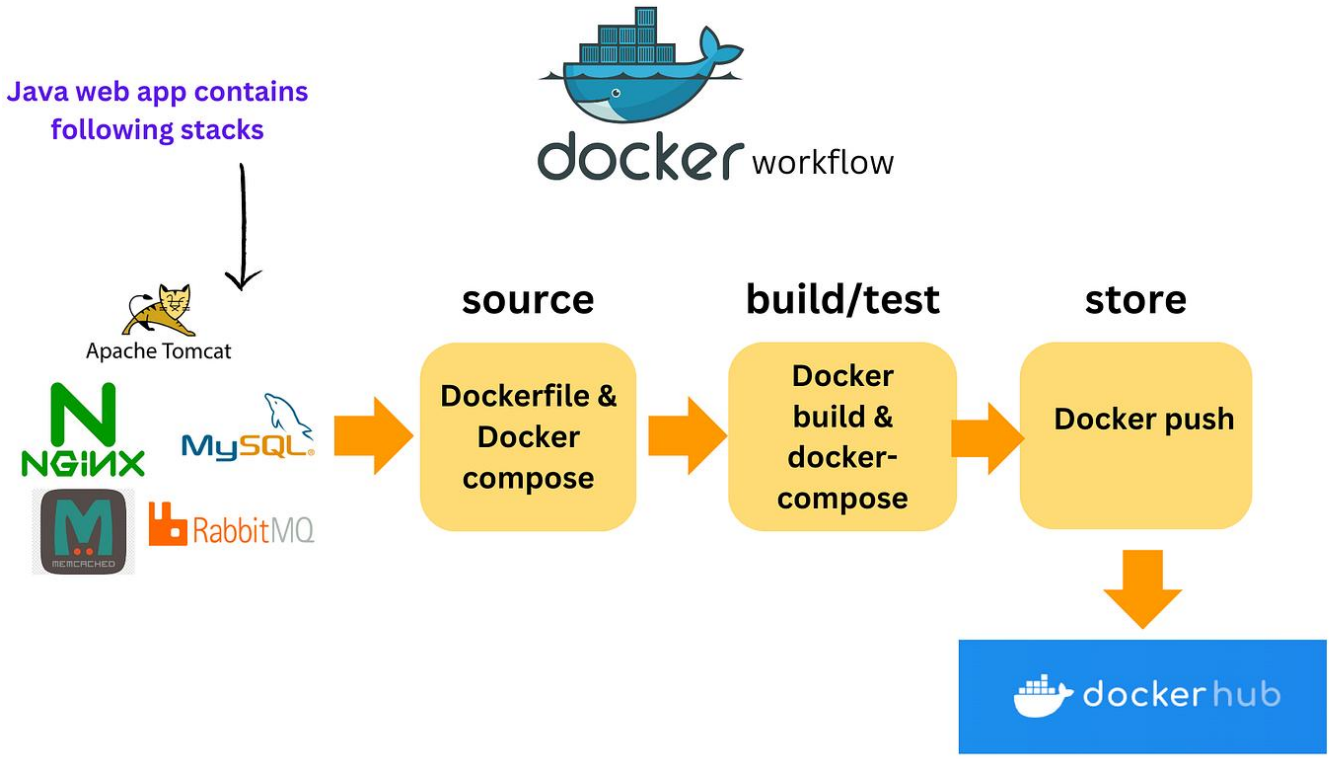
Práctica UD3
Apache Tomcat 11
en
Docker



Autor: Jorge

Fecha: Enero 2026

Curso: 2º DAW - Despliegue de Aplicaciones Web (UD3)

IES: Matemático Puig Adam

Índice

1. Introducción del proyecto

1.1 Objetivo General

nota:

Esta práctica está realizada con el sistema operativo Ubuntu por lo que los comandos no serán iguales en otras distribuciones.

Configurar y administrar un servidor Apache Tomcat 11 en un contenedor Docker con Ubuntu 24.04 y OpenJDK 25, desplegando dos sitios virtuales con aplicaciones Java que demuestren dominio en administración, seguridad y virtualización de servidores de aplicaciones.

1.2 Tecnologías Utilizadas

- **Sistema Operativo Base:** Ubuntu 24.04 LTS
- **JDK:** OpenJDK 25 (Temurin Hotspot)
- **Servidor de Aplicaciones:** Apache Tomcat 11.0.2
- **Contenedorización:** Docker + Docker Compose
- **Build Tool:** Maven 3.x

1.3 Requisitos previos

- ***Hardware Mínimo***

- CPU: 2 núcleos (recomendado 4 núcleos)
- RAM: 4 GB (recomendado 8 GB)
- Disco: 10 GB libres (recomendado 20 GB)
- Arquitectura: x86_64 (AMD64)
- Sistemas Operativos Compatibles
- Linux: Ubuntu 20.04+, CentOS 8+, Debian 10+, Fedora 32+
- macOS: 10.15+ (Catalina o superior)
- Windows: Windows 10 Pro/Enterprise/Education (con WSL2)

- ***Software Requerido***

- Docker Engine

```
sudo apt update
sudo apt install docker.io
sudo systemctl start docker
sudo systemctl enable docker
```

- Verificar instalación

```
docker --version
```

- Docker Compose

```
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

- Verificar instalación

```
docker-compose –version
```

2. Preparación de entorno

- ***Verificar versión de Linux***

release -a debe mostrar versión

```
jorge@jorge:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 24.04.3 LTS
Release:        24.04
Codename:       noble
```

- ***Ver espacio disponible***

df -h

- ***java - obligatorio:***
  - ***Comprobar***

java -version

```
jorge@jorge:~$ java -version
openjdk version "21.0.9" 2025-10-21
OpenJDK Runtime Environment (build 21.0.9+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 21.0.9+10-Ubuntu-124.04, mixed mode, sharing)
```

javac -version

```
jorge@jorge:~$ javac -version
javac 21.0.9
```

  - ***Instalar***

sudo apt-get update
sudo apt-get install openjdk-21-jdk

- ***maven - obligatorio:***
  - ***Comprobar***

mvn -version

```
jorge@jorge:~$ mvn -version
Apache Maven 3.8.7
Maven home: /usr/share/maven
Java version: 21.0.9, vendor: Ubuntu, runtime: /usr/lib/jvm/java-21-openjdk-amd6
4
Default locale: es_ES, platform encoding: UTF-8
OS name: "linux", version: "6.14.0-36-generic", arch: "amd64", family: "unix"
```

  - ***Instalar***

sudo apt-get update
sudo apt-get install maven

- ***GIT - Recomendado:***
  - ***Comprobar***

git -version

```
jorge@jorge:~$ git --version
git version 2.43.0
```

  - ***Instalar***

sudo apt-get install git

- ***Visual Studio Code - Recomendado:***
    - ***Comprobar***

code -version

```
jorge@jorge:~$ code --version
1.104.3
```

    - ***Instalar***

Descargar desde: https://code.visualstudio.com/

- ***Extensiones de VS CODE***
    - ***Obligatorias***

- Extension **Pack for Java** (vscjava.vscode-java-pack)
- **XML** (redhat.vscode-xml)
    - ***Recomendadas***
- **nota: Tomcat for Visual Studio Code** (adashen.vscode-tomcat)
  Sustituido por: **Community Server Connectors** (redhat.vscode-community-server-connector)
- **Git Graph** (mhutchie.git-graph)
- **Prettier - Code formatter** (esbenp.prettier-vscode)

- ***Instalación rápida***

nota: las que ya están instaladas las ignorará a no ser que se utilice –force para actualizarlo.

code --install-extension vscjava.vscode-java-pack
code --install-extension redhat.vscode-xml
*code --install-extension **redhat.vscode-community-server-connector***
code --install-extension mhutchie.git-graph
code --install-extension esbenp.prettier-vscode

```
Installing extensions...
Extension 'vscjava.vscode-java-pack' v0.30.5 is already installed. Use '--force' option to update to latest version or provide '@<version>' to install a specific v
ersion, for example: 'vscjava.vscode-java-pack@1.2.3'.
Installing extensions...
Installing extension 'redhat.vscode-xml'...
Extension 'redhat.vscode-xml' v0.29.0 was successfully installed.
Installing extensions...
Installing extension 'adashen.vscode-tomcat'...
Extension 'adashen.vscode-tomcat' v0.12.1 was successfully installed.
Installing extensions...
Installing extension 'mhutchie.git-graph'...
Extension 'mhutchie.git-graph' v1.30.0 was successfully installed.
Installing extensions...
Installing extension 'esbenp.prettier-vscode'...
Extension 'esbenp.prettier-vscode' v11.0.2 was successfully installed.
```

- ***Docker - Obligatorio:***
    - ***Comprobar***

docker –version

```
jorge@jorge:~$ docker --version
Docker version 28.5.0, build 887030f
```

docker compose version

```
jorge@jorge:~$ docker compose version
Docker Compose version v2.40.0
```

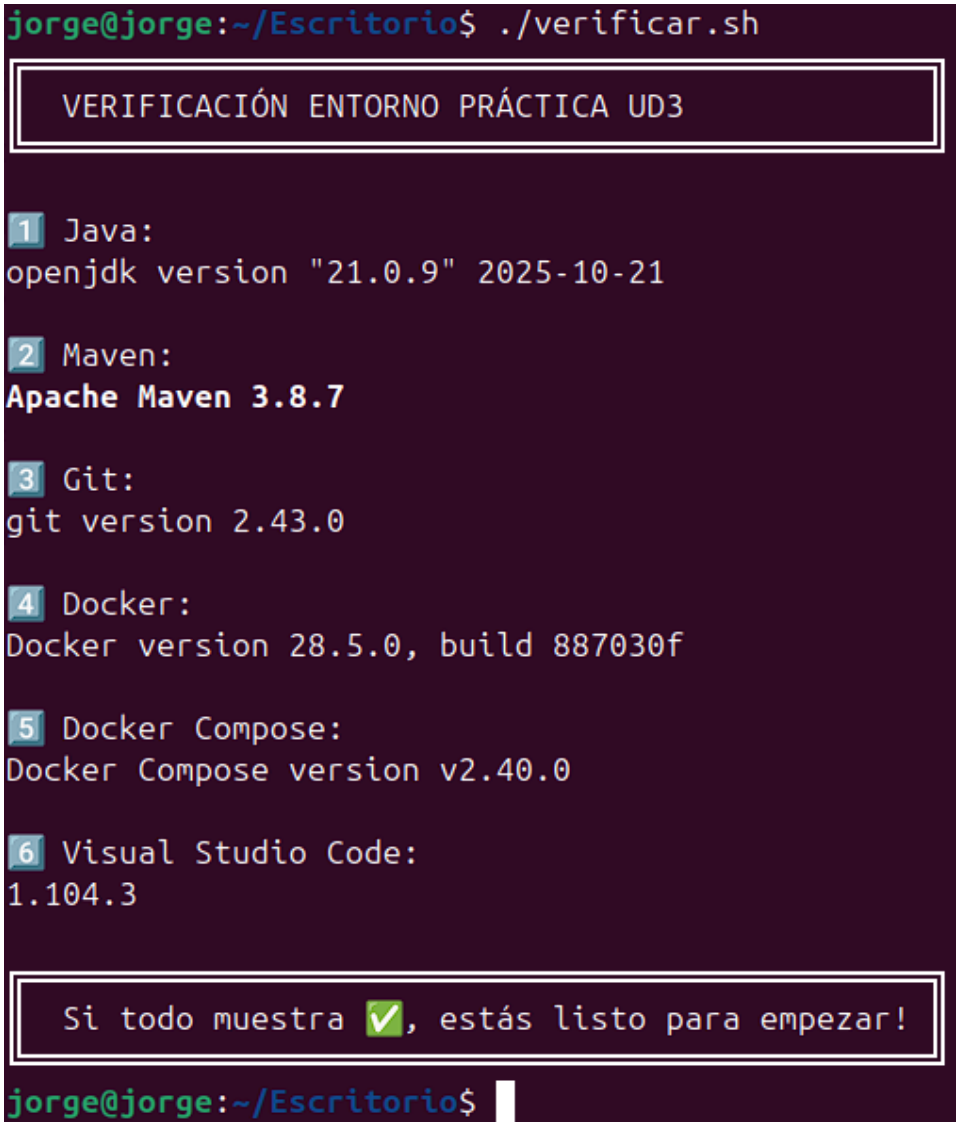
    - ***Instalar compose si no lo tiene***

sudo apt update
sudo apt install docker-compose-plugin

2.1 Verificación completa

Ejecutar el archivo **verificar.sh**

chmod +x verificar.sh
./verificar.sh



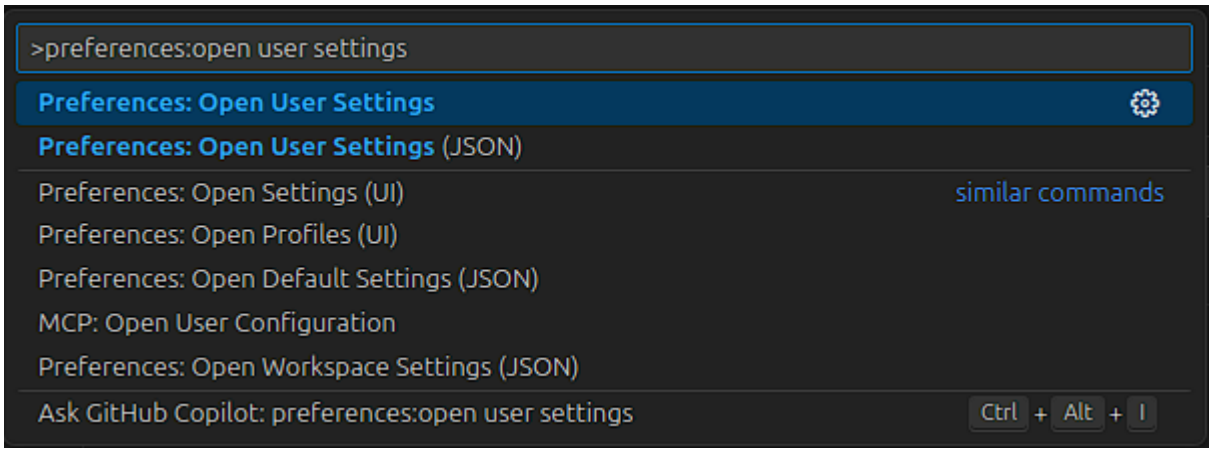
2.1.1 Desde VS Code (opcional)

1. **Crear carpeta .vscode**
2. **crear un settings.json**

Crear un settings.json en la estructura principal del proyecto para personalizarlo.

3. **Entrar en settings.json**

Ctrl+Shift+P → "Preferences: Open User Settings (JSON)"



2.2 Crear estructura de directorios

1. **Crear directorio principal**

mkdir -p ~/Practica_UD3_Jorge
cd ~/Practica_UD3_Jorge

2. **Crear estructura de directorios**

mkdir -p Sitio1/src/main/java/com/sitio1
mkdir -p Sitio1/src/main/webapp/WEB-INF
mkdir -p Sitio2/src/main/java/com/sitio2
mkdir -p Sitio2/src/main/webapp/WEB-INF
mkdir -p conf
mkdir -p webapps
mkdir -p logs

2.3 Crear aplicaciones Java (sitio1 y sitio2)

Sitio 1

1. **Crear Sitio1/pom.xml**

cd ~/Practica_UD3_Jorge/Sitio1
nano pom.xml

2. **Crear HelloServlet.java**

nano src/main/java/com/sitio1/HelloServlet.java

3. **Crear web.xml**

nano src/main/webapp/WEB-INF/web.xml

4. **Compilar**

cd ~/Practica_UD3_Jorge/Sitio1
mvn clean package

5. **Borrar cache de Maven si salió algo mal**

mvn clean

6. **Verificar WAR creado**

ls -lh target/sitio1.war

```
[INFO] Building war: /home/jorge/Practica_UD3_Jorge/Sitio1/target/sitio1.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.559 s
[INFO] Finished at: 2026-01-12T12:28:03+01:00
[INFO] ------------------------------------------------------------------------
jorge@jorge:~/Practica_UD3_Jorge/Sitio1$ ls -lh target/sitio1.war
-rw-rw-r-- 1 jorge jorge 3,4K ene 12 12:28 target/sitio1.war
jorge@jorge:~/Practica_UD3_Jorge/Sitio1$
```

Sitio 2 mismos pasos.

2.1 Fase 1: Preparación del Entorno Docker

2.1.1 Dockerfile:

Dockerfile automatiza la construcción de la imagen del contenedor. Define cada paso necesario para crear un entorno reproducible con todas las dependencias.

Justificación de decisiones clave:

- ***Imagen base***

```
FROM ubuntu:24.04
```

Se usa Ubuntu 24.04 porque proporciona un sistema estable y actualizado, compatible con las versiones más recientes de Java.

- ***2.2. OpenJDK 25 por temuri***

```
RUN mkdir -p /opt && \
  cd /opt && \
  wget https://github.com/adoptium/temurin25-binaries/releases/download/jdk-25.0.1%2B8/OpenJDK25U-jdk_x64_linux_hotspot_25.0.1_8.tar.gz && \
  tar -xzf OpenJDK25U-jdk_x64_linux_hotspot_25.0.1_8.tar.gz && \
  rm OpenJDK25U-jdk_x64_linux_hotspot_25.0.1_8.tar.gz && \
  mv jdk-25.0.1+8 $JAVA_HOME
```

- ***Apache Tomcat 11 en /opt/tomcat***

```
RUN cd /opt && \
  wget https://archive.apache.org/dist/tomcat/tomcat-11/v11.0.2/bin/apache-tomcat-11.0.2.tar.gz && \
  tar -xzf apache-tomcat-11.0.2.tar.gz && \
  mv apache-tomcat-11.0.2 tomcat && \
  rm apache-tomcat-11.0.2.tar.gz
```

- ***variables de entorno***

```
ENV JAVA_HOME=/opt/jdk25
ENV CATALINA_HOME=/opt/tomcat
ENV PATH=$JAVA_HOME/bin:$CATALINA_HOME/bin:$PATH
```

Estas variables permiten que el sistema localice Java y Tomcat sin especificar rutas absolutas cada vez.

- ***Estructura de directorios***

```
RUN mkdir -p $CATALINA_HOME/webapps/sitio1 $CATALINA_HOME/webapps/sitio2
```

Creamos carpetas separadas para cada sitio virtual, permitiendo aislamiento de aplicaciones mediante el concepto de appBase de Tomcat.

- ***Empaquetado como ROOT.war***

```
COPY Sitio1/target/sitio1.war $CATALINA_HOME/webapps/sitio1/ROOT.war
```

Al renombrar los WAR a ROOT.war, las aplicaciones se sirven en la raíz de cada host (/hello en lugar de /sitio1/hello), simplificando las URLs. También me permitio localizar mejor las carpetas.

- ***Exposición de puertos (HTTP HTTPS Y PROTOCOLO AJP(8009))***

```
EXPOSE 8080 8081 8009 8443
```

8009: Protocolo AJP para integración con servidores web como Apache

8443: HTTPS con SSL/TLS

2.2 Fase 2: Configuración de Tomcat

2.1 Configuración del servidor (server.xml)

1. **Crear archivo server.xml**

```
cd ~/Practica_UD3_Jorge
nano server.xml
```

- ***Ajustes clave:***

connectionTimeout="20000": Evita conexiones colgadas (20 segundos es estándar para aplicaciones web)

maxThreads="250": Soporta alta concurrencia sin saturar el sistema

bufferSize="16384": Buffer de 16KB optimiza transferencias de datos moderadamente grandes

- ***Configurar dos conectores HTTP en puertos distintos (ej: 8080 y 8081) para servir los dos sitios***
  - ***puerto 8080 HTTP***

```
<Connector port="8080" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443"
  maxThreads="250"
  bufferSize="16384"
  URIEncoding="UTF-8" />
```

  - ***puerto 8081 HTTP***

```
<Connector port="8081" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443"
  maxThreads="250"
  URIEncoding="UTF-8" />
```

  - ***conector AJP en puerto 8009***

```
<Connector protocol="AJP/1.3" address="0.0.0.0" port="8009" redirectPort="8443"
  secretRequired="false" />
```

para comunicación con servidor web y HTTPS
AJP (Apache JServ Protocol) permite que un servidor web frontal (Apache HTTP, Nginx) envíe peticiones a Tomcat de forma más eficiente que HTTP. secretRequired="false" se usa aquí por simplicidad en desarrollo, pero en producción debería habilitarse con una contraseña compartida.

  - ***puerto 8843 HTTPS***

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
  maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
  clientAuth="want" sslProtocol="TLS">
  <SSLHostConfig>
    <Certificate certificateKeystoreFile="conf/keystore.jks"
      certificateKeystorePassword="changeit" type="RSA" />
  </SSLHostConfig>
</Connector>
```

clientAuth="want": Solicita certificado del cliente pero no lo hace obligatorio, permitiendo conexiones sin certificado mientras se mantiene la opción de validación mutua TLS.

- ***Nombre único y evaluacion de host***

```
<Engine name="Catalina" defaultHost="localhost">
  <Realm className="org.apache.catalina.realm.LockOutRealm">
    <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
      resourceName="UserDatabase" />
  </Realm>
```

- ***Definir dos aplicaciones virtuales (hosts) con nombres distintos***

Opté por cambiarlo y hacer virtuales separados para que cada host tenga su propio appBase, lo que significa que Sitio1 y Sitio2 tienen directorios independientes es util por tema Aislamiento y gestion de logs aparte de una configuración mas independiente de recursos etc.

  - ***Host Virtual Sitio 1***

```
<Host name="sitio1.local" appBase="webapps/sitio1" unpackWARs="true" autoDeploy="true">
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
    prefix="sitio1_access." suffix=".log"
    pattern="%h %l %u %t &quot;%r&quot; %s %b %D" />
</Host>
```

  - ***Host Virtual Sitio 2***

```
<Host name="sitio2.local" appBase="webapps/sitio2" unpackWARs="true" autoDeploy="true">
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
    prefix="sitio2_access." suffix=".log"
    pattern="%h %l %u %t &quot;%r&quot; %s %b %D" />
</Host>
```

  - ***Host Virtual manager***

```
<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">
  <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
    prefix="localhost_access_log" suffix=".txt"
    pattern="%h %l %u %t &quot;%r&quot; %s %b" />
</Host>
```

- ***Auditoría y logging de accesos***

```
<Valve className="org.apache.catalina.valves.AccessLogValve"
pattern="%h %l %u %t &quot;%r&quot; %s %b %D" />/>
```

El patrón incluye %D que mide el tiempo de respuesta en milisegundos, util para cuellos de botella.

- ***Users***

```
<role rolename="manager-gui" />
<role rolename="admin-gui" />
```

admin: Acceso total (manager + admin + scripts JMX)

manager: Solo interfaz gráfica del Manager

viewer: Solo visualización de estado

2. 2.2 Configuración de Seguridad

1. Creamos tomcat-users.xml

```
cd ~/Practica_UD3_Jorge
nano tomcat-users.xml:
```

- ***Implementar gestión de usuarios en tomcat-users.xml (usuario administrador)***

```
<user username="admin"
    password="admin123"
    roles="manager-gui,manager-script,manager-jmx,manager-status,admin-gui,admin-script"/>

<!-- Usuario manager -->
<user username="manager"
    password="manager123"
    roles="manager-gui,manager-status"/>

<!-- Usuario viewer -->
<user username="viewer"
    password="viewer123"
    roles="viewer,manager-status"/>
```

- Definir **roles y permisos** (admin, manager, viewer)

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<role rolename="admin-gui"/>
<role rolename="admin-script"/>
<role rolename="viewer"/>
```

2. **Restricción de acceso (solo localhost o red específica)**

- ***Crear context-xml***

```
nano context.xml
```

- ***Límites de recursos por aplicación***

```
<Resources cachingAllowed="true" cacheMaxSize="100000" cacheTtl="5000" />
```

cacheMaxSize="100000": Caché de 100MB evita recargas innecesarias

cacheTtl="5000": Los recursos se revalidan cada 5 segundos, balance entre rendimiento y actualización

- Configurar **restricciones de acceso** al Manager

```
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127\.0\.0\.1|::1|192\.168\.\d+\.\d+|.*" />
```

Nota de seguridad: El patrón ^.*$ permite acceso desde cualquier IP.

3. **Establecer HTTPS (certificado autofirmado SSL/TLS) en uno de los sitios**

- ***Creamos carpeta***

```
mkdir -p conf
cd conf
```

- ***Creamos certificado***

```
keytool -genkey -alias tomcat -keyalg RSA -keysize 2048 -validity 365 \
 -keystore keystore.jks -storepass changeit -keypass changeit \
 -dname "CN=localhost, OU=IT, O=IES, L=Madrid, ST=Madrid, C=ES"
```

Generamos un certificado autofirmado RSA de 2048 bits. En producción se usaría un certificado firmado por una CA (Let's Encrypt, DigiCert).

2.2.1 Componentes Web

Ahora crearemos las aplicaciones Java con todos los componentes requeridos.

- ***Crear estructura de directorios (si aún no se ha creado)***
- ***Crear pom.xml***

```
cd ~/Practica_UD3_Jorge/Sitio1
nano pom.xml
```

1. **Crear Servlet (HelloServlet.java) con funcionalidad básica**

```
nano src/main/java/com/sitio1/HelloServlet.java
```

```
String javaVersion = System.getProperty("java.version");
String tomcatVersion = getServletContext().getServerInfo();
String hostname = InetAddress.getLocalHost().getHostName();
```

El servlet captura información del sistema para verificar que las variables de entorno están correctamente configuradas.

Filtro de Logging (LoggingFilter.java)

Propósito: Intercepta cada petición para registrar metadatos antes de que llegue al servlet.

Ventaja: Centraliza el logging sin modificar cada servlet individualmente.

Listener de Aplicación (AppListener.java)

Propósito: Detecta eventos del ciclo de vida de la aplicación (inicio/parada).

Uso típico: Inicialización de conexiones a bases de datos, carga de configuración global.

Gestión de Sesiones (web.xml)

2. **Gestion de sesiones (web.xml)**

```
<session-config>
  <session-timeout>30</session-timeout>
  <cookie-config>
    <http-only>true</http-only>
  </cookie-config>
</session-config>
```

http-only="true": Protege las cookies de sesión contra ataques XSS (JavaScript no puede acceder a ellas)

session-timeout="30": Sesiones expiran tras 30 minutos de inactividad

- ***JSP para renderizado dinámico***

nano src/main/webapp/info.jsp

- ***Filtros de autenticación y logging***

nano src/main/java/com/sitio1/ LoggingFilter.java

- ***Listeners para eventos de aplicación***

nano src/main/java/com/sitio1/AppListener.java

- ***Gestión de sesiones persistentes***

```
<Manager className="org.apache.catalina.session.PersistentManager">
  <Store className="org.apache.catalina.session.FileStore" />
</Manager>
```

2.2.2 Las aplicaciones java serán:

Sitio 1:

- Servlet que devuelve: "¡Hola mundo desde el Sitio 1!"
- Incluir información de versión de Java, Tomcat y hostname

Sitio 2:

- Servlet que devuelve: "¡Hola mundo desde el Sitio 2!"
- Incluir timestamp y dirección IP del contenedor

Ambas aplicaciones deben:

- Tener estructura estándar Maven/Gradle
- Incluir un archivo web.xml configurado
    - sitio1

nano src/main/webapp/WEB-INF/web.xml

- Ser desplegables como archivos WAR
- Responder en rutas distintas por sitio

3. **Al acabar acordarse de Compilar**

cd ~/Practica_UD3_Jorge/Sitio1
mvn clean package

Hacer lo mismo para sitio2

2.3 Docker Compose

Crear un archivo docker-compose.yml

cd ~/Practica_UD3_Jorge
nano docker-compose.yml

Debe cumplir:

- ***Construya la imagen Docker automáticamente***

```
build: .
container_name: tomcat-jorge-ud3
```

build: . indica que debe construir la imagen usando el Dockerfile en el directorio actual.

- ***Mapeo de puertos (8080, 8081, 8443, 8009)***

```
ports:
 - "8080:8080"
 - "8081:8081"
 - "8443:8443"
 - "8009:8009"
```

Formato "host:contenedor" - permite acceder desde el navegador local a los puertos del contenedor.

- ***Monte volúmenes para persistencia de configuración y logs***

```
volumes:
- ./server.xml:/opt/tomcat/conf/server.xml
- ./logs:/opt/tomcat/logs
```

    - ***Los volúmenes son muy útiles por dos motivos:***

Persistencia: Los logs sobreviven, aunque se elimine el contenedor

Hot-reload: Cambios en server.xml sin reconstruir la imagen (reiniciando Tomcat)

El resto no los traigo porque los voy a copiar directamente excepto **context.xml**

- ***Variables de entorno***

environment:
JAVA_OPTS: "-Xmx512m -Xms256m"
CATALINA_OPTS: "-Dfile.encoding=UTF-8"

-Xmx512m: Heap máximo de 512MB

-Xms256m: Heap inicial de 256MB

Se aplica al rendimiento y consumo de memoria.

- ***fácil inicio/parada del servicio***

restart: unless-stopped
networks:
- tomcat-network

2.4 Construcción y ejecución

1. **Compilar aplicaciones:**

en Sitio1 y Sitio2 usar desde terminal:

mvn clean package

2. **Paso 1: Construir la imagen**

cd ~/Practica_UD3_Jorge
docker compose build --no-cache

3. **Paso 2: Iniciar el contenedor**

docker compose up -d

4. **Ver logs**

docker compose logs -f

```
jorge@jorge:~/Practica_UD3_Jorge$ docker compose logs -f
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.734 WARNING [main] org.apache.tomcat.util.digester.SetPropertiesRule.begin Match [Server/Service/Connector] failed to set property
 [clientAuth] to [false]
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.736 WARNING [main] org.apache.tomcat.util.digester.SetPropertiesRule.begin Match [Server/Service/Connector] failed to set property
 [sslProtocol] to [TLS]
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.763 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version name:   Apache Tomcat/11.0.2
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.763 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built:          Dec 5 2024 15:19:37 UTC
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.764 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version number: 11.0.2.0
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.764 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Name:               Linux
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.764 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Version:            6.14.0-37-generic
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.764 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Architecture:          amd64
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.764 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Java Home:             /opt/jdk25
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.764 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Version:           25.0.1+8-LTS
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.764 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Vendor:            Eclipse Adoptium
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.764 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_BASE:         /opt/tomcat
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.765 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_HOME:         /opt/tomcat
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.765 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.config.file=/opt/
tomcat/conf/logging.properties
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.766 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djava.util.logging.manager=org.apach
e.juli.ClassLoaderLogManager
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.766 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Xmx512m
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.766 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Xms256m
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.766 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Djdk.tls.ephemeralDHKeySize=2048
tomcat-jorge-ud3  | 14-Jan-2026 21:43:14.766 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line argument: -Dorg.apache.catalina.security.Securi
```

5. **Verificar que está corriendo**

docker ps

```
jorge@jorge:~/Practica_UD3_Jorge$ docker ps
CONTAINER ID   IMAGE                      COMMAND             CREATED       STATUS         PORTS
                                                                NAMES
faa1c03a274f   practica_ud3_jorge-tomcat   "catalina.sh run"   7 hours ago   Up 8 minutes   0.0.0.0:8009->8009/tcp, [::]:8009->8009/tcp, 0.0.0.0:8080-8081->8080-808
1/tcp, [::]:8080-8081->8080-8081/tcp, 0.0.0.0:8443->8443/tcp, [::]:8443->8443/tcp   tomcat-jorge-ud3
```

2.5 Tarea 6: Pruebas de Funcionamiento

Hice Resolución de nombres local en /etc/hosts

por lo que accederé a las URL con ello. Simplificare llamandolo DNS aunque no sea exactamente lo mismo en estos casos.

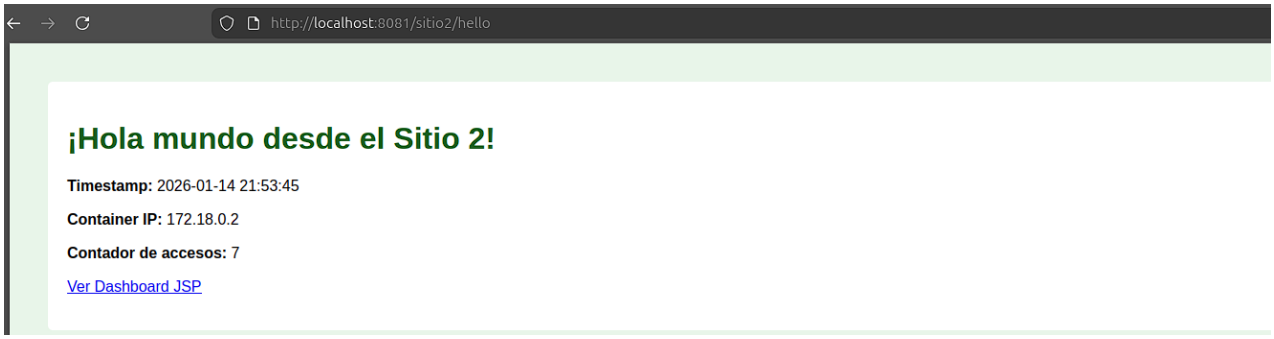
```
  GNU nano 7.2
127.0.0.1 localhost
127.0.1.1 jorge

127.0.0.1 empresa.local
127.0.0.1 clientes.local

127.0.0.1 sitio1.local
127.0.0.1 sitio2.local
```

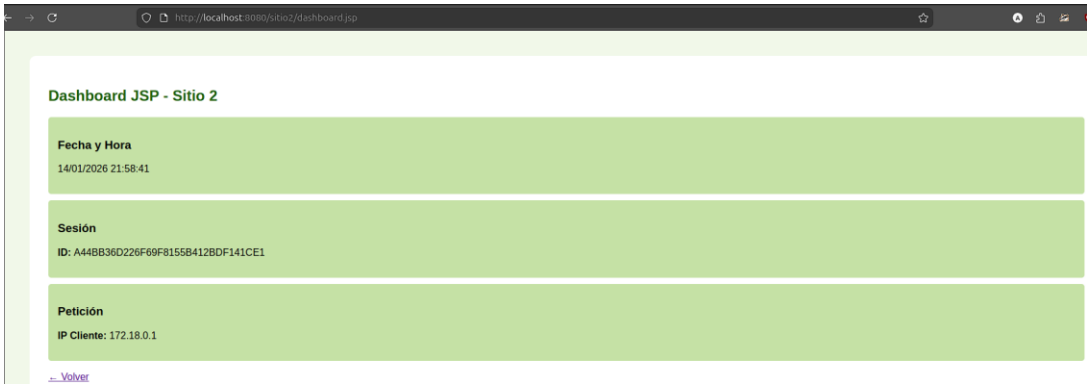
- **Pruebas de acceso HTTP** a ambos sitios desde navegador
    - **Sitio 1**

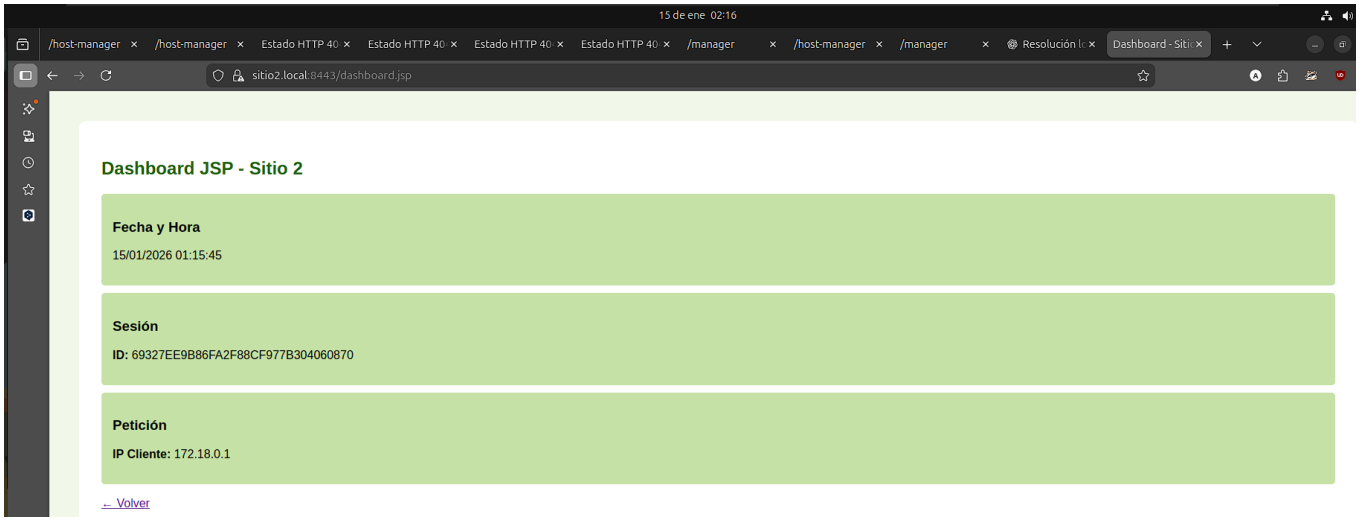
- con DNS → http://sitio1.local:8080/hello
    - Si tuviese ambos en localhost
- curl http://localhost:8080/sitio1/hello
- abrir navegador: http://localhost:8080/sitio1/hello

- ***Sitio 2***

- con DNS → http://sitio2.local:8081/hello (actual activo)
- abrir navegador: http://localhost:8081/sitio2/hello

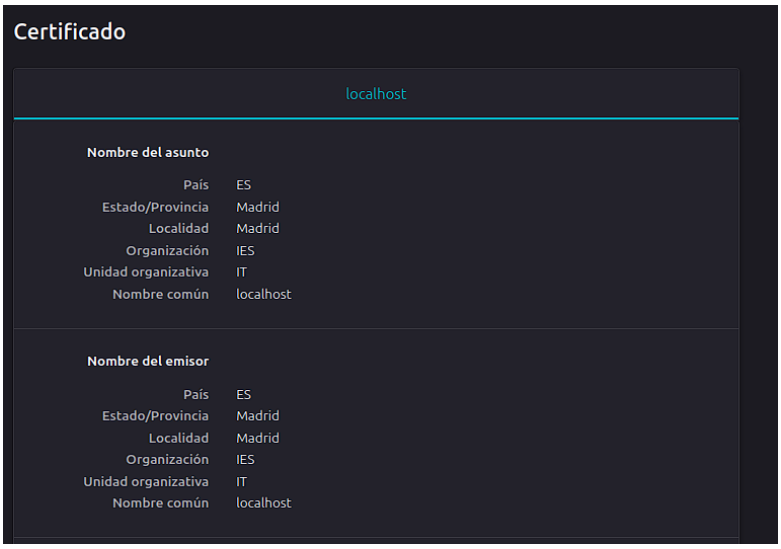






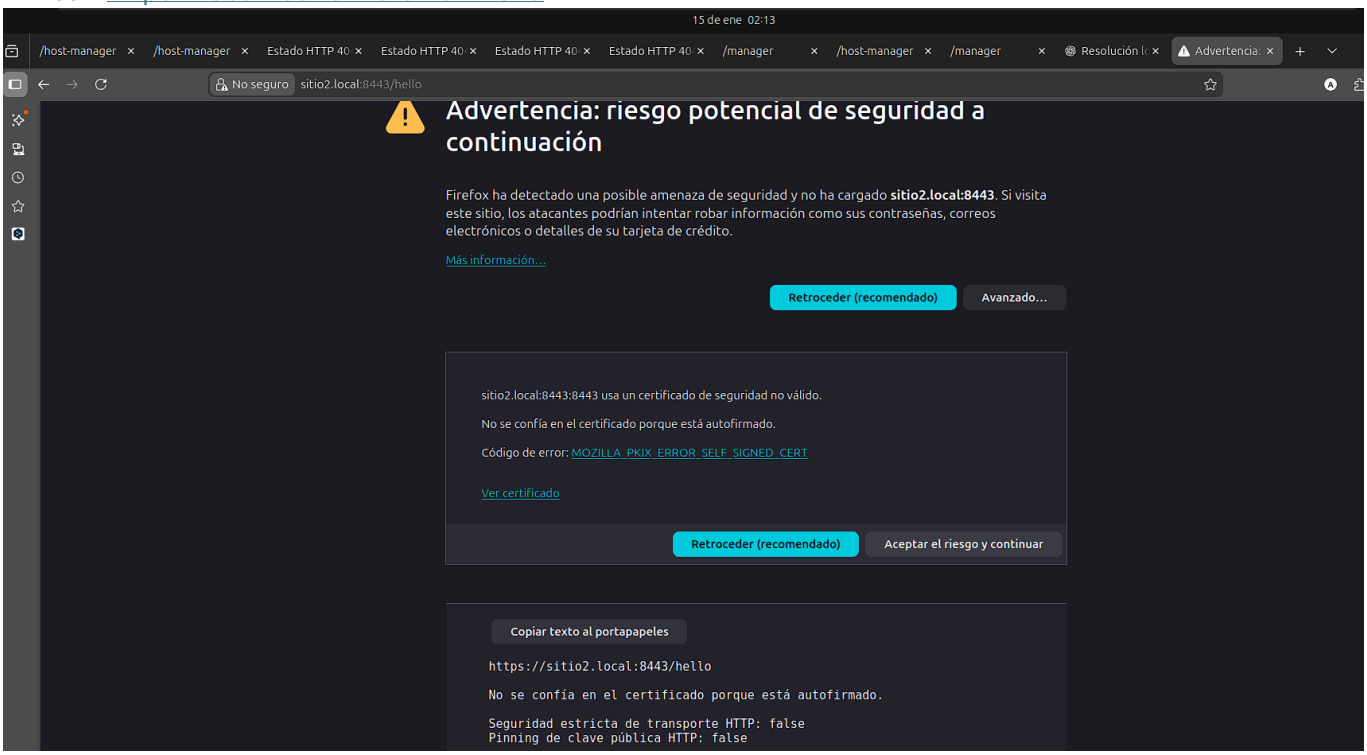
- **Pruebas de acceso HTTPS Sitio1** con validación de certificado

- con DNS
- curl https://localhost:8443/sitio1/hello

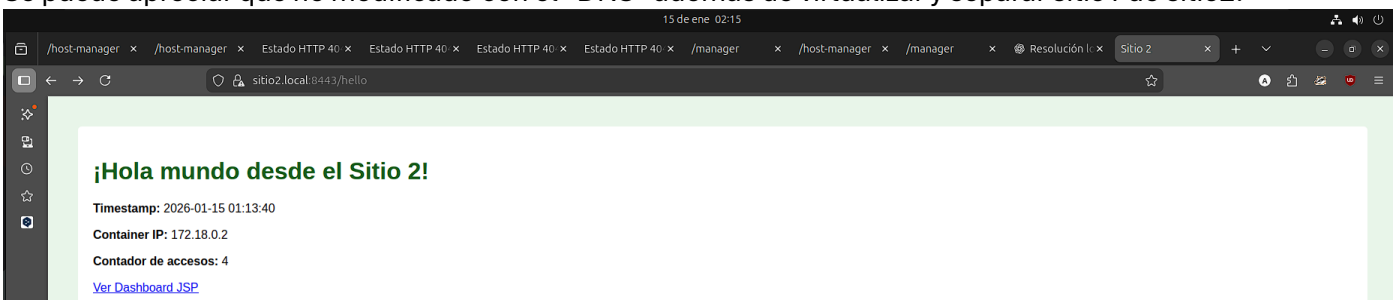


- **Pruebas de acceso HTTPS Sitio2** con validación de certificado

- curl https://localhost:8443/sitio2/hello

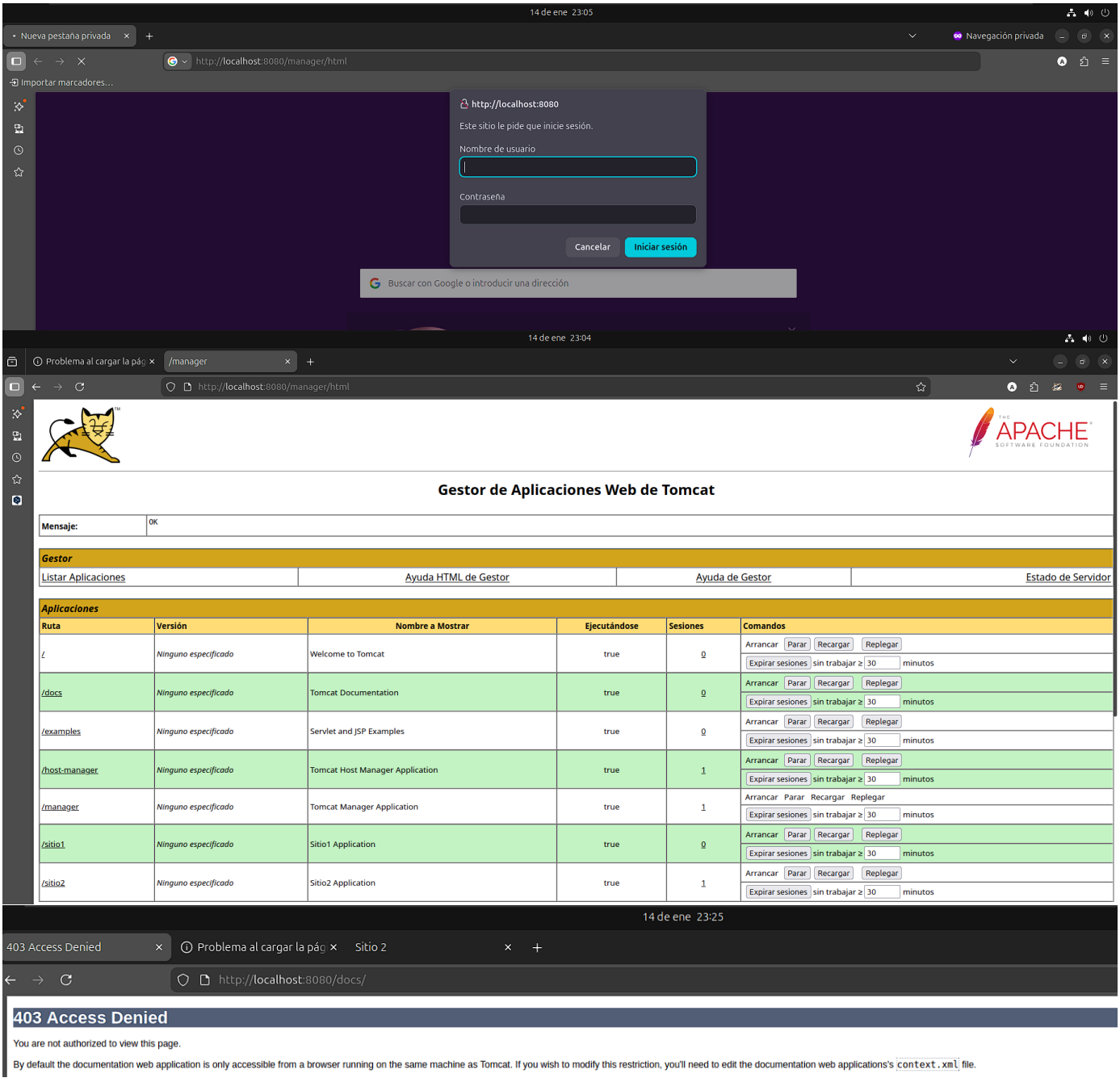


Se puede apreciar que he modificado con el “DNS” además de virtualizar y separar sitio1 de sitio2.



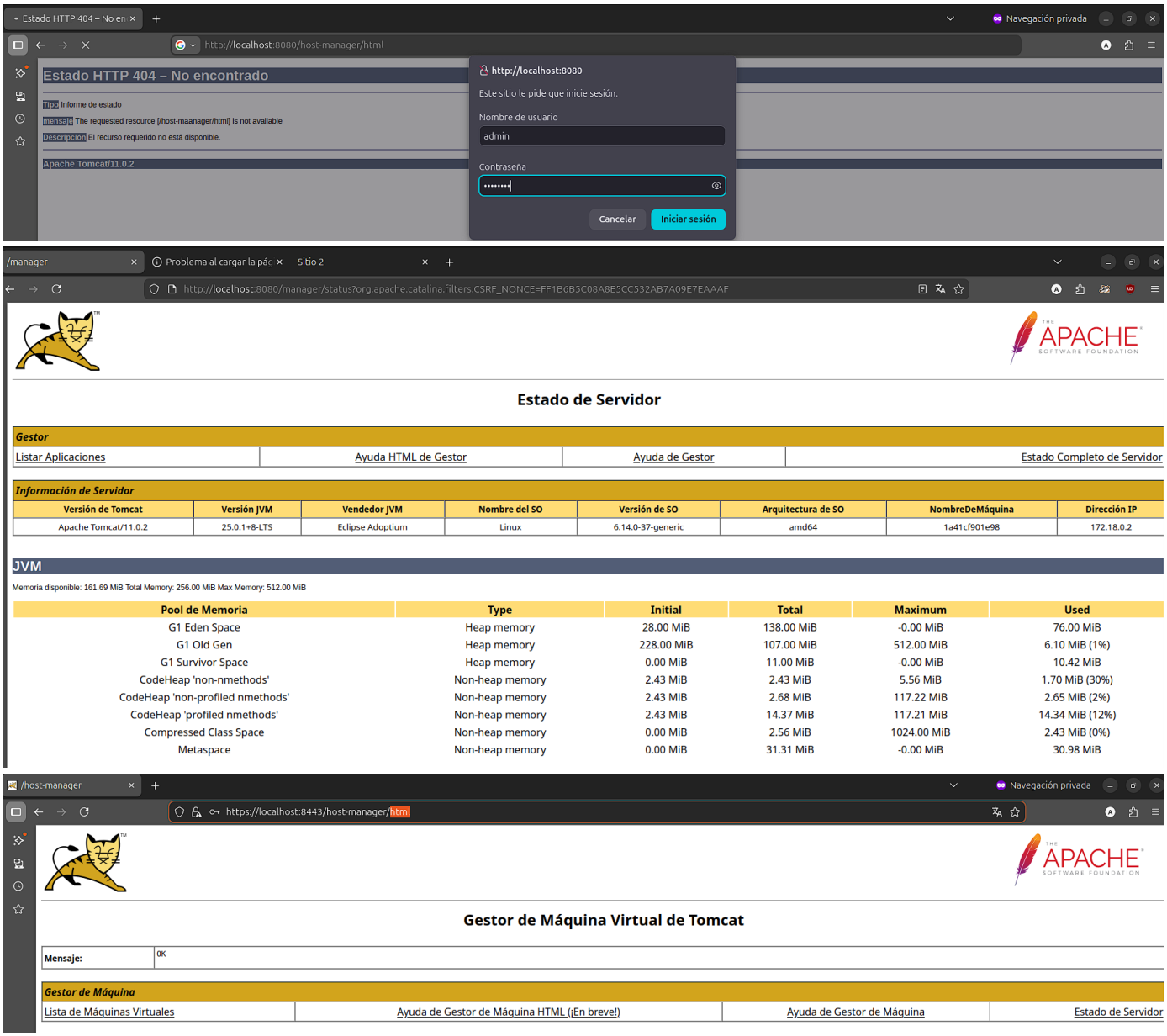
- **Acceso al Manager** de Tomcat

- Abrir navegador: http://localhost:8080/manager/html
    - Usuario: admin
    - Contraseña: admin123



- **Acceso al Host-Manager** de Tomcat

- Abrir navegador: http://localhost:8080/host-manager/html
    - Usuario: admin
    - Contraseña: admin123



Como el conector HTTPS está definido a nivel de **Service**, cualquier aplicación que esté dentro de ese servicio es accesible de forma segura.

Esto es gracias a que he separado el server.xml, el conector de la red (el puerto 8443) y el motor de las aplicaciones (el Engine).

Y como El **Connector HTTPS** es como una "puerta de entrada" una vez que pasas por esa puerta, el **Engine** te da acceso a todas las páginas establecidas (sitio1, sitio2, manager).

- ***¿Por qué el 8080 y 8081 sirven para las dos páginas?***

Esto se debe a la jerarquía de Tomcat.

- **Los Conectores (Puertos):** He definido tres (8080, 8081 y 8443). Todas ellas escuchan peticiones y las mandan al **mismo motor** (Engine).
- **El Engine y el Host:** Como solo hay un Host llamado localhost, y ese host tiene acceso a la carpeta webapps, cualquier petición que llegue por cualquier puerto acabará buscando en esa carpeta.

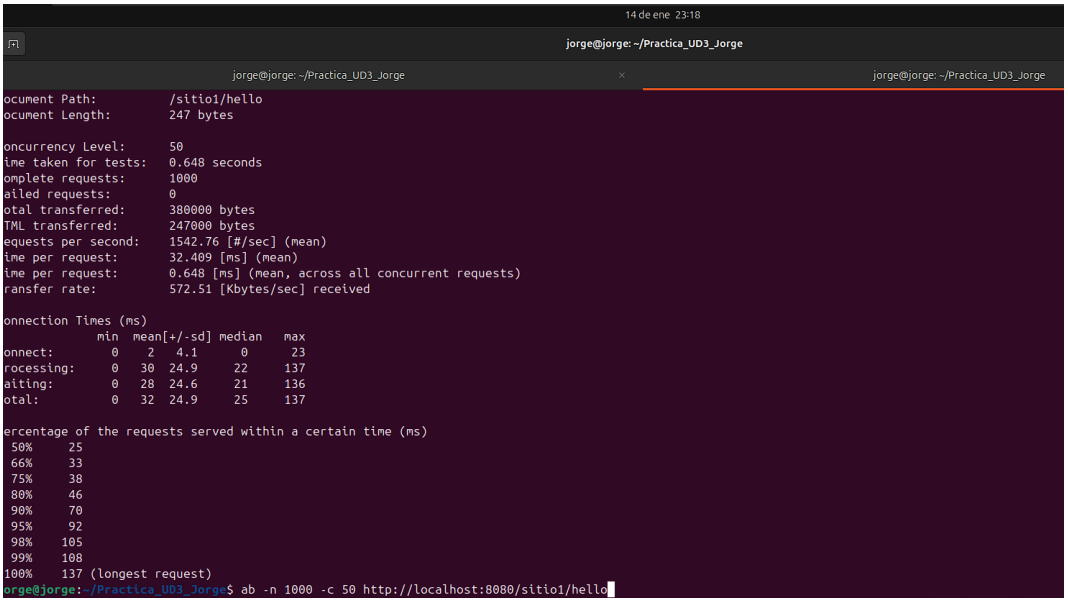
- ***Pruebas de carga usando herramientas como ab (Apache Bench) o curl***

sudo apt-get install apache2-utils
ab -n 1000 -c 50 http://localhost:8080/sitio1/hello
ab -n 500 -c 20 http://sitio1.local:8080/hello
ab -n 500 -c 20 http://sitio2.local:8080/hello

Parámetros:

- **-n 1000:** 1000 peticiones totales
- **-c 50:** 50 peticiones concurrentes

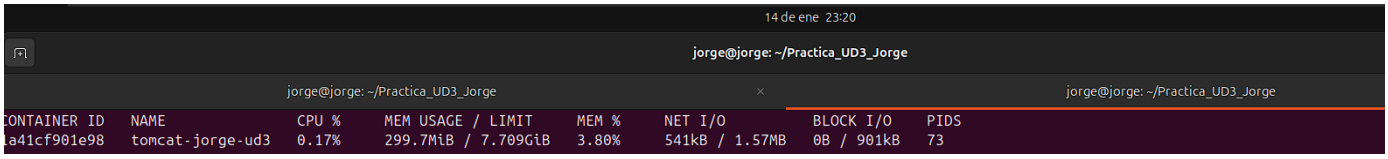
Resultado esperado: Tiempo de respuesta < 50ms bajo carga moderada.



- **Monitoreo de recursos** (CPU, memoria, conexiones)

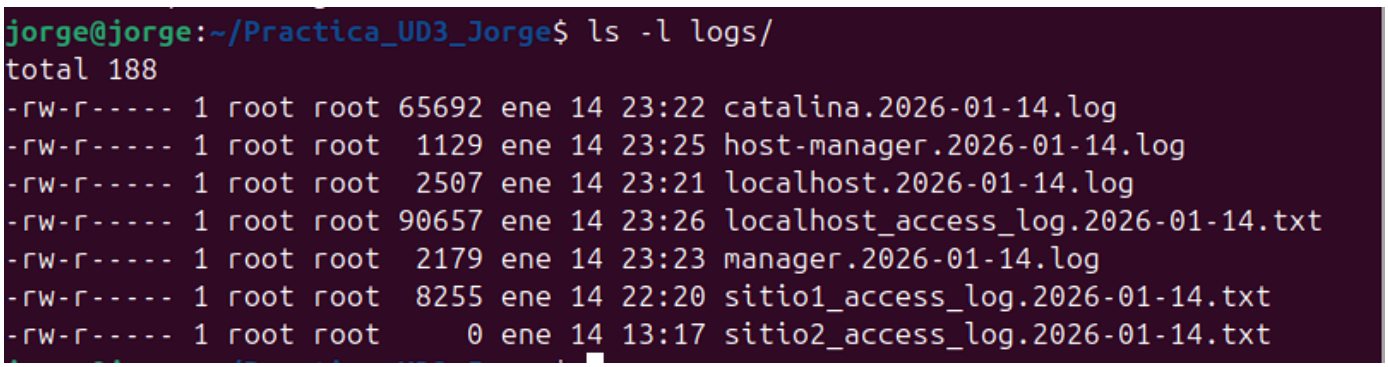
docker stats tomcat-jorge-ud3

Muestra CPU, memoria y I/O en tiempo real.



- **Logs y auditoría** de accesos exitosos y fallidos

localhost_access_log.2026-01-14.txt: Este es el que registra todo ahora mismo (incluidos Sitio 1, Sitio 2 y Manager) porque unifiqué el Host. Tiene **90 KB**



- ***Para poder leer los logs hay que dar permisos al usuario***

sudo chown -R $USER:$USER logs/

- ***logs de los servlets***

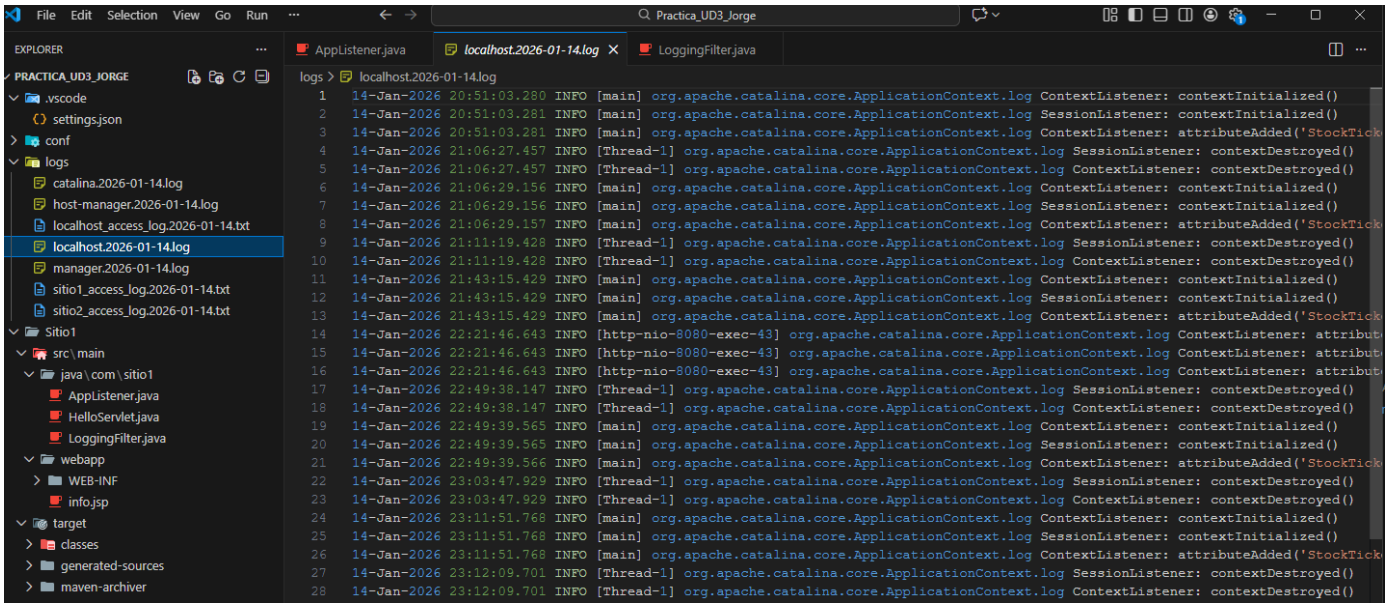
docker logs -f tomcat-jorge-ud3

Recargo la página varias veces para asegurarme de que lo registra bien

- ***logs***

Como he tenido dos versiones la de localhost y separados se me han generado logs antiguos que no he eliminado

Podría elimarlos con el siguiente comando



*rm -rf ./logs/**

- ***Seguimiento de logs***

tail -f logs/sitio1_access_log.txt
tail -f logs/sitio2_access_log.txt
tail -f logs/catalina.out
tail -f logs/localhost_access_log.2026-01-14.txt

  - ***También puedo usar un grep para filtrar por sitio2***

tail -f logs/localhost_access_log.2026-01-14.txt | grep "/sitio2"

- ***Reinicio de aplicaciones sin reiniciar servidor***

Una de las funcionalidades del Manager es reiniciar las aplicaciones por comandos sin tener que apagar y encender todo el contenedor Docker. Es lo que se usa en producción para actualizar una web sin tirar el servidor entero.

  - **Para ejecutarlo correctamente**

- **El usuario y pass:** Que coincida con lo que puse en tu tomcat-users.xml
  en mi caso → admin:admin123
- **La ruta:** /sitio1 es el nombre del contexto que definimos.

Si no se encuentra la ruta se puede usar el siguiente comando para ver la lista:

curl -u admin:admin123 http://localhost:8080/manager/text/list

Ventaja: Actualizaciones sin downtime del servidor completo.

```
jorge@jorge:~/Practica_UD3_Jorge$ curl -u admin:admin123 http://localhost:8080/manager/text/list
OK - Listed applications for virtual host [localhost]
/:running:0:ROOT
/examples:running:0:examples
/host-manager:running:2:host-manager
/sitio2:running:1:sitio2
/manager:running:1:manager
/docs:running:0:docs
```

Por eso no aparecia sitio1 se borró en los pasos anteriores.

Por lo que lo arreglaré con

mvn clean package

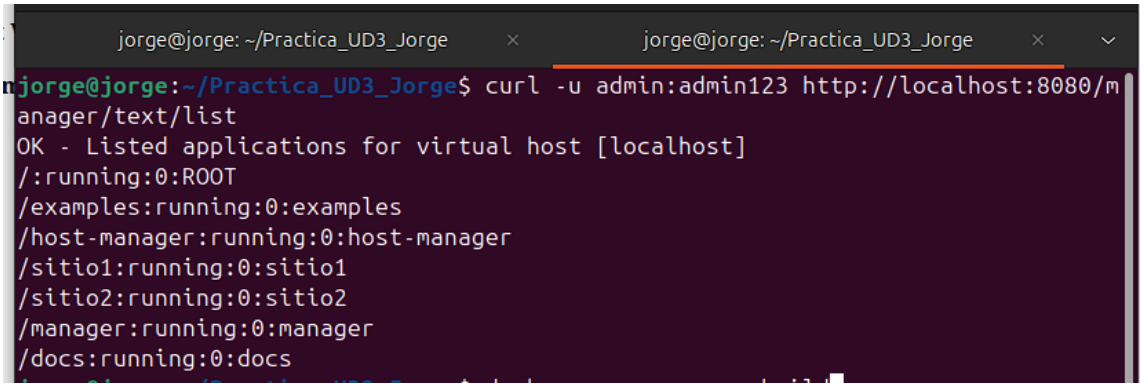
en el sitio1 por si tenía un archivo antiguo o corrupto, así borro la carpeta target y limpio la basura y luego con package compilo el nuevo codigo y genero un .war

y luego con

docker-compose up –build

para indicar a docker los nuevos archivos a copiar

curl -u admin:admin123 http://localhost:8080/manager/text/reload?path=/sitio1



Ahora ya puedo usar el reload:

curl -u admin:admin123 http://localhost:8080/manager/text/reload?path=/sitio1

```
jorge@jorge:~/Practica_UD3_Jorge$ curl -u admin:admin123 "http://localhost:8080/manager/text/reload?path=/sitio1"
OK - Reloaded application at context path [/sitio1]
jorge@jorge:~/Practica_UD3_Jorge$
```

2.6 Decisiones técnicas destacadas

1. **7.1 ¿Por qué 8080 y 8081 sirven ambos sitios?**

Tomcat tiene una arquitectura jerárquica:

2. **Conectores (puertos 8080, 8081, 8443) escuchan peticiones**
3. **Todas las peticiones van al mismo Engine**

1. El Engine decide qué **Host** responde según el header Host: de HTTP

Sin embargo, con la resolución DNS local (/etc/hosts):

- ***sitio1.local:8080 → resuelve al Host sitio1.local***
- ***sitio2.local:8081 → resuelve al Host sitio2.local***

4. **7.2 Estructura Unificada vs. Separada**

Decisión final: Host localhost unificado + Hosts virtuales separados

Ventaja: El Manager es accesible desde cualquier puerto, mientras que las aplicaciones mantienen aislamiento mediante appBase.

2.7 Recomendaciones DE PRODUCCIÓN

1. **Certificados:** Usar Let's Encrypt con renovación automática
2. **Contraseñas:** Encriptar con algoritmo PBKDF2 en tomcat-users.xml
3. **Firewall:** Restringir acceso al Manager por IP
4. **Logs:** Rotar logs automáticamente (logrotate)
5. **Backups:** Automatizar respaldos de webapps/ y configuraciones
6. **Monitoreo:** Integrar con Prometheus/Grafana
7. **Escalabilidad:** Usar balanceador de carga (mod_jk) con múltiples instancias Tomcat

2.8 Resolución de problemas

- Problema: "Address already in use" al iniciar Docker
  Solución: Verificar que no haya otra instancia corriendo
- Certificado SSL rechazado
  Solución: Importar certificado en el navegador o usar flag `-k` en curl
- Manager no accesible
  Solución: Verificar que `context.xml` permite la IP del cliente

Este proyecto demuestra una implementación completa de Apache Tomcat 11 con:
Virtualización mediante Docker
Aislamiento de aplicaciones con hosts virtuales
Seguridad reforzada (SSL/TLS)
Optimización de rendimiento (thread pools, buffers)
Auditoría completa (logs independientes por host)

La arquitectura es escalable y adaptable a entornos de producción con las recomendaciones aplicadas.

Anexo A: Estructura Completa del Proyecto

```
Practica_UD3_Jorge/
├── Servidor Web TomCat.docx
├── Dockerfile
├── docker-compose.yml
├── README.md
├── server.xml
├── tomcat-users.xml
├── context.xml
├── conf/
│   └── keystore.jks
├── logs/
├── Sitio1/
│   ├── pom.xml
│   ├── src/main/java/com/sitio1/
│   │   ├── HelloServlet.java
│   │   ├── LoggingFilter.java
│   │   └── AppListener.java
│   ├── src/main/webapp/
│   │   ├── info.jsp
│   │   └── WEB-INF/web.xml
│   └── target/sitio1.war
└── Sitio2/
    ├── pom.xml
    ├── src/main/java/com/sitio2/
    │   ├── HelloServlet.java
    │   ├── AuthFilter.java
    │   └── SessionListener.java
    ├── src/main/webapp/
    │   ├── dashboard.jsp
    │   └── WEB-INF/web.xml
    └── target/sitio2.war
```