MEDAC
Instituto Oficial de Formación Profesional

DESPLIEGUE DE APLICACIONES WEB
TÉCNICO EN DESARROLLO DE APLICACIONES WEB

Introducción a los servidores de aplicación

06

ÍNDICE

OBJETIVOS

Conocer los servidores de aplicaciones.

Conocer los fundamentos y protocolos que utiliza un servidor de aplicaciones.

Clasificar los servidores de aplicaciones.

Instalar y configurar servidores de aplicaciones.

Empaquetar aplicaciones.

Desplegar aplicaciones web con Docker.

/ 1. Introducción y contextualización práctica

En este tema, hablaremos de los servidores de aplicaciones, cómo funcionan y para qué sirven. Analizaremos las características básicas de un servidor de aplicaciones y compararemos las funcionalidades que aportan frente a un servidor web.

Además, aprenderemos a configurar servidores de aplicaciones. Realizaremos, también, despliegues de aplicaciones web con Docker.

Por otro lado, comentaremos qué compone una aplicación web y realizaremos un despliegue básico.

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. Servidores de aplicaciones web.

Audio intro. "Alojar una aplicación web JavaScript"
https://bit.ly/39BnHbk

/ 2. ¿Qué son los servidores de aplicaciones?

Del mismo modo que los servidores webs alojan páginas estáticas o dinámicas, los servidores de aplicaciones se encargan de **albergar aplicaciones desarrolladas en un determinado lenguaje**, comunicándose con otros elementos de la red, para **entregar al cliente final el contenido solicitado.** En otras palabras, un servidor de aplicaciones es el software intermedio (middleware) que permite la ejecución de código y estructura los datos para poder visualizarlo en un navegador web. También llamamos así al equipo que realiza estas funciones.

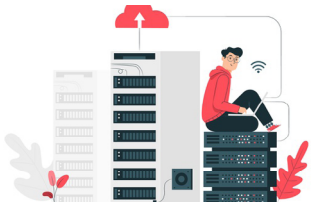

Fig. 2. Servidores de aplicaciones.

Además de ejecutar código escrito en lenguajes, como Java o JavaScript, el middleware se comunica con un servidor de bases de datos (que podría estar en el mismo servidor o no) para resolver las peticiones y entregar el contenido sin necesidad de almacenar información en el cliente.

Los servidores de aplicaciones aparecen ante la necesidad de desarrollar aplicaciones en otros lenguajes que permitan ampliar las funciones de un servidor web.

2.1. Diferencias entre un servidor web y un servidor de aplicaciones

Un servidor web está diseñado para **servir contenido estático** (HTML, CSS, etc.) que se ejecutará en el lado del cliente. En el caso de contenido dinámico, como por ejemplo PHP, el servidor necesitará un software adicional que le permita realizar estas funciones.

En el caso de JavaScript, es un lenguaje de programación que se conoce más bien por encontrarse en el lado del cliente, aunque cada vez más se usa en el lado del servidor, utilizando la herramienta de Node.js

Sin embargo, un servidor de aplicaciones contiene todo el software necesario para ejecutar código en el servidor y entregar contenido a cualquier navegador web.

Aunque un servidor de aplicaciones es capaz de servir sitios webs de manera autónoma, no es raro encontrarlo trabajando en simbiosis con un servidor web.

Audio 1. "Java Runtine Environment"
https://bit.ly/3g97psQ

/ 3. Servidores de aplicaciones. Funcionamiento y tipos

Un servidor de aplicaciones organiza la información y los servicios alojados de manera similar a como lo hace un servidor web. Por ejemplo, en el caso de Apache, sabemos que utiliza Virtual Hosts para poder alojar varios sitios webs. En el caso de los servidores de aplicaciones, se utilizarán contenedores para gestionar los servicios hospedados.

Cada contenedor albergará una aplicación web, y cada una de ellas escuchará las peticiones por un puerto diferente. De este modo, podremos implementar servidores de aplicaciones "multisitio".

En el esquema clásico de arquitectura web, podremos encontrar un servidor de aplicaciones y un servidor de base de datos que se comunicarán para servir las peticiones a los clientes. Aunque los servidores de aplicaciones por sí mismos pueden gestionar peticiones HTTP, no es extraño encontrar un servidor web que redirija las peticiones al servidor de aplicaciones actuando como proxy.

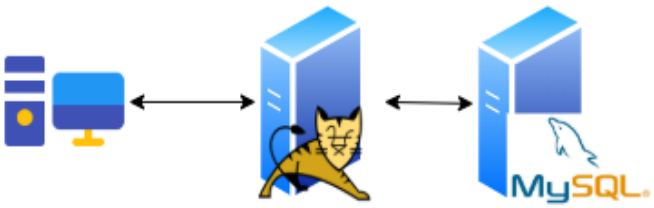

Fig. 3. Esquema de funcionamiento de un servidor de aplicaciones.

3.1. Tipos de servidores de aplicaciones: Libres y propietarios

Como en cualquier software, podemos encontrar dos tipos de licencias: licencia comercial y licencia Open Source.

A continuación, listaremos algunos de los servidores de aplicaciones que podemos encontrar diferenciados en dos grupos: servidor de aplicaciones Java y servidor de aplicaciones JavaScript.

- **Java:**

Producto	Desarrollador	Compatibilidad JAVA	Licencia
Tomcat	Apache Software fundation	8 partial platform	Free, Apache v2
WebLogic Server	Oracle Corporation	8 full platform	Proprietary, commercial
WildFly	Red Hat	8 full platform	Free, LGPL
IBM WebSphere Application Server	IBM	8 full platform	Proprietary, commercial

Tabla 1. Comparativa de servidor de aplicaciones Java. Extracto de: https://en.wikipedia.org/wiki/List_of_application_servers

- **JavaScript**

OPERADOR	OPERACIÓN	Compatibilidad JAVA	Licencia
Node.js	Apache Software fundation	8 partial platform	Free, Licencia MIT
Wakanda	Wakanda SAS	8 full platform	Community / Enterprise / Cloud

Tabla 2. Comparativa de servidor de aplicaciones Java. Extracto de: https://en.wikipedia.org/wiki/List_of_application_servers

/ 4. Apache Tomcat

Apache Tomcat es un servidor de aplicaciones Open Source liberado en 1999. Tomcat realiza funciones de servidor web Apache y también es un contenedor de Servlets y JSP (Java Server Pages).

Fig. 4. Logo Apache Tomcat.

Cuando hablamos de contenedor de Servlets y JSP, nos referimos al entorno de ejecución incluido en Apache Tomcat, que permite a los clientes realizar llamadas y ejecución de programas alojados en este servidor.

Tomcat tiene dos modos de funcionamiento:

- **Contenedores independientes:** Apache Tomcat funciona como servidor de aplicaciones y servidor web de forma autónoma.
- **Coexistencia con un servidor web:** Apache Tomcat se limita a ejecutar los Servetls, mientras que el servidor web se encarga de servir el contenido solicitado a Tomcat.

En el caso de funcionar de manera conjunta con un servidor web, el proceso es el siguiente:

- El servidor web carga las librerías necesarias para comunicarse con Tomcat.
- Al recibir una petición, evalúa si se corresponde a un Servlet o JSP y, si procede, reenvía la petición al servidor de aplicaciones.
- El servidor Tomcat procesa la petición y devuelve el resultado al servidor web, que, a su vez, responde la petición del cliente.

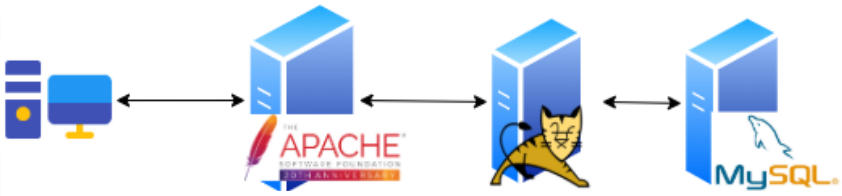

Fig. 5. Esquema de funcionamiento Tomcat con Apache.

Apache Tomcat entra en el grupo de middleware, ya que incluye todo lo necesario para comunicarse con el sistema operativo y el resto de componentes de la arquitectura, como podría ser un servidor de BBDD.

Al estar escrito en Java, Apache Tomcat es multiplataforma, y se puede instalar en cualquier S.O. El único requisito para poder utilizarlo es tener instalada JRE (Java Runtine Envioremnt).

Vídeo 1. "Instalación Tomcat en Ubuntu"
https://bit.ly/3gbeljH

/ 5. Caso práctico 1: “Instalación Tomcat en Windows”

Planteamiento: Luis está pensando en desarrollar una aplicación web en Java y necesita un servidor para poner en marcha su idea. Actualmente, tiene pocos recursos y solo dispone de un equipo con Windows 10. Se ha decidido por Apache Tomcat porque es compatible con su sistema operativo.

Nudo: ¿Qué pasos seguirá para instalar un servidor Tomcat en Windows? ¿Qué requisitos tiene Tomcat en Windows?

Desenlace: Para instalar y configurar Apache Tomcat, el único requisito es tener instalado JDK (Java Development Kit). Este paquete se puede descargar desde la web de Oracle.

Fig. 6. Instalación de Java.
https://www.java.com/es/download/manual.jsp

Una vez cumplido este requisito, podrá descargar Apache Tomcat desde la web oficial y ejecutar el paquete binario para Windows.

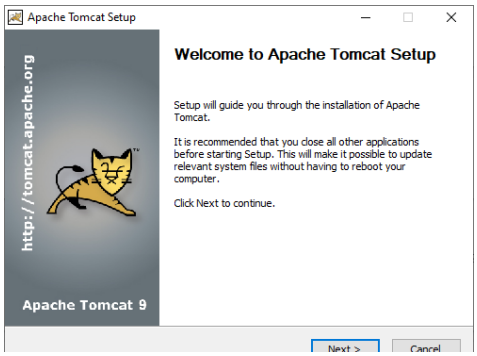

Fig. 7. Instalación Apache Tomcat en Windows.
https://tomcat.apache.org/download-90.cgi

Cuando concluya la instalación, podremos acceder a través de un navegador web en la siguiente dirección http://localhost:8080

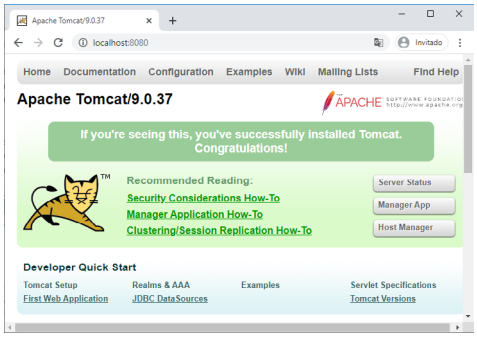

Fig. 8. Index de Apache Tomcat.

/ 6. Aplicaciones web: Estructura

Al igual que un sitio web, una aplicación web está compuesta por una serie de archivos que permiten su correcto funcionamiento. Estos ficheros pueden ser Servlets, JSP, HTML, CSS, etc.

En el caso de las aplicaciones webs, además, podemos agruparlas en un único archivo y desplegarlas en otros servidores.

La estructura de las aplicaciones web está estandarizada de manera que permita su despliegue en diferentes servidores sin necesidad de instalar software adicional como librerías en cada uno de ellos. Todos los archivos, contenido multimedia y librerías se pueden comprimir en un único archivo con extensión WAR (Web Application Archive). A este proceso de compresión de todos los archivos y carpetas necesarias en un único fichero se le conoce como **empaquetado.**

La estructura básica de un fichero WAR es la siguiente:

- **Carpeta raíz (/):** almacenan los ficheros típicos de un sitio web como documentos HTML, CSS, y los programas, en JSP.
- **/WEB-INF:** contiene las librerías de la aplicación y el descriptor de despliegue.
- **/WEB-INF/classes:** contiene las clases y los Servlets.
- **/WEB-INF/lib:** contiene las librerías necesarias para conectar con la base de datos.
- **/META-INF:** contiene el manifiesto de la aplicación.

```
sample.war
  |  hello.jsp
  |  index.html
  |___images
  |     tomcat.gif
  |___META-INF
  |     MANIFEST.MF
  |___WEB-INF
     |  web.xml
     |___classes
     |  _____mypackage
     |        Hello.class
     |___lib
```

Código 1. Ejemplo de aplicación web empaquetada en formato WAR.

No debemos confundir los archivos WAR con los JAR de Java, ya que estos últimos solo almacenan los archivos compilados, mientras que WAR incorpora todo lo necesario para desplegar una aplicación web.

Veamos, a continuación y en mayor profundidad, algunos de los conceptos estudiados en este apartado, así como una serie de elementos relacionados con las aplicaciones web.

/ 7. Aplicaciones web: Componentes

7.1. Entornos de desarrollo integrado

A la hora de hacer cualquier desarrollo, ya sea una aplicación web, un sitio estático escrito con HTML o una hoja de estilo CSS, podemos ayudarnos de programas que nos facilitan esta tarea. Estos programas son los **entornos de desarrollo integrado o IDE** (*Integrated Development Environment*).

Una de las ventajas de utilizar un IDE es que al incorporar intérpretes de lenguaje de programación, nos avisa si cometemos errores de sintaxis y nos indica dónde está el error que no permite la compilación del programa. Entre los entornos de desarrollo más utilizados, encontraremos NetBeans o Eclipse Java EE.

7.2. Empaquetado de una aplicación web

A la hora de preparar una aplicación web para su despliegue, tendremos que empaquetarla en un archivo .WAR. Para ello, utilizaremos un entorno de desarrollo para compilar nuestro proyecto y exportarlo a un archivo apto para servidores de aplicaciones

En el siguiente QR encontrarás un vídeo de YouTube en el que verás cómo empaquetar una aplicación web con NetBeans.

7.3. Descriptor de despliegue

Los descriptores de despliegue son archivos en formato XML que describen los parámetros necesarios para el despliegue de una aplicación web. Un descriptor de despliegue se puede modificar sin tener que cambiar el código fuente de la aplicación. Como vimos, el descriptor de despliegue se almacena en la carpeta WEB-INF y se llama web.xml. Este fichero es un estándar compatible con cualquier servidor de aplicaciones Java EE.A continuación, analizaremos un descriptor de despliegue de Tomcat.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.
xsd"
    version="2.4">

    <display-name>Hello, World Application</display-name>
    <description>
        This is a simple web application with a source code organization
        based on the recommendations of the Application Developer's Guide.
    </description>

    <servlet>
      <servlet-name>HelloServlet</servlet-name>
      <servlet-class>mypackage.Hello</servlet-class>
    </servlet>

    <servlet-mapping>
      <servlet-name>HelloServlet</servlet-name>
      <url-pattern>/hello</url-pattern>
    </servlet-mapping>
</web-app>
```

Código 2. Ejemplo de un descriptor de despliegue. Sample.war extraído de: https://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/sample.war

/ 8. Node.JS

Node.js es un servidor de aplicaciones que nace como un entorno de ejecución de JavaScript. Node.js utiliza el motor de JavaScript V8 de Google y utiliza un modelo de entrada-salida asíncrono controlado por eventos, convirtiendo Node.js en un runtime liviano y de alto rendimiento.

¿Qué es un modelo entrada-salida (I/O) asíncrono?

Los modelos asíncronos o non-blocking no ejecutan el código en el mismo orden que fue escrito y no esperan a que una operación de lectura/escritura termine para pasar a la siguiente línea. Es decir, dejan que el sistema operativo la dé por finalizada sin mostrar resultados.

En el siguiente enlace, podrás ampliar conocimientos sobre los modelos de I/O en Node.js

https://bytearcher.com/articles/blocking-vs-non-blocking-in-node.js/

Node.js se presenta como un sistema propicio para desarrollar sistemas escalables argumentando que es inmune a los bloqueos provocados por altas cargas de trabajo, ya que casi ninguna función de Node.js realiza I/O directamente.

Node.js no utilizará un descriptor de despliegue en formato XML, en cambio, utilizará un archivo llamado package.json donde listaremos las dependencias que requiera nuestra aplicación.

No se recomienda instalar Node.js desde los repositorios de Ubuntu, pues estos podrían estar desactualizados; en su lugar, es mejor instalar el repositorio oficial de Node.js:

```
curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Código 3. Código node.js para "hola mundo". Extracto de developer.mozilla.org:

https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/development_environment

Al escribir programas en Node.js, declaramos variables como el host y puerto, pudiendo crear tantos programas como puertos podamos utilizar.

```
const http = require('http');
const hostname = '127.0.0.1';
const port = 3000;
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hola Mundo');
});
server.listen(port, hostname, () => {
  console.log(`El servidor se está ejecutando en http://${hostname}:${port}/`);
});
```

Código 4. Código node.js para "hola mundo". Extracto de la web de node.js https://nodejs.org/es/about/

Encontrarás más información sobre Node.js en la web oficial: https://nodejs.org/es/about/

/ 9. Caso práctico 2: “Despliegue de Node.js en Docker”

Planteamiento: Gloria trabaja en el departamento de sistemas de una franquicia de comida rápida. Tiene que implementar diez instancias de una aplicación web que ya tiene desarrollada y está buscando la forma más rápida de realizar el despliegue. Ha pensado en Docker para empaquetar su aplicación escrita con Node.js y realizar un despliegue masivo.

Nudo: ¿Es posible? ¿Cómo lo hará?

Desenlace: Para empaquetar la aplicación utilizando Docker, Gloria deberá instalar el paquete adecuado para su sistema operativo. Puede ayudarse de las guías de la página oficial. En el siguiente apartado, profundizaremos, además, sobre ello:

https://docs.docker.com/engine/install/

Una vez instalado, el siguiente paso es situarnos en la carpeta del proyecto donde Gloria tendrá el descriptor de despliegue package.json. Utilizaremos el comando npm para descargar las librerías y dependencias necesarias.

```
cd ruta_proyecto/
npm install
npm notice created a lockfile as package-lock.json. You should commit this file.
added 50 packages from 37 contributors and audited 50 packages in 3.436s
found 0 vulnerabilities
```

Código 5. Docker + Node.js.

A continuación, crearemos un archivo Dockerfile indicando parámetros necesarios para el despliegue como la versión de node que utilizará las rutas, o archivos, que ejecutará.

```
nano Dockerfile
FROM node:12
. . .
EXPOSE 3000
CMD [ “node”, “server.js” ]
```

Código 6. Docker + Node.js. Extracto nodejs.org.

También, crearemos un archivo llamado .dockerignore indicando qué archivos no se copiarán a la imagen.

```
nano .dockerignore
```

Código 7. Docker + Node.js. Extracto nodejs.org.

Por último, crearemos la imagen de la aplicación con el comando docker build.

```
sudo docker build -t test/app-node-02 .
Sending build context to Docker daemon  19.97kB
Step 1/7 : FROM node:12
12: Pulling from library/node
81fc19181915: Pull complete
. . .
```

Código 8. Docker + Node.js.

Al finalizar, podremos comprobar que la imagen se ha creado correctamente y desplegar una instancia con docker run:

```
sudo docker images
REPOSITORY        TAG     IMAGE ID        CREATED          SIZE
test/app-node-02  latest  5352571a3eb6    7 seconds ago    920MB

sudo docker run -p 3001:3000 -d test/app-node-02
```

Código 9. Docker + Node.js.

/ 10. Desplegar aplicaciones web con Docker

Como vimos en temas anteriores, los contenedores nos permiten desplegar en un mismo host varios servicios de forma rápida y ligera. Hasta el momento, lo hemos utilizado para levantar servidores webs y poder alojar varios sitios en nuestro servidor de Docker.

En este punto, veremos cómo utilizar los contenedores para desplegar aplicaciones webs previamente creadas utilizando Node.js y Docker. Algunas de las ventajas de utilizar contenedores para desplegar aplicaciones web son las siguientes:

- **Escalabilidad:** permite crear nuevas instancias de una aplicación web en un corto período de tiempo.
- **Independencia de software:** son independientes de la versión del sistema operativo o de las versiones de librerías y dependencias disponibles en el sistema operativo.
- **Facilidad de mantenimiento:** al tener una imagen principal desplegada en los diferentes servidores, en caso de tener que actualizar algún componente, solo tendremos que realizar cambios una vez y volver a lanzar el proceso en el resto de máquinas.

En el siguiente enlace, encontrarás la documentación completa para poder probar a 'Dockerizar' una aplicación escrita en node.js.

https://www.digitalocean.com/community/tutorials/como-crear-una-aplicacion-node-js-con-docker-es

Y aquí podrás ver un ejemplo práctico para ejecutar una aplicación de Node dentro de un contendor.

Vídeo 2. Paquetizar aplicaciones con Node.js y Docker
https://bit.ly/334EfXX

/ 11. Resumen y resolución del caso práctico de la unidad

En esta unidad, hemos visto qué son y para qué sirven los **servidores de aplicaciones,** analizando las funcionalidades que aportan y comparándolos con los servidores webs.

Por otro lado, hemos profundizado sobre dos de los servidores de aplicaciones más comunes: **Apache Tomcat y Node.js.** También hemos conocido el proceso de **instalación y despliegue de aplicaciones sencillas.**

Finalmente, hemos aprendido **cómo utilizar Docker para realizar un despliegue masivo** de una aplicación escrita con Node.js.

Resolución del caso práctico de la unidad

En el caso práctico inicial se plantea una situación en la que Carmen debe pensar qué infraestructura utilizará para alojar una aplicación web escrita en JavaScript.

La solución más acertada será utilizar un servidor de aplicaciones Node.js que le permitirá ejecutar código JavaScript en el servidor y hacerla funcionar en un navegador web sin necesidad de instalar componentes o plugins.

/ 12. Webgrafía

https://es.wikipedia.org/wiki/Servidor_de_aplicaciones#:~: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.

https://en.wikipedia.org/wiki/List_of_application_servers

https://blog.idrsolutions.com/2015/04/top-10-open-source-java-and-javaee-application-servers/

https://idoc.pub/documents/cual-es-la-diferencia-entre-servidor-web-y-servidor-de-aplicaciones-datafull-3no7kp72dxld

https://www.digitalocean.com/community/tutorials/how-to-write-asynchronous-code-in-node-js-es

https://www.digitalocean.com/community/tutorials/como-crear-una-aplicacion-node-js-con-docker-es