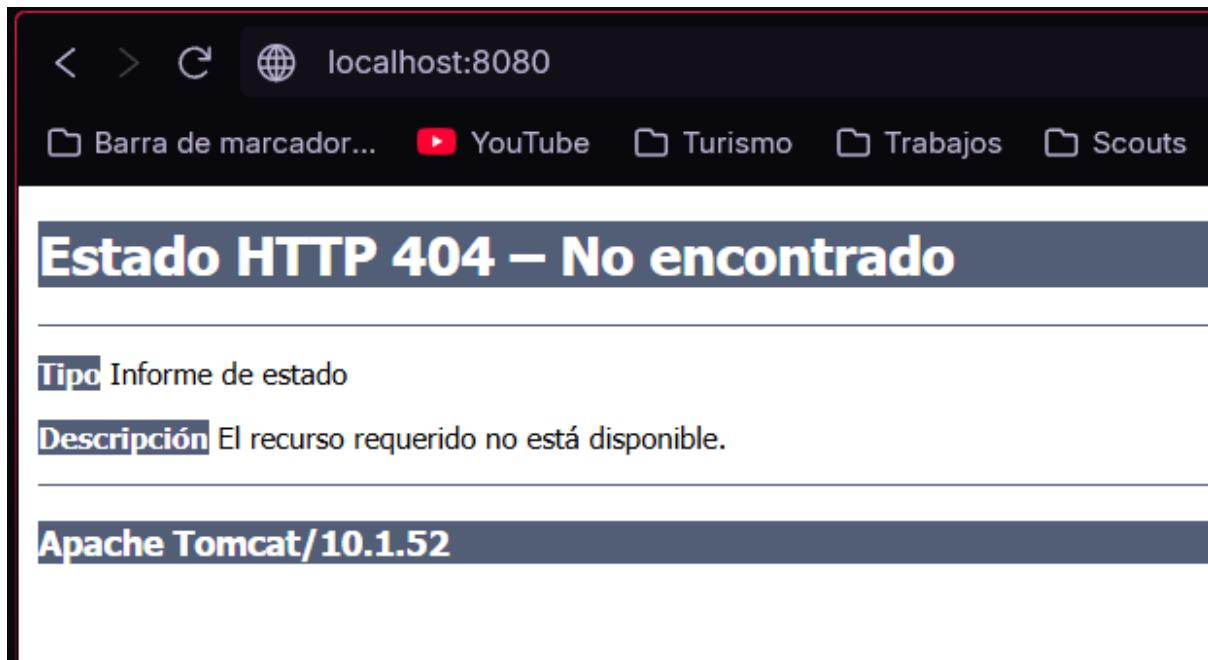
Crea tu archivo docker-compose y comprueba que tu contenedor se levanta con normalidad

```
version: "3.8"
services:
  tomcat:
    image: tomcat:10.1
    container_name: tomcat_t3
    ports:
      - "8080:8080"
      - "8443:8443"
    restart: unless-stopped
```

Este es el archivo de docker-compose que he creado por ahora. Compruebo que se levanta:

```
PS C:\Users\dantr\OneDrive\Escritorio\Despliegue-Entrega-Tomcat\src\main> docker compose up
time="2026-02-03T00:46:07+01:00" level=warning msg="C:\\Users\\dantr\\OneDrive\\Escritorio\\Despliegue-Entrega-Tomcat\\src\\main\\docker-compose.yml: the attribute `version` will be ignored, please remove it to avoid potential confusion"
[+] Running 9/8
  ✓ tomcat Pulled
    ✓ 4f4fb700ef54 Pull complete
    ✓ 8be760c2a9ab Pull complete
    ✓ bd41d65c3828 Pull complete
    ✓ a3629acs5bf4 Pull complete
    ✓ ae2fca99c299 Pull complete
    ✓ db072afe109d Pull complete
    ✓ eabd93b5d461 Pull complete
[+] Running 2/2
  ✓ Network main_default Created
  ✓ Container tomcat_t3 Created
Attaching to tomcat_t3
```

Trabajando en mi PC particular con SO Windows, al principio no me ha funcionado el comando “docker compose up -d”, pero eso se debía a que no tenía docker desktop abierto al mismo tiempo (jajan’t). Una vez lo he abierto, ya se ha levantado el contenedor sin problemas, mostrando la página por defecto de Apache.



PASO 2

Generar los certificados autofirmados SSL/TLS y modificar docker-compose en consecuencia. Comprueba que puedes entrar por “https”.

Con este comando, he creado los certificados SSL/TLS a través del contenedor de docker:

Se me han creado los certificados en la carpeta “src”, así que los he movido a la carpeta de “certs”.

Tal y como tengo los archivos, todavía no puedo acceder con el método https.

PASO 3

Instala Maven y comprueba que lo has hecho correctamente.

```
B Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnologia PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\dantr> scoop install main/maven
Updating Scoop...
Updating Buckets...
* 4d4003284d83 typos: Update to version 1.43.0           main      8 hours ago
* 349b54f68559 air: Update to version 1.64.5             main      8 hours ago
* ea71d1349da5 xray: Update to version 26.2.2            main     12 hours ago
* 0970daea3013a rumldl: Update to version 0.1.10          main     12 hours ago
* 175e6f6fe99 oxlint: Update to version 1.43.0            main     12 hours ago
* 93853898bd63 mise: Update to version 2026.2.1           main     12 hours ago
* 149b6ebc1771 meilisearch: Update to version 1.35.0        main     12 hours ago
* 84daf8bb9d15 localstack: Update to version 4.13.1         main     12 hours ago
* 9111c343e63e kyverno-cli: Update to version 1.17.0        main     12 hours ago
* d5c89n4h19d7 idtcls: Update to version 1.57.0-0-202602021128 main     12 hours ago
```

Compruebo que lo he instalado bien: todo perfe

```
PS C:\Users\dantr> mvn --version
Apache Maven 3.9.12 (848fb4bf2d427b72bdb2471c22fcfd7ebd9a7a1)
Maven home: C:\Users\dantr\scoop\apps\maven\current
Java version: 23, vendor: Oracle Corporation, runtime: D:
Default locale: es_ES, platform encoding: UTF-8
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
PS C:\Users\dantr> ■
```

PASO 4

Crea el archivo pom.xml y asegúrate de que concuerde con la versión de java que está usando maven.

The screenshot shows a code editor with two tabs: 'docker-compose.yml' and 'pom.xml'. The 'pom.xml' tab is active, displaying the XML configuration for the Maven build. The code includes annotations explaining the purpose of certain sections:

```
1  <!--
2   pom.xml = configuración de Maven.
3   Maven se encarga de:
4   - Descargar dependencias
5   - Compilar el código Java
6   - Empaquetar la aplicación como WAR para desplegar en Tomcat
7  -->
8  <project xmlns="http://maven.apache.org/POM/4.0.0">
9    <modelVersion>4.0.0</modelVersion>
10   <groupId>calculadora</groupId> <!-- Identidad del proyecto -->
11   <artifactId>calculadora</artifactId> <!-- artifactId: nombre del proyecto/artefacto -->
12   <version>1.0.0</version>
13   <packaging>war</packaging> <!-- packaging: indica que el resultado final será un WAR -->
14
15   <properties>
16     <!--
17      Versión de Java con la que compilamos.
18      Recomendación docente: usar 17 para evitar diferencias entre equipos.
19      -->
20     <maven.compiler.release>17</maven.compiler.release>
21     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
22   </properties>
23
24   <dependencies>
25     <!--
26       API de Servlets Jakarta (para Tomcat 10).
27       scope=provided significa:
28       - Se necesita para COMPIALAR
29       - NO se empaqueta dentro del WAR
30       - Porque Tomcat ya incluye esta librería en el servidor.
31     -->
32     <dependency>
33       <groupId>jakarta.servlet</groupId>
34       <artifactId>jakarta.servlet-api</artifactId>
35       <version>6.0.0</version>
36       <scope>provided</scope>
37     </dependency>
```

El uso de la etiqueta `<scope>provided</scope>` es una buena práctica. Significa que Maven usará la librería para compilar el código, pero no la meterá dentro del archivo final porque el servidor Tomcat ya la tiene instalada.

No debería darme ningún problema la compatibilidad de las versiones de Java que usan tanto Maven en mi equipo (Java 23) como el archivo pom.xml (Java 17), porque hay retrocompatibilidad y la etiqueta `maven.compiler.release` genera un archivo compatible con la versión 17.

PASO 5

Crea tu archivo server.xml, copiando el generado por Tomcat en tu proyecto.

En este paso no ha habido ningún problema.

```

15 |     <properties>
16 | PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
17 | 
18 | ● PS C:\Users\dantr\OneDrive\Escritorio\Despliegue-Entrega-Tomcat> cd src/main
19 | ● PS C:\Users\dantr\OneDrive\Escritorio\Despliegue-Entrega-Tomcat> docker cp tomcat_t3:/usr/local/tomcat/conf/server.xml ./server.xml
20 | successfully copied 8.7kB to C:\Users\dantr\OneDrive\Escritorio\Despliegue-Entrega-Tomcat\src\main\server.xml
21 | ○ PS C:\Users\dantr\OneDrive\Escritorio\Despliegue-Entrega-Tomcat\src\main>

```

Para que Tomcat use este nuevo archivo, lo incluyo en el docker-compose.yml:

```

2   ◇ Run All Services
3 services:
4     ◇ Run Service
5       tomcat:
6         image: tomcat:10.1
7         container_name: tomcat_t3
8         ports:
9           - "8080:8080"
10        restart: unless-stopped
11        volumes:
12          - ./certs:/usr/local/tomcat/certs
13          - ./server.xml:/usr/local/tomcat/conf/server.xml

```

PASO 6

Modifica el server.xml para indicar donde encontrará el contenedor los certificados.

Descomento el bloque y cambio las rutas de los certificados designados para que Tomcat los busque:

```

... docker-compose.yml pom.xml server.xml
src > main > server.xml
22   <Server port="8005" shutdown="SHUTDOWN">
23     <Service name="Catalina">
24       <Connector port="8080" protocol="org.apache.coyote.http11.Http11NioProtocol"
25         maxThreads="150" SSLEnabled="true"
26         maxParameterCount="1000"
27         >
28           <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
29           <SSLHostConfig>
30             <Certificate certificateFile="/usr/local/tomcat/certs/tomcat.crt"
31               certificateKeyFile="/usr/local/tomcat/certs/tomcat.key"
32               type="RSA" />
33           </SSLHostConfig>
34       </Connector>
35       <!-- Define an AJP 1.3 Connector on port 8009 -->
36       <!--
37         <Connector protocol="AJP/1.3"
38           port="8009" redirectPort="8443" />
39       -->
40     </Service>
41   </Server>
42   <!-- Define a globalNamingFactory for JNDI lookups -->
43   <GlobalNamingFactory factoryName="tomcat" type="org.apache.naming.java.javaURLContextFactory" />
44   <GlobalNamingFactory factoryName="java" type="org.apache.naming.java.javaURLContextFactory" />
45   <GlobalNamingFactory factoryName="jndi" type="org.apache.naming.java.javaURLContextFactory" />
46   <GlobalNamingFactory factoryName="jmx" type="org.apache.naming.java.javaJMXContextFactory" />
47   <GlobalNamingFactory factoryName="jmxremote" type="org.apache.naming.java.javaJMXContextFactory" />
48   <GlobalNamingFactory factoryName="jmxremotessl" type="org.apache.naming.java.javaJMXContextFactory" />
49   <GlobalNamingFactory factoryName="jmxrmi" type="org.apache.naming.java.javaJMXContextFactory" />
50   <GlobalNamingFactory factoryName="jmxrmissl" type="org.apache.naming.java.javaJMXContextFactory" />
51   <GlobalNamingFactory factoryName="jndirmi" type="org.apache.naming.java.javaRMIContextFactory" />
52   <GlobalNamingFactory factoryName="jndirmissl" type="org.apache.naming.java.javaRMIContextFactory" />
53   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
54   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
55   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
56   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
57   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
58   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
59   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
60   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
61   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
62   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
63   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
64   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
65   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
66   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
67   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
68   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
69   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
70   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
71   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
72   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
73   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
74   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
75   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
76   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
77   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
78   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
79   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
80   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
81   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
82   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
83   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
84   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
85   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
86   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
87   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
88   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
89   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
90   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
91   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
92   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
93   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
94   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
95   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
96   <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
97   <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
98   <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
99   <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
100  <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
101  <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
102  <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />
103  <GlobalNamingFactory factoryName="jndih2" type="org.apache.naming.java.javaH2ContextFactory" />
104  <GlobalNamingFactory factoryName="jndih2ssl" type="org.apache.naming.java.javaH2ContextFactory" />
105  <GlobalNamingFactory factoryName="jndih2rmissl" type="org.apache.naming.java.javaH2RMIContextFactory" />
106  <GlobalNamingFactory factoryName="jndih2rmi" type="org.apache.naming.java.javaH2RMIContextFactory" />

```

Tumbo el contenedor y lo vuelvo a levantar para comprobar que los cambios sean correctos. Aquí he tenido problemas para comprobar que se estableciera conexión, pero después de tumbar otra vez el contenedor, parece que ha funcionado. Usando el comando “docker logs tomcat_t3”, veo que se ha inicializado el manejador de protocolos de comunicación segura:

```
[main] org.apache.catalina.core.AprLifecycleListener.initializeSSL OpenSSL successfully initialized [OpenSSL 3.0.13 30 Jan 2024]
[main] org.apache.coyote.AbstractProtocol.init Initializing ProtocolHandler ["http-nio-8080"]
[main] org.apache.coyote.http11.AbstractHttp11Protocol.configureUpgradeProtocol The ["https-openssl-nio-8443"] connector has been
  ALPN
[main] org.apache.coyote.AbstractProtocol.init Initializing ProtocolHandler ["https-openssl-nio-8443"]
[main] org.apache.tomcat.util.net.AbstractEndpoint.logCertificate Connector [https-openssl-nio-8443], TLS virtual host [_default_]
[main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent Loaded Apache Tomcat Native library [2.0.12] using APR version 1.6.1
[main] org.apache.catalina.core.AprLifecycleListener.initializeSSL OpenSSL successfully initialized [OpenSSL 3.0.13 30 Jan 2024]
[main] org.apache.coyote.AbstractProtocol.init Initializing ProtocolHandler ["http-nio-8080"]
```

PASO 7

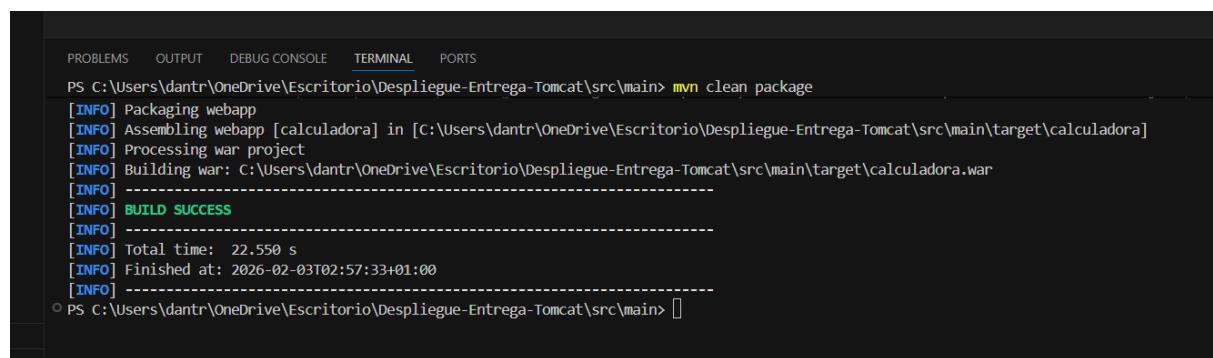
Crea tu aplicación calculadora en dos pasos, un archivo java que contendrá la lógica y un index.jsp que será tu interfaz.

Voy a probar primero con los archivos orientativos que aparecen en la guía de la práctica para ver si funcionan.

PASO 8

Añade tu archivo web.xml e intenta probar a generar el archivo.war con maven.

Al parecer, en un momento había descargado el daemon de Maven y había referenciado su carpeta bin en las variables de entorno, así que he tenido que descargar el binario y cambiar el path. Después de esto y cerrar y volver a abrir todas las terminales, ya me funciona el comando mvn clean package:



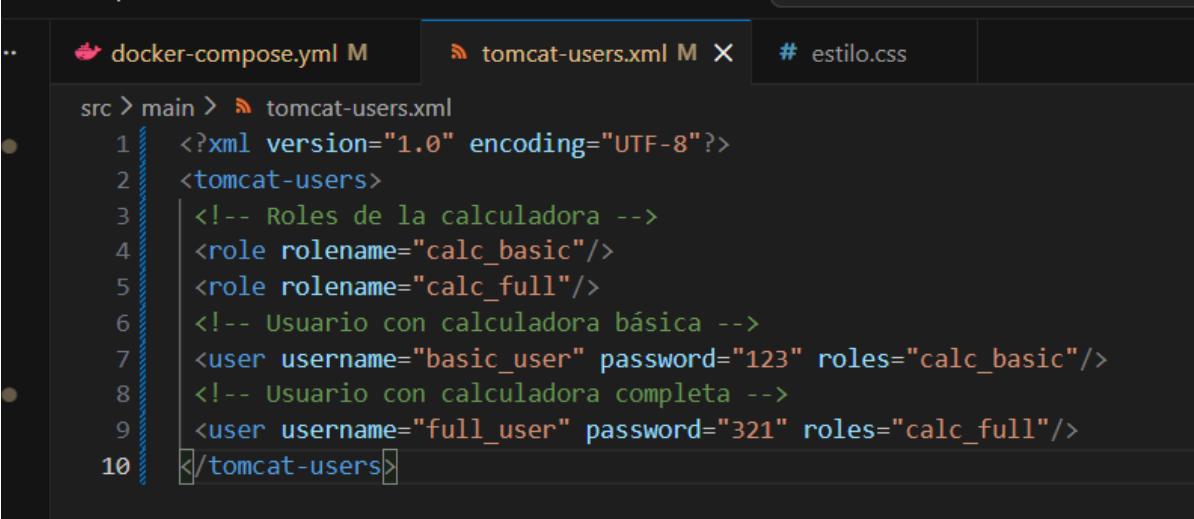
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\dantr\OneDrive\Escritorio\Despliegue-Entrega-Tomcat\src\main> mvn clean package
[INFO] Packaging webapp
[INFO] Assembling webapp [calculadora] in [C:\Users\dantr\OneDrive\Escritorio\Despliegue-Entrega-Tomcat\src\main\target\calculadora]
[INFO] Processing war project
[INFO] Building war: C:\Users\dantr\OneDrive\Escritorio\Despliegue-Entrega-Tomcat\src\main\target\calculadora.war
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 22.550 s
[INFO] Finished at: 2026-02-03T02:57:33+01:00
[INFO] -----
```

Con esto, se me ha creado una carpeta “target” con varios archivos dentro, “calculadora.war” entre ellos. Ahora tengo que actualizar el docker-compose.yml, tirar el contenedor de docker y volver a levantarla para poder ver el archivo generado en el navegador en la url “<https://localhost:8443/calculadora/>”.

PASO 9

Añade roles a tu app con el archivo tomcat-users.xml y comprobar su correcto funcionamiento, tanto de los usuarios con sus contraseñas como con las restricciones impuestas a los roles.

Edito tomcat-users.xml para que tenga los datos dados en la guía:



```
src > main > tomcat-users.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <tomcat-users>
3  <!-- Roles de la calculadora -->
4  <role rolename="calc_basic"/>
5  <role rolename="calc_full"/>
6  <!-- Usuario con calculadora básica -->
7  <user username="basic_user" password="123" roles="calc_basic"/>
8  <!-- Usuario con calculadora completa -->
9  <user username="full_user" password="321" roles="calc_full"/>
10 </tomcat-users>
```

Y entonces, edito el docker-compose.yml para que tomcat busque a los usuarios en la dirección de tomcat-users.xml:



```
M 10  restart: unless-stopped
  11  volumes:
  12    - ./certs:/usr/local/tomcat/certs
  13    - ./server.xml:/usr/local/tomcat/conf/server.xml
  14    - ./tomcat-users.xml:/usr/local/tomcat/conf/tomcat-users.xml
  15    - ./target/calculadora.war:/usr/local/tomcat/webapps/calculadora.war
```

Sigo sin poder acceder a ninguna vista de los archivos .js que tengo. He intentado de todo: docker compose down/up -d, comprobar que existe el archivo calculadora.war en la carpeta target, revisar los volumes del archivo docker-compose.yml....y nada.