MEDAC
Instituto Oficial de Formación Profesional

DESPLIEGUE DE APLICACIONES WEB
TÉCNICO EN DESARROLLO DE APLICACIONES WEB

Administración de servidores de aplicaciones

07

ÍNDICE

 ISBN: 978-84-18983-26-9

OBJETIVOS

Conocer la arquitectura de un servidor de aplicaciones.

Realizar configuraciones básicas en un servidor de aplicaciones.

Configurar un servidor seguro con SSL.

/ 1. Introducción y contextualización práctica

En este tema, hablaremos de la arquitectura básica de un servidor de aplicaciones y de los procesos que tienen lugar desde que se realiza una petición hasta que se muestra el resultado en el navegador del cliente.

Aprenderemos dónde se almacenan los ficheros de configuración y los logs de los servidores de aplicaciones, además, realizaremos configuraciones sobre estos, en concreto, en Apache Tomcat y Node.js.

Planteamiento del caso práctico inicial

A continuación, vamos a plantear un caso a través del cual podremos aproximarnos de forma práctica a la teoría de este tema.

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema, encontrarás su resolución en el apartado Resumen y Resolución del caso práctico.

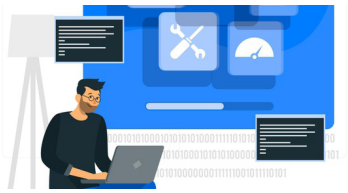

Fig. 1. Administración de servidores de aplicaciones web.

Audio intro. "Error en la configuración del puerto"
https://bit.ly/2X71ddg

/ 2. ¿Qué son los servidores de aplicaciones?

Dependiendo de las necesidades de nuestro desarrollo, podremos encontrar que la arquitectura de una aplicación web puede ser cliente-servidor o regirse por el modelo de tres capas.

Esta variación dependerá de la existencia de un servidor web.

En caso de tener un servidor web que atienda las peticiones y las derive al servidor de aplicaciones si fuese necesario, estaremos hablando del modelo de tres capas. En caso contrario, se establecerá una relación cliente servidor.

En cuanto a los propios servidores de aplicaciones, también utilizan una arquitectura interna y unos procesos bien definidos para atender peticiones. En caso de Apache Tomcat, la estructura del servidor es la siguiente:

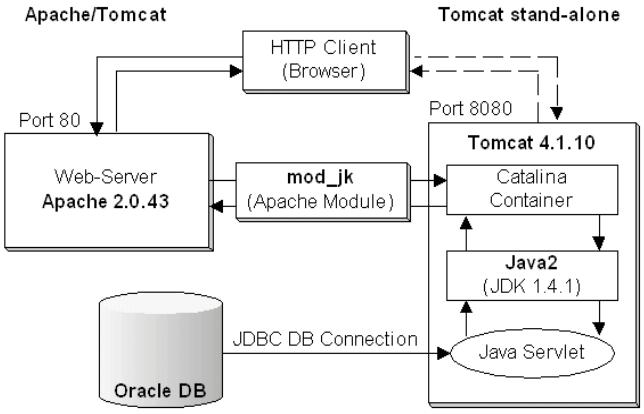

Fig. 2. Apache Tomcat

En el caso de Node.js, las peticiones tienen un curso diferente y entran en bucle de eventos de JavaScript, realizando peticiones a la base de datos por hilos.

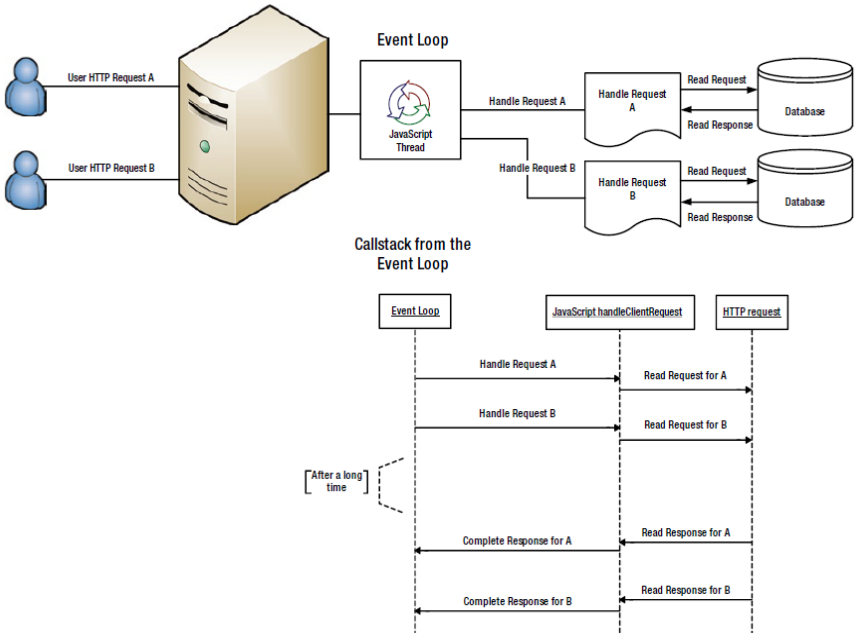

Fig. 3. Gestión de un servidor de Node.js de dos peticiones. Extracto de "Beginning Node.js":

En el siguiente enlace, podrás ampliar la información sobre el funcionamiento de Node.js: https://www.genbeta.com/desarrollo/como-funciona-node-js

/ 3. Apache Tomcat: Configuración

Bajo un sistema operativo Linux, Apache Tomcat almacena los ficheros de configuración en el directorio /usr/share/tomcat/conf.

```
/usr/share/tomcat9/                    DIRECTORIO RAÍZ
|-- bin
|-- BUILDING.txt
|-- conf
|--- CONTRIBUTING.md
|-- -lib
|--- LICENSE
|--- logs
|--- NOTICE
|--- README.md
|--- RELEASE-NOTES
|--- RUNNING.txt
|--- temp
|--- webapps
|--- work
```

Código 1. Listado de directorios de Tomcat en Ubuntu.

Entre los archivos de configuración que encontramos en la carpeta /usr/share/tomcat/conf, los más importantes son:

- **Web.xml:** En este archivo, se pueden modificar los parámetros generales del servidor de aplicaciones.
- **Tomcat-users.xml:** En este fichero, creamos los usuarios que pueden acceder al apartado de administración de Apache Tomcat.
- **Server.xml:** En este fichero, se especifican los conectores. Cada conector establece un puerto de escucha para Apache Tomcat que podremos cambiar.

Dentro del directorio de Tomcat, encontraremos la carpeta bin, que almacena los scripts y archivos necesarios para que Tomcat funcione. Apache Tomcat utiliza un script llamado **Catalina** para gestionar el servicio. **Catalina.sh** permite iniciar o parar el servicio, comprobar la configuración, o mostrar la versión.

```
root@lnxsrv-tomcat-01:~# /usr/share/tomcat9/bin/catalina.sh version
Using CATALINA_BASE:   /usr/share/tomcat9
Using CATALINA_HOME:   /usr/share/tomcat9
Using CATALINA_TMPDIR: /usr/share/tomcat9/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /usr/share/tomcat9/bin/bootstrap.jar:/usr/share/tomcat9/bin/tomcat-juli.jar
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.rmi/sun.rmi.transport=ALL-UNNAMED
Server version: Apache Tomcat/9.0.37
Server built:   Jun 30 2020 20:09:49 UTC
Server number:  9.0.37.0
OS Name:        Linux
OS Version:     4.15.0-109-generic
Architecture:   amd64
JVM Version:    11.0.7+10-post-Ubuntu-2ubuntu218.04
JVM Vendor:     Ubuntu
```

Código 2. Resultado de la ejecución del comando catalina.sh version en Ubuntu.

/ 4. Apache Tomcat: Logs y SSL

- **Logs:** Como ya sabemos, cualquier servicio utiliza los logs para almacenar un listado de los eventos que han ocurrido. Por defecto, los registros de Apache Tomcat se almacenan en la carpeta /usr/share/tomcat/logs y entre todos los ficheros que hay, encontraremos los siguientes:
  - **Catalina.out:** es el archivo principal y almacena errores y eventos producidos en el servidor.
  - **Catalina.*.log:** generará un archivo diario con los eventos almacenados en catalina.out.
  - **Localhost_access.*.txt:** log de acceso. Almacena la IP de los clientes que han solicitado información al servidor.

```
cat /var/log/tomcat9/localhost_access_log.2020-07-10.txt
192.168.1.10 - - [10/Jul/2020:16:40:48 +0000] "GET / HTTP/1.1" 200 1895
192.168.1.10 - - [10/Jul/2020:16:40:48 +0000] "GET /favicon.ico HTTP/1.1" 404 1022
192.168.1.10 - - [10/Jul/2020:16:40:54 +0000] "GET /docs/ HTTP/1.1" 200
```

Código 3. Ejemplo de access log.

En el momento de creación de esta documentación, la versión más reciente de Tomcat es la 9, y asumimos que las rutas de los ficheros de configuración o de los registros podrían variar dependiendo de la versión que tengamos instalada.

- **SSL Apache Tomcat:** Al igual que ocurre con cualquier servidor web, para cifrar la conexión utilizando el protocolo HTTPS, necesitaremos un certificado SSL.

Los certificados SSL se guardan en un almacén llamado **keystore** utilizando el comando **keytool.**

Una vez que tengamos creado el almacén y hayamos importado el certificado, podremos modificar el archivo **server.xml** y activar un conector.

```
<Connector
    protocol="org.apache.coyote.http11.Http11NioProtocol"
    port="443" maxThreads="200"
    scheme="https" secure="true" SSLEnabled="true"
    keystoreFile="/root/.tomcatKeystore" keystorePass="usuario"
    clientAuth="false" sslProtocol="TLS"/>
```

Código 4. Conector de Apache Tomcat configurado para escuchar por el puerto 443 y usar HTTPS.

En el vídeo 1 de la unidad, realizaremos la configuración paso a paso para activar HTTPS en un servidor de aplicaciones Apache Tomcat. Además, podemos ampliar la información sobre el uso de certificados SSL en la web de Apache Tomcat:

https://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html

Vídeo 1. "Activar HTTPs en Apache Tomcat"
https://bit.ly/3jY7d1Z

/ 5. Caso práctico 1: “Acceso a Host Manager de Tomcat”

Planteamiento: Isabel está desarrollando una aplicación web para un cliente sobre Ubuntu 18.04 con Tomcat v9.

Al intentar entrar en el gestor de aplicaciones web de Tomcat, le devuelve el siguiente error:

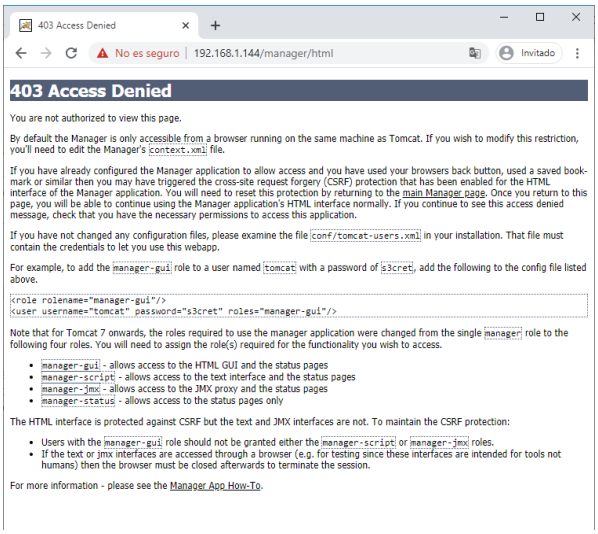

Fig. 4. Error accediendo a Host Manager de Tomcat.

Nudo: ¿Qué debe hacer para resolverlo?

Desenlace: La respuesta que recibe Isabel no es un error, es un feature (característica) de Tomcat para protegerse de accesos indeseados. Por defecto permite acceder desde la IP del propio servidor.

Para darnos acceso, editaremos el fichero context.xml (~/webapps/manager/META-INF/context.xml), añadiendo a la línea “allow=127\...”, nuestra dirección IP.

```
nano /usr/share/tomcat9/webapps/manager/META-INF/context.xml
<Context antiResourceLocking=”false” privileged=”true” >
  <Valve className=”org.apache.catalina.valves.RemoteAddrValve”
        allow=”127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1|IP_PERMITIDA” />

  <ManagersessionAttributeValueClassNameFilter=”java\.lang\.(?:Boolean|Integer|Long|Number|String)|org\.
apache\.catali$</Context>
```

Código 5. Error accediendo a Host Manager de Tomcat.

A continuación, reiniciamos Tomcat, al que podremos acceder siempre y cuando, previamente, hayamos configurado el fichero tomcat-users.xml.

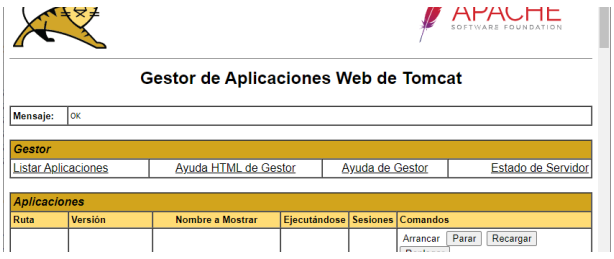

Fig. 5. Gestor de aplicaciones web de Tomcat

/ 6. Configuración Node.js. MongoDB

Uno de los puntos fuertes de Node.js es su escalabilidad, y la mejor forma de demostrar este factor es que no requiere de ficheros de configuración en el servidor para su funcionamiento.

La configuración de cada proyecto de Node.js se gestionará directamente desde el fichero app.js. En este fichero, se definen los parámetros de cada aplicación, cómo será, el puerto por el que escucha o el protocolo utilizado.

Las librerías de Node.js se almacenan en /usr/libr/nodejs.

Cada módulo tendrá un fichero package.json con el descriptor de despliegue necesario para su funcionamiento.

```
sudo cat /usr/lib/node_modules/npm/node_modules/http-signature/package.json
{
 "_from": "http-signature@~1.2.0",
 "_id": "http-signature@1.2.0",
 "_inBundle": false,
 "_integrity": "sha1-muzZJRFHcvPZW2WmCruPfBj7rOE=",
 "_location": "/http-signature",
 "_phantomChildren": {},
 "_requested": {
  "type": "range",
  "registry": true,
  "raw": "http-signature@~1.2.0",
  "name": "http-signature",
  "escapedName": "http-signature",
  "rawSpec": "~1.2.0",
  "saveSpec": null,
  "fetchSpec": "~1.2.0"
```

Código 6. Comienzo del descriptor de despliegue de un módulo de Node.js.

MongoDB

Actualmente, mongoDB es uno de los sistemas gestores de bases de datos no relacionales más utilizados.

La unión de mongoDB con Node.js permite acceder a la información de forma rápida y fiable, siendo una opción más practica en entornos que buscan un alto rendimiento.

Audio 1. "MongoDB"
https://bit.ly/30X6W6H

/ 7. Node.js y SSL

En este apartado, documentaremos los procedimientos necesarios para activar HTTPS en un proyecto que se ejecute sobre Node.js.

Crearemos una aplicación básica sin JavaScript que solo muestre "hola mundo", puesto que lo interesante es aprender a configurar un sitio sobre SSL.

Como en cualquier proyecto en el que queramos utilizar el protocolo HTTPS, lo primero que tendremos que tener preparado son los certificados del servidor.

En el caso de querer utilizar certificados autofirmados, podemos crearlos con el siguiente comando:

```
openssl genrsa -out key.pem
openssl req -new -key key.pem -out csr.pem
openssl x509 -req -days 9999 -in csr.pem -signkey key.pem -out cert.pem
```

Código 7. Creación de certificados. Extracto de:

https://nodejs.org/en/knowledge/HTTP/servers/how-to-create-a-HTTPS-server/

Es muy importante que estos certificados estén en la misma ruta que la aplicación web o esta no funcionará.

El siguiente paso será crear la aplicación en Node.js declarando las variables necesarias para forzar el uso de HTTPS y cargar los certificados que acabamos de crear.

Para ello, crearemos una carpeta para el nuevo proyecto y dentro inicializaremos un fichero app.js con el siguiente contenido:

```
1  const https = require('https');
2  const fs = require('fs');
3  const options = {
4    key: fs.readFileSync('./key.pem'),
5    cert: fs.readFileSync('./cert.pem')
6  };
7  https.createServer(options, function (req, res) {
8    res.writeHead(200);
9    res.end("hello world\n");
10 }).listen(443);
```

Código 8. Creación de certificados. Extracto de:

https://nodejs.org/en/knowledge/HTTP/servers/how-to-create-a-HTTPS-server/

A continuación, lanzamos la aplicación con el comando node:

```
node app.js
```

Código 9. Creación de certificados. Extracto de:https://nodejs.org/en/knowledge/HTTP/servers/how-to-create-a-HTTPS-server/

Y la vista en el navegador sería la siguiente:

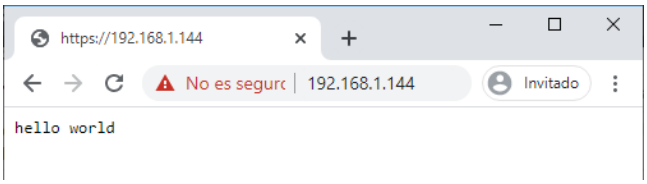

Fig. 6. Creación de certificados. Extracto de:
https://nodejs.org/en/knowledge/HTTP/servers/how-to-create-a-HTTPS-server/

Vídeo 2. "HTTPS y 443 en Node.js"
https://bit.ly/33iOEj9

/ 8. Caso práctico 2: "Despliegue de Node.js en Docker"

Planteamiento: Andrés trabaja en el departamento de sistemas de una empresa que ofrece servicios webs. Uno de los desarrolladores acaba de reportar una incidencia indicando que la versión de Node.js que tiene instalada en el servidor está anticuada y necesita actualizarla. El servidor es un Ubuntu 18.04 LTS y Node está instalado utilizando el comando apt-get install Node.js.

Nudo: ¿Qué ha podido pasar? ¿Cómo comprobará la versión de Node.js que utiliza el servidor? ¿Qué pasos seguirá para actualizarlo?

Desenlace: Tras realizar una búsqueda por internet, Andrés se ha percatado que utilizar los repositorios de Ubuntu para descargar e instalar Node.js puede provocar este fallo. Lo primero que hace es ejecutar el comando node –v para comprobar la versión que tiene instalada.

```
node -v
v8.10.0
npm -v
3.5.2
```

Código 10. Comprobación de versiones instaladas de node.js y de npm.

A continuación, desinstalará Node.js del sistema y cargará los repositorios oficiales de node en el sistema operativo.

```
sudo apt-get remove nodejs npm
```

Código 11. Desinstalación de nodejs.

```
curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -

## Installing the NodeSource Node.js 12.x repo...
. . .
```

Código 12. Instalación de repositorio oficial de Node.js.

```
sudo apt-get install nodejs
```

Código 13. Instalación de Node.js.

Una vez terminada la actualización, volverá a comprobar las versiones y ya podrá avisar al desarrollador.

```
node -v
v12.18.2
npm -v
6.14.5
```

Código 14. Comprobación de versiones instaladas después de la actualización.

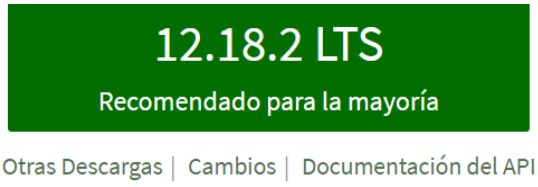

Fig. 7. Versión recomendada por Node.js. Fuente: https://nodejs.org

/ 9. Resumen y resolución del caso práctico de la unidad

Es esta unidad, hemos continuado aprendiendo sobre los **servidores de aplicaciones:** arquitectura básica, formas de funcionamiento y configuraciones.

Principalmente, nos hemos adentrado en la **configuración de aplicaciones seguras** con Apache Tomcat y Node.js, así como en el cambio de los puertos por defecto de ambos servidores.

Resolución del caso práctico de la unidad

En el caso práctico inicial se plantea la siguiente situación:

Al intentar lanzar una aplicación escrita en Node.js, no es posible iniciar la aplicación en el puerto solicitado, debido a que ese puerto ya está en uso.

La solución más rápida para solventar esto, es cambiar el puerto de escucha de la aplicación Node.js siempre que sea posible, en caso contrario, se tendrá que realizar la instalación sobre un nuevo servidor.

Para cambiar el puerto de una aplicación Node.js, tendremos que modificar el fichero *.js de la aplicación cambiando la siguiente línea:

```
}).listen(PUERTO);
```

Código 15. Configuración de aplicación Node.js. Cambiar el puerto de escucha.

/ 10. Webgrafía

Colaboradores de Wikipedia. (2020, 7 marzo). Servidor de aplicaciones. Recuperado de https://es.wikipedia.org/wiki/Servidor_de_aplicaciones#:%7E:text=Caracter%C3%ADsticas%20comunes,-Los%20servidores%20de&text=Los%20servidores%20de%20aplicaci%C3%B3n%20tambi%C3%A9n%20brindan%20soporte%20a%20una%20gran,de%20datos%2C%20sistemas%20y%20dispositivos.

Wikipedia contributors. (2020, 10 julio). List of application servers. Recuperado de https://en.wikipedia.org/wiki/List_of_application_servers

Marshal, A. (2019, 17 enero). Top 10 Open Source Java and JavaEE Application Servers. Recuperado de https://blog.idrsolutions.com/2015/04/top-10-open-source-java-and-javaee-application-servers/

Syed, B. (2014). Beginning Node.js. Recuperado de https://github.com/chensokheng123/allfile/blob/master/Basarat%20Ali%20Syed%20-%20Beginning%20Node.js-Apress%20(2014).pdf

Cómo escribir código asíncrono en Node.js | DigitalOcean. (2020, 19 marzo). Recuperado de https://www.digitalocean.com/community/tutorials/how-to-write-asynchronous-code-in-node-js-es

Span Class=. (2020, 8 enero). Cómo crear una aplicación Node.js con Docker. | DigitalOcean. Recuperado de https://www.digitalocean.com/community/tutorials/como-crear-una-aplicacion-node-js-con-docker-es

N. (2011, 26 agosto). How to create an https server? Recuperado de https://nodejs.org/en/knowledge/HTTP/servers/how-to-create-a-HTTPS-server/

Álvarez, C. (2014, 9 julio). ¿Cómo funciona Node.js? Recuperado de https://www.genbeta.com/desarrollo/como-funciona-node-js