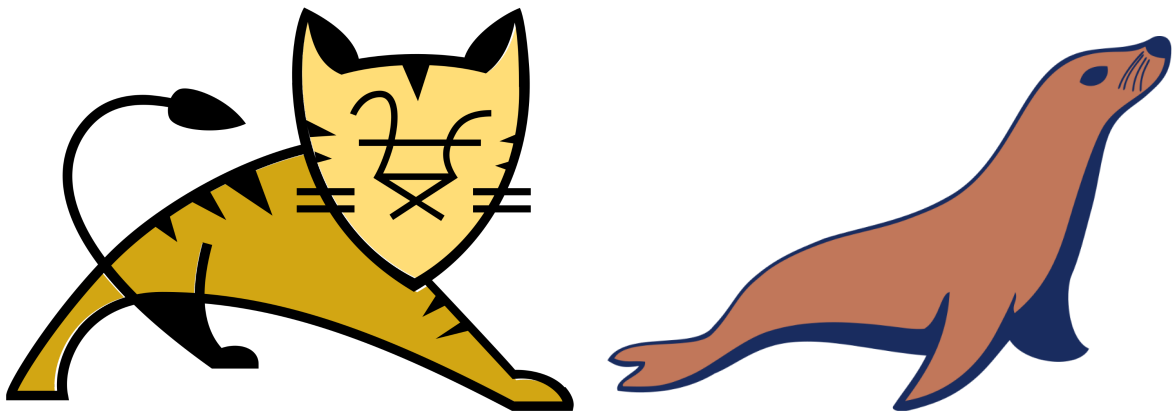


Memoria de prácticas de CRUD



DESPLIEGUE DE APLICACIONES WEB

SANDRA RUIZ JIMÉNEZ
2º DAW

Lista de contenidos

1. INTRODUCCIÓN	1
2. CREACIÓN DE LOS CONTENEDORES	1
2.1. CREANDO EL DOCKER-COMPOSE	1
3. CONFIGURACIÓN DE TOMCAT	3
3. CONFIGURACIÓN DE LA BASE DE DATOS EN MARIADB	6
4. CONFIGURACIÓN DE LA APLICACIÓN	7
5. DESPLIEGUE DE APLICACIÓN	12
6. BIBLIOGRAFÍA	14

1. INTRODUCCIÓN

El objetivo de esta memoria es el despliegue de una aplicación con acceso a base de datos en un contenedor Apache Tomcat.

2. CREACIÓN DE LOS CONTENEDORES

Como nuestra aplicación necesita tener acceso a base de datos, necesitamos crear dos contenedores, uno para Tomcat, el cual va a contener a nuestra aplicación, y otro para la base de datos.

Para que ambos contenedores puedan interactuar entre ellos, es preciso que ambos se encuentren en la misma red. Por este motivo, vamos a crear los contenedores a través de un fichero docker-compose, que automáticamente los añade a la misma red y así no tendremos dificultades.

2.1. CREANDO EL DOCKER-COMPOSE

```
1 version: "3.7" #versión a actualizar
2
3 services:
4
5   tomcat: #servicio que se va a desplegar al ejecutar el compose
6     container_name: tomcatCRUD #nombre que le voy a asignar al contenedor que creará el compose
7     image: tomcat:10.0 #imagen de tomcat que voy a usar para el contenedor
8     ports:
9       - 8887:8080 #mapeo de puertos, del 8080 del contenedor al 8887 de mi localhost
10    volumes:
11      - ./context.xml:/usr/local/tomcat/conf/context.xml #archivo context.xml configurado
12      - ./tomcat-users.xml:/usr/local/tomcat/conf/tomcat-users.xml #archivo tomcat-users.xml configurado
13
14   mariadb: #servicio que se va a desplegar al ejecutar el compose
15     container_name: bdtomcatCRUD
16     image: mariadb #imagen de mariaDB que vamos a utilizar
17     ports:
18       - 3306:3306
19     environment: #Se establecen las distintas variables de entorno que se usarán para configurar el servicio
20       MYSQL_ROOT_PASSWORD: '1234' #contraseña de la base de datos
21     volumes: #declaramos la ruta del volumen que se va a montar
22       - '/db:/var/lib/mariadb'
23
```

IMPORTANTE: debemos usar la versión 10.0 de Tomcat porque la 9.0 da problemas.

Declaramos los dos servicios que utilizaremos, tomcat y mariadb.

Para el servicio de tomcat necesitamos incorporar en el mismo directorio del docker-compose.yml los ficheros **context.xml** y **tomcat-users.xml**, que tenemos que configurar como en la práctica anterior para que tengan acceso al manager de tomcat.

```
sandra@sandra-UX410UAK: ~/Downloads/4. Despliegue de a...
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/TomcatMariaDB$ ls
context.xml  docker-compose.yml  tomcat-users.xml
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/TomcatMariaDB$
```

El fichero **context.xml** queda así:

```
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/TomcatMariaDB$ cat context.xml
<?xml version="1.0" encoding="UTF-8"?>
<Context privileged="true" antiResourceLocking="false" docBase="${catalina.home}/webapps/manager">
    <Valve className="org.apache.catalina.valves.RemoteAddrValve" allow="^.*$"/>
</Context>
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/TomcatMariaDB$
```

Y el fichero **tomcat-users.xml**, así:

```
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/TomcatMariaDB$ cat tomcat-users.xml
<?xml version="1.0" encoding="UTF-8"?>

<tomcat-users xmlns="http://tomcat.apache.org/xml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
               version="1.0">
    <role rolename="manager-gui"/>
    <user username="tomcat" password="tomcat" roles="manager-gui"/>
</tomcat-users>
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/TomcatMariaDB$
```

Abrimos una terminal desde el directorio que contiene el compose y ejecutamos el comando **docker-compose up**

```
sandra@sandra-UX410UAK: ~/Downloads/4. Despliegue de aplicac...
sandra@sandra-UX410UAK:~/Downloads/4. Despliegue de aplicaciones web/TomcatMariaDB$
docker-compose up
Creating network "tomcatmariadb_default" with the default driver
Creating bdtomcatCRUD ... done

Creating tomcatCRUD ... done

Attaching to tomcatCRUD, bdtomcatCRUD
tomcatCRUD | NOTE: Picked up JDK_JAVA_OPTIONS: --add-opens=java.base/java.lang=ALL-
UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.util=AL
L-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rm
i/sun.rmi.transport=ALL-UNNAMED
bdtomcatCRUD | 2023-01-24 19:11:51+00:00 [Note] [Entrypoint]: Entrypoint script for
MariaDB Server 1:10.10.2+maria~ubu2204 started.
bdtomcatCRUD | 2023-01-24 19:11:51+00:00 [Note] [Entrypoint]: Switching to dedicated
user 'mysql'
bdtomcatCRUD | 2023-01-24 19:11:51+00:00 [Note] [Entrypoint]: Entrypoint script for
MariaDB Server 1:10.10.2+maria~ubu2204 started.
bdtomcatCRUD | 2023-01-24 19:11:51+00:00 [Note] [Entrypoint]: Initializing database
files
tomcatCRUD | 24-Jan-2023 19:11:52.039 INFO [main] org.apache.catalina.startup.Versio
nLoggerListener.log Server version name: Apache Tomcat/9.0.70
tomcatCRUD | 24-Jan-2023 19:11:52.044 INFO [main] org.apache.catalina.startup.Versio
nLoggerListener.log Server built: Dec 1 2022 14:05:47 UTC
```

En otro terminal, con el comando **docker ps**, comprobamos que ambos contenedores se han creado:

```
sandra@sandra-UX410UAK: ~
sandra@sandra-UX410UAK:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
e4946ac085bf   tomcat:9.0 "catalina.sh run"        42 seconds ago Up 41 seconds 0.0.0.0:8887->8080/tcp, :::8887->8080/tcp tomcatCRUD
536e7b695fcf   mariadb    "docker-entrypoint.s..." 42 seconds ago Up 40 seconds  0.0.0.0:3306->3306/tcp, :::3306->3306/tcp bdtomcatCRUD
```

3. CONFIGURACIÓN DE TOMCAT

Para que nos aparezca la página de Tomcat debemos comprobar que el directorio webapps (directorio que contiene las aplicaciones web) contiene el host-manager, el manager y el directorio docs.

Sin embargo, estos ficheros se encuentran dentro de webapps.dist por lo tenemos que copiarlos en el directorio webapps.

Entramos dentro de su sistema de ficheros:

docker exec -it tomcatCRUD /bin/bash

A continuación comprobamos que están dentro del directorio webapps.dist

ls webapps.dist/

Como están, para copiarlos, ejecutamos el siguiente comando:

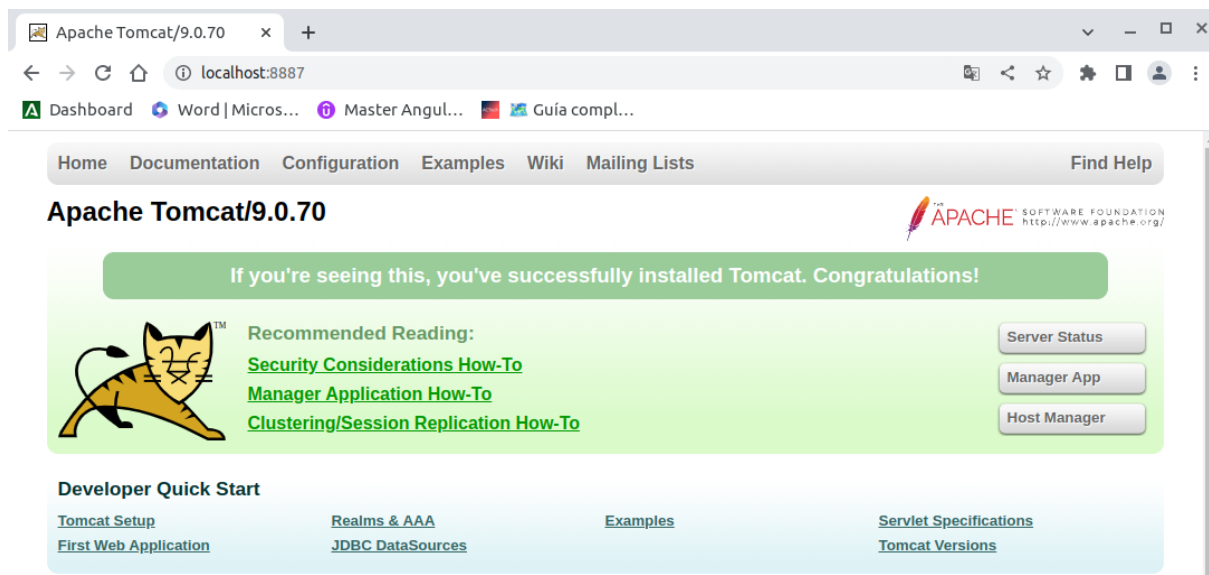
cp -R webapps.dist/* webapps

Comprobamos que se han copiado dentro del directorio webapps:

ls webapps

```
sandra@sandra-UX410UAK:~$ docker start tomcatCRUD
tomcatCRUD
sandra@sandra-UX410UAK:~$ docker exec -it tomcatCRUD /bin/bash
root@e4946ac005bf:/usr/local/tomcat# ls webapps.dist/
docs  examples  host-manager  manager  ROOT
root@e4946ac005bf:/usr/local/tomcat# cp -R webapps.dist/* webapps
root@e4946ac005bf:/usr/local/tomcat# ls webapps
docs  examples  host-manager  manager  ROOT
root@e4946ac005bf:/usr/local/tomcat#
```

Después de esto, nos debería aparecer la página de inicio de Tomcat en el navegador, escribiendo localhost:8887:



Sin embargo, aunque hemos configurado el manager para entrar y tener permiso, aún no nos lo permite. Esto se debe a que Tomcat no está utilizando el fichero context.xml que hemos modificado, sino el fichero context.xml que se encuentra en la ruta /usr/local/tomcat/webapps/manager/META-INF.

Para solucionarlo, tan solo tenemos que borrar este archivo context.xml:

rm context.xml

```
root@e4946ac005bf:/usr/local/tomcat# cd webapps/manager/META-INF
root@e4946ac005bf:/usr/local/tomcat/webapps/manager/META-INF# ls
context.xml
root@e4946ac005bf:/usr/local/tomcat/webapps/manager/META-INF# rm context.xml
root@e4946ac005bf:/usr/local/tomcat/webapps/manager/META-INF# ls
root@e4946ac005bf:/usr/local/tomcat/webapps/manager/META-INF#
```

Ahora reiniciamos Tomcat para que se apliquen los cambios. Para reiniciar, debemos ir al directorio /usr/local/tomcat/bin y ejecutar ./startup.sh.



```
root@e4946ac005bf:/usr/local/tomcat/bin# ls
bootstrap.jar          configtest.sh         startup.sh
catalina.sh            daemon.sh             tomcat-juli.jar
catalina-tasks.xml    digest.sh             tool-wrapper.sh
ciphers.sh            makebase.sh          version.sh
commons-daemon.jar    setclasspath.sh
commons-daemon-native.tar.gz shutdown.sh
root@e4946ac005bf:/usr/local/tomcat/bin# ./startup.sh
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /opt/java/openjdk
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
root@e4946ac005bf:/usr/local/tomcat/bin#
```

Si ahora intentamos entrar al manager de nuevo, usando usuario y contraseña tomcat, ya debemos tener acceso al manager.

/manager

localhost:8887/manager/html

Dashboard Word | Micros... Master Angul... Guía compl...

Tomcat Web Application Manager

Message: OK

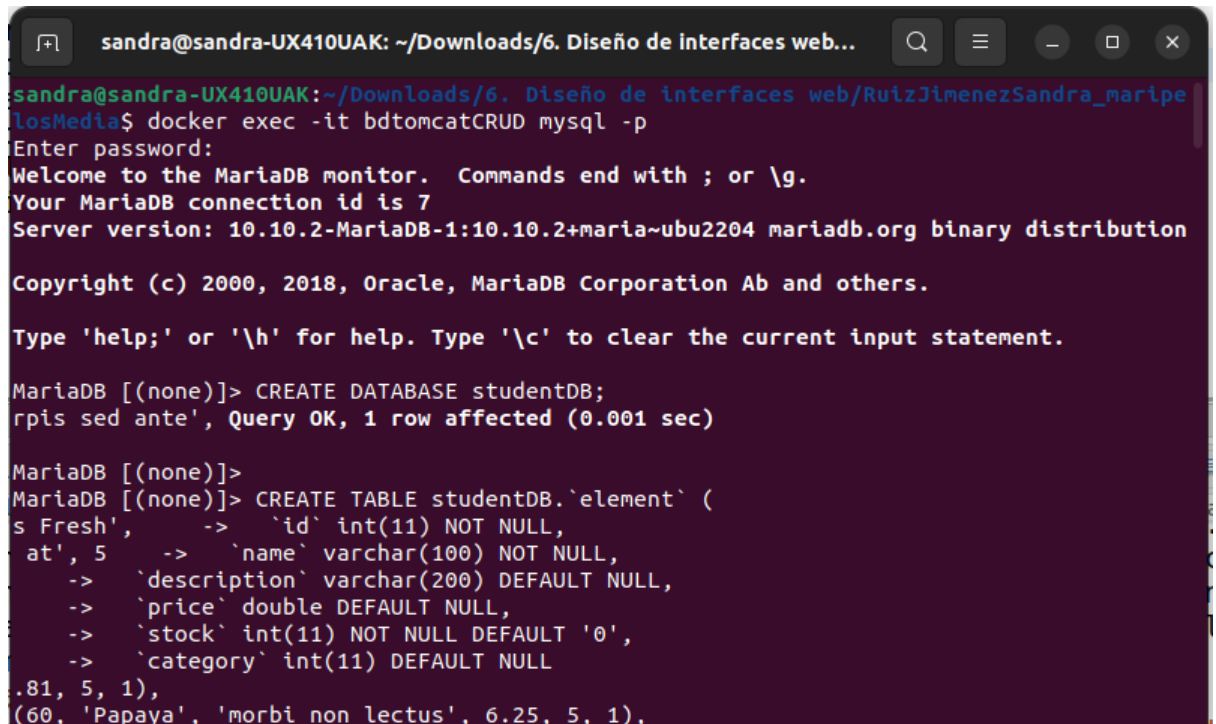
Manager

[List Applications](#)
[HTML Manager Help](#)
[Manager Help](#)
[Server Status](#)

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/docs	None specified	Tomcat Documentation	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/examples	None specified	Servlet and JSP Examples	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/host-manager	None specified	Tomcat Host Manager Application	true	0	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>
/manager	None specified	Tomcat Manager Application	true	1	<div>Start Stop Reload Undeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>

3. CONFIGURACIÓN DE LA BASE DE DATOS EN MARIADB



```
sandra@sandra-UX410UAK: ~/Downloads/6. Diseño de interfaces web...
sandra@sandra-UX410UAK:~/Downloads/6. Diseño de interfaces web/RuizJimenezSandra_marip...
losMedia$ docker exec -it bdtomcatCRUD mysql -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 7
Server version: 10.10.2-MariaDB-1:10.10.2+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE studentDB;
rps sed ante', Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]>
MariaDB [(none)]> CREATE TABLE studentDB.`element` (
s Fresh',      ->  `id` int(11) NOT NULL,
at', 5        ->  `name` varchar(100) NOT NULL,
              ->  `description` varchar(200) DEFAULT NULL,
              ->  `price` double DEFAULT NULL,
              ->  `stock` int(11) NOT NULL DEFAULT '0',
              ->  `category` int(11) DEFAULT NULL
.81, 5, 1),
(60, 'Papaya', 'morbi non lectus', 6.25, 5, 1),
```

¡IMPORTANTE! Recuerda que al ser un proyecto con base de datos, antes de ejecutarlo, el contenedor de la base de datos debe estar levantado y que ésta debe contener un esquema con el nombre que se indicó en el application.properties. De no ser así, al hacer el Run As, la aplicación fallará.

4. CONFIGURACIÓN DE LA APLICACIÓN

Vamos a desplegar una aplicación de Spring con acceso a base de datos.

Debemos tener en cuenta algunos aspectos previos para que nuestra aplicación funcione en su despliegue:

1. El fichero Application (que encontramos en src/main/java) de nuestra aplicación debe extender de ServletInitializer.

```

SpringProjectToWorkApplication.java x
1 package com.jacaranda.springProjecToWork;
2
3 import org.springframework.boot.SpringApplication;
4
5
6
7 @SpringBootApplication
8 public class SpringProjectToWorkApplication extends SpringBootServletInitializer {
9
10     public static void main(String[] args) {
11         SpringApplication.run(SpringProjectToWorkApplication.class, args);
12     }
13
14 }
15

```

2. En el `application.properties` (en `src/main/resources`) debemos hacer algunas configuraciones tales como indicar la dirección IP del contenedor de MariaDB que hemos creado previamente. Esto es importante para que nuestra aplicación desde el contenedor de Tomcat se pueda comunicar con la base de datos. De una manera muy sencilla, a través del comando `docker inspect nombreDelContenedor` podremos averiguar la IP del contenedor de MariaDB. Si nos fijamos después del puerto 3306 aparece `/mariaBDtomcat`.

Este es el nombre del esquema que hemos creado en nuestra base de datos en el que cargaremos la información que queremos almacenar.

`pom.xml`

```

41 <dependency>
42     <groupId>org.mariadb.jdbc</groupId>
43     <artifactId>mariadb-java-client</artifactId>
44     <version>2.5.2</version>
45     <scope>runtime</scope>
46 </dependency>

```

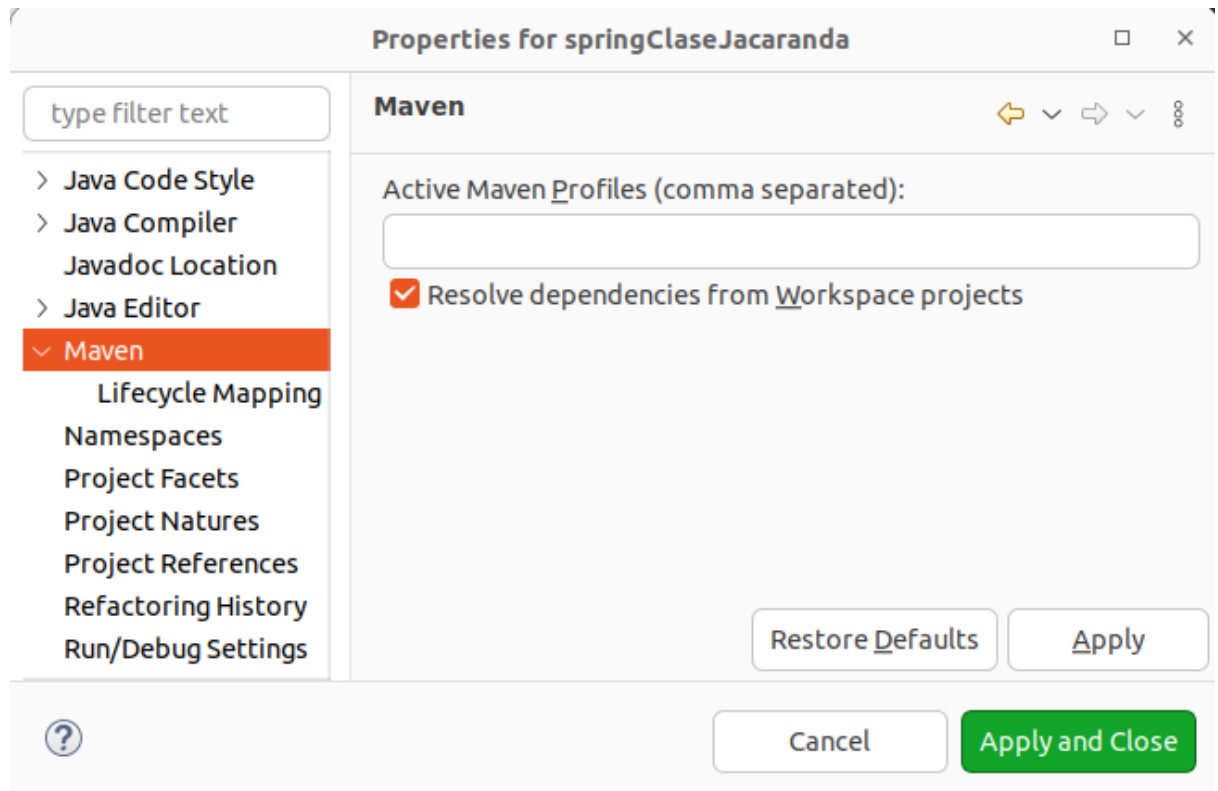
```

springClaseJacaran  application.properties x  springClaseJacaran  studentDB.sql  ElementController.java  ElementService.java
1 server.port=8199
2 spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
3 spring.datasource.url=jdbc:mariadb://172.23.0.3:3306/studentDB?useSSL=false
4 spring.datasource.username=root
5 spring.datasource.password=1234
6 spring.jpa.hibernate.ddl-auto=create-drop
7 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
8

```

En la línea 3, se llama `bdtomcatCRUD` porque es el nombre del contenedor de Docker.

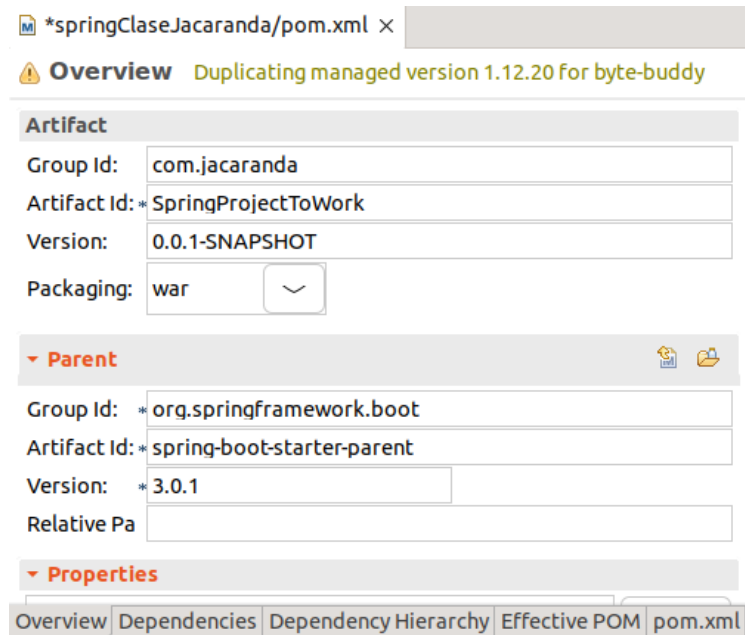
3. En la configuración de Maven, tendremos que dejar la configuración del perfil activo en blanco, la cual por defecto nos especifica que es `pom.xml`. Así que lo borramos.



4. Comprobamos que las rutas de las vistas de nuestra aplicación están correctamente nombradas. Es decir, revisamos las etiquetas href de los html, así como los href que introducen las hojas de estilo:

Una vez realizados los pasos anteriores, ya solo nos queda conseguir nuestro fichero .war que es el que contendrá a nuestra aplicación.

Para conseguirlo, primero vamos al fichero de configuración pom.xml del proyecto y en la pestaña Overview cambiamos el packaging por defecto jar por war:



Guardamos los cambios y ejecutamos nuestra aplicación a través de Run As → Maven Install.


Una vez terminado, obtendremos un mensaje de éxito si todo ha ido correctamente:





























```

[INFO] --- spring-boot-maven-plugin:3.0.1:repackage (repackage) @ SpringProjectToWork ---
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-bu
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-bui
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-loa
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-loa
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-bu
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-loa
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-bui
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-loa
[INFO] Replacing main artifact with repackaged archive
[INFO] --- maven-install-plugin:3.0.1:install (default-install) @ SpringProjectToWork ---
[INFO] Installing /home/sandra/entornoservidorspring-workspace/springClaseJacaranda/pom.xml to /home/sa
[INFO] Installing /home/sandra/entornoservidorspring-workspace/springClaseJacaranda/target/SpringProje
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 10.852 s
[INFO] Finished at: 2023-01-25T12:31:30+01:00
[INFO] -----

```

Nuestro archivo .war ya se ha generado y podemos encontrarlo en el directorio /target del proyecto:

✓  > springClaseJacaranda [boot] [springClaseJacaranda main]

- ✓  > src/main/java
 - >  > com.jacaranda.springProjectToWork
 - ✓  > com.jacaranda.springProjectToWork.controller
 - >  > ElementController.java
 - >  > com.jacaranda.springProjectToWork.model
 - >  > com.jacaranda.springProjectToWork.repository
 - ✓  > com.jacaranda.springProjectToWork.service
 - >  > ElementService.java
- >  > src/main/resources
- >  src/test/java
- >  JRE System Library [jre]
- >  Maven Dependencies
- ✓  > script
 - >  > studentDB.sql
- >  > src
- ✓  target
 - >  generated-sources
 - >  generated-test-sources
 - >  m2e-wtp
 - >  maven-archiver
 - >  maven-status
 - >  SpringProjectToWork-0.0.1-SNAPSHOT
 - >  surefire-reports
 - >  SpringProjectToWork-0.0.1-SNAPSHOT.war
 - >  SpringProjectToWork-0.0.1-SNAPSHOT.war.original
- >  mvnw
- >  mvnw.cmd
- >  > pom.xml

5. DESPLIEGUE DE APLICACIÓN

En el manager de Tomcat debemos dirigirnos a la parte de Desplegar, a la opción de Archivo WAR a desplegar y en Examinar... seleccionamos el archivo .WAR de nuestra aplicación.

Deploy

Deploy directory or WAR file located on server

Context Path:

Version (for parallel deployment):

XML Configuration file path:

WAR or Directory path:

Deploy

WAR file to deploy

Select WAR file to upload No file chosen


Deploy

Una vez desplegado nos debe aparecer en la parte superior de la página, en Aplicaciones:

/manager

localhost:8888/manager/html/upload?org.apache.catalina.filters.CSRF_NONCE=935131B3F0211752A3A312A0C9BF1ACC

Dashboard Word | Micros... Master Angul... Guía compl... ¿Cómo imple... English Spanish



¿Cómo implementar Lazy Loading en Angular? | by Henry Dev | Medium
https://medium.com/@HenryGBC/cómo-implementar-lazy-loading-en-angular-74b6e85d0211

ACME SOFTWARE FOUNDATION

Tomcat Web Application Manager

Message: OK

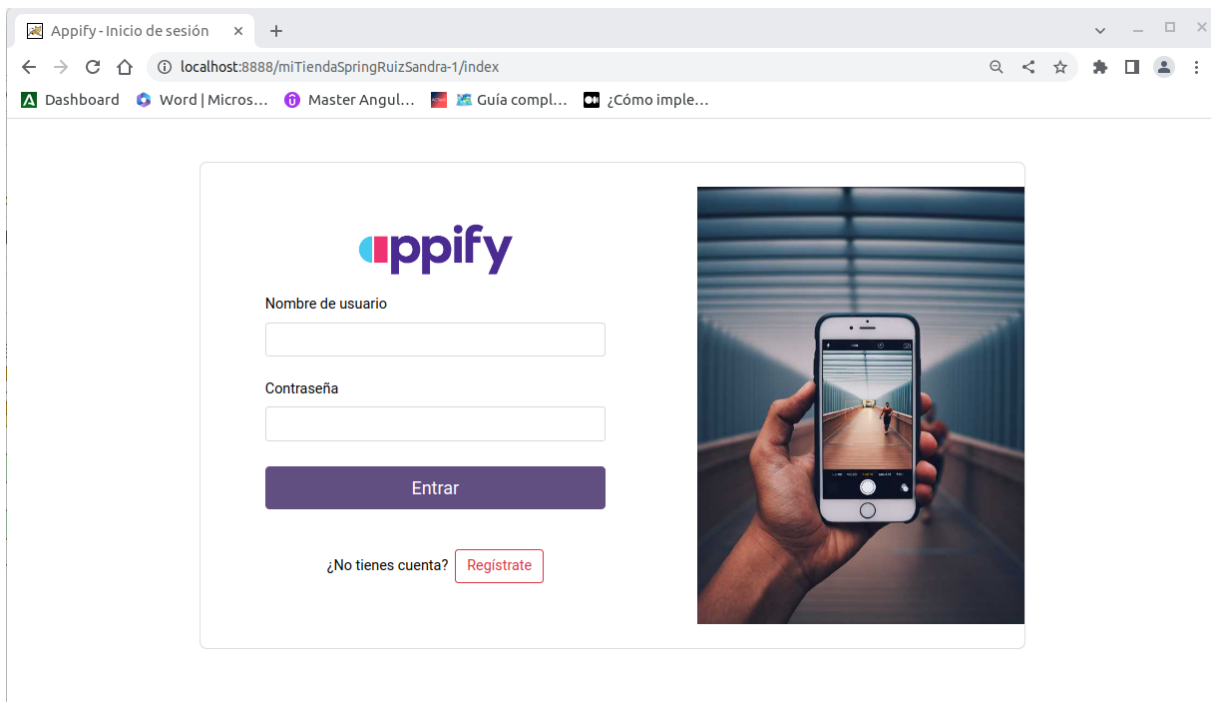
Manager

List ApplicationsHTML Manager HelpManager HelpServer Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/miTiendaSpringRuizSandra-1	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Si hacemos click en el enlace que se proporciona, se abrirá en una ventana del navegador nuestra aplicación:



6. BIBLIOGRAFÍA

García, Roberto (s.f.). *Tema 5. Administración de servidores de aplicaciones (Tomcat)*. Disponible en: <https://sites.google.com/site/desplieguedeaplicacionesweb/tema-5-administracion-de-servidores-de-aplicaciones-tomcat>. Consultado: 25/01/2023

Microsoft (s.f.). *Tutorial: Creación de aplicaciones de varios contenedores con MySQL y Docker Compose*. Disponible en: <https://learn.microsoft.com/es-es/visualstudio/docker/tutorials/tutorial-multi-container-app-mysql>. Consultado: 25/01/2023

programador clic (s.f.). *Error al iniciar el proyecto de arranque Spring: java.lang.IllegalArgumentException: LoggerFactory no es un Logback*. Disponible en: <https://programmerclick.com/article/4892741451/>. Consultado: 26/01/2023

programador clic (s.f.). *Obtenga la ruta de referencia css y js en SpringMVC*. Disponible en: <https://programmerclick.com/article/2068802451/>. Consultado: 26/01/2023