Servidores web y servidores de aplicaciones

Álvaro González Sotillo

23 de septiembre de 2024

Índice

1. Introducción

- Una aplicación es una solución *software* para un problema
- Este *software* se ejecuta en un entorno que incluye
  - Otro *software* del que depende
  - *Hardware*
  - Conexiones de red
  - Otros servicios en ejecución

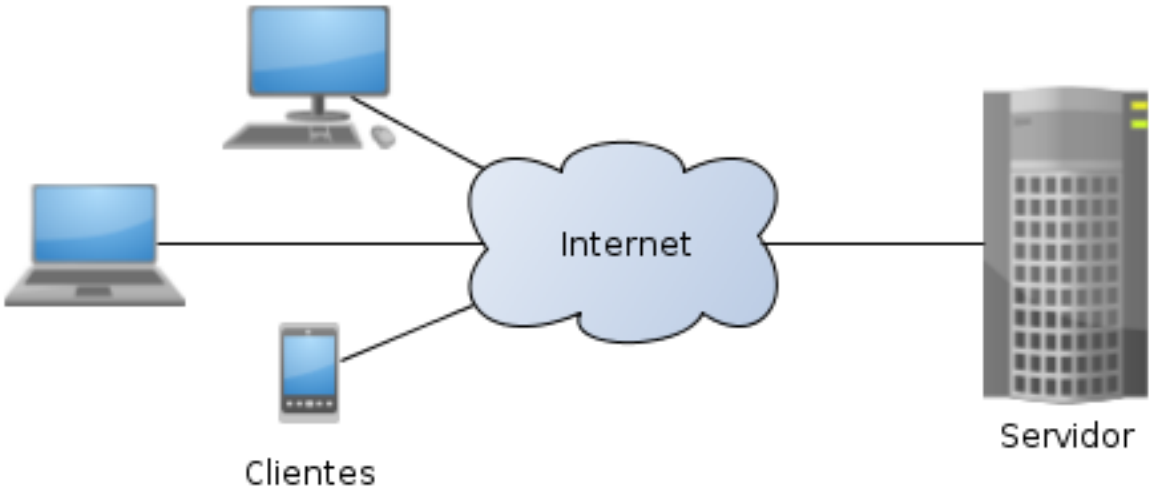
2. Aplicaciones web VS. escritorio

- Una aplicación de escritorio
  - Se lanza por el sistema operativo
  - Interactúa directamente con el resto del *software* y *hardware* del ordenador
  - Por tanto, suele ejecutarse en *workstations*
- Una aplicación web
  - Se ejecuta dentro de un navegador web
  - Interacción limitada con el resto del *software* y *hardware* del ordenador
  - Se ejecuta en un servidor web. Se ejecuta parcialmente en el navegador.

Pregunta: ¿es Firefox una aplicación web o de escritorio? Pregunta: ¿rollup es web o escritorio?

3. Arquitectura cliente-servidor

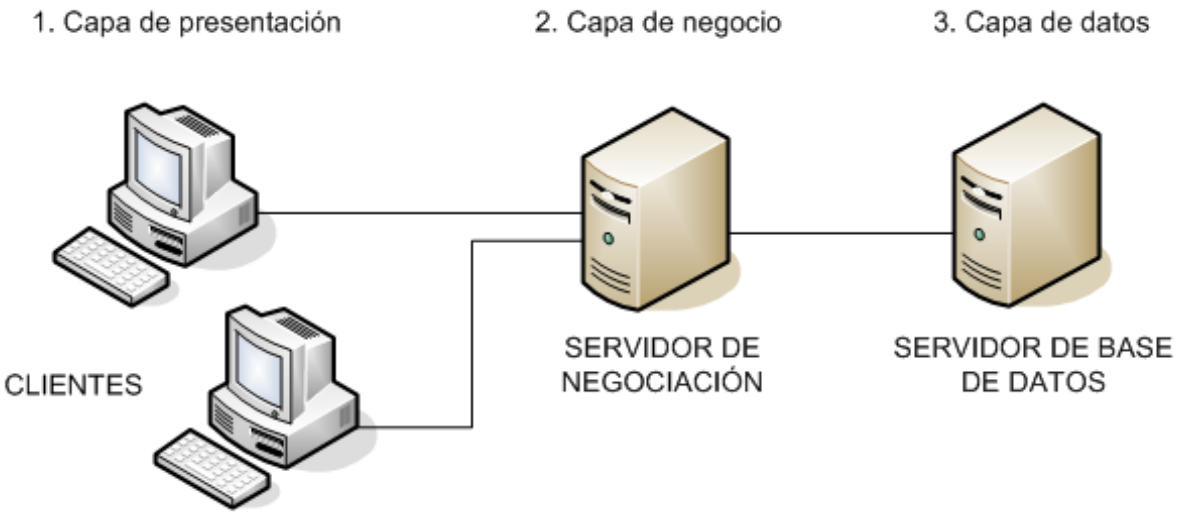
- La parte cliente permite al usuario interaccionar con la aplicación
- Parte de la lógica puede estar en el cliente, y parte en el servidor
- Los datos se almacenan en el servidor



Fuente: Wikipedia

4. Arquitectura de tres niveles

- El servidor se convierte en un servidor de negocio
  - Lógica de validación, usuarios, accesos, diferentes *pantallas*
- La base de datos es un servidor aparte



Fuente: Wikipedia

4.1. Arquitectura multicapa

- El negocio y la base de datos pueden a su vez dividirse en capas
- Las capas pueden estar en
  - Diferentes servidores físicos
  - Diferentes procesos en un servidor
  - Diferentes partes del código de la misma aplicación

5. Requisitos de la aplicación web

5.1. Del equipo servidor:

- procesador
- memoria
- almacenamiento
- tolerancia a fallos
- conexiones de red

5.2. Del sistema operativo anfitrión

- sistema de ficheros
- versiones
- componentes opcionales

5.3. De la propia aplicación

- tiempos de respuesta
- conexiones concurrentes
- niveles de acceso
- funcionalidad

6. Servidores web

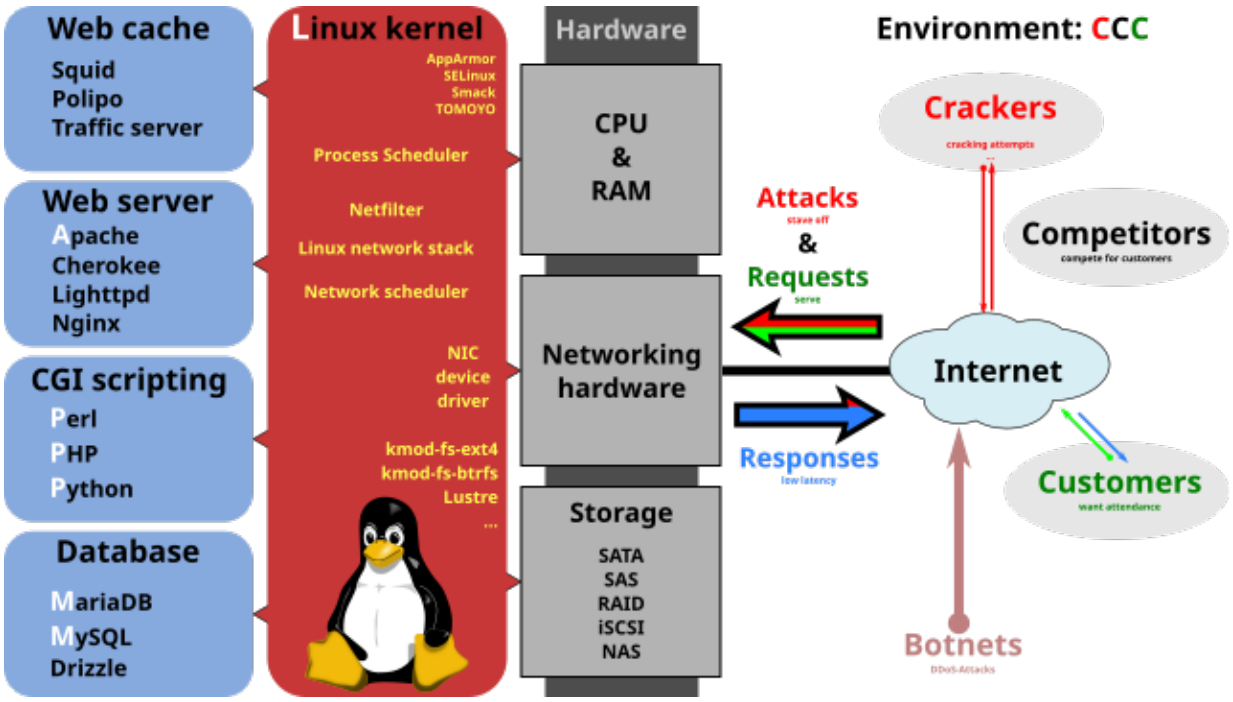
- Servidores web "puros"
  - Apache, nginx...
  - Sirven documentos con el protocolo HTTP/HTTPS
  - Los documentos son ficheros en disco
- Servidores de aplicaciones
  - IIS, Tomcat, Weblogic
  - Incluyen facilidades para generar el documento a devolver: ASP, JSP...
- Los servidores web "puros"pueden incluir *plugins* para poder generar documentos con lenguajes de *script*

6.1. Servidor de aplicaciones

- Los componentes de un servidor de aplicaciones son
  - Servidor web
  - Módulo de ejecución de *scripts*
  - Servidor de base de datos
  - Opcionales: *caché*, balanceador de carga, *firewall*

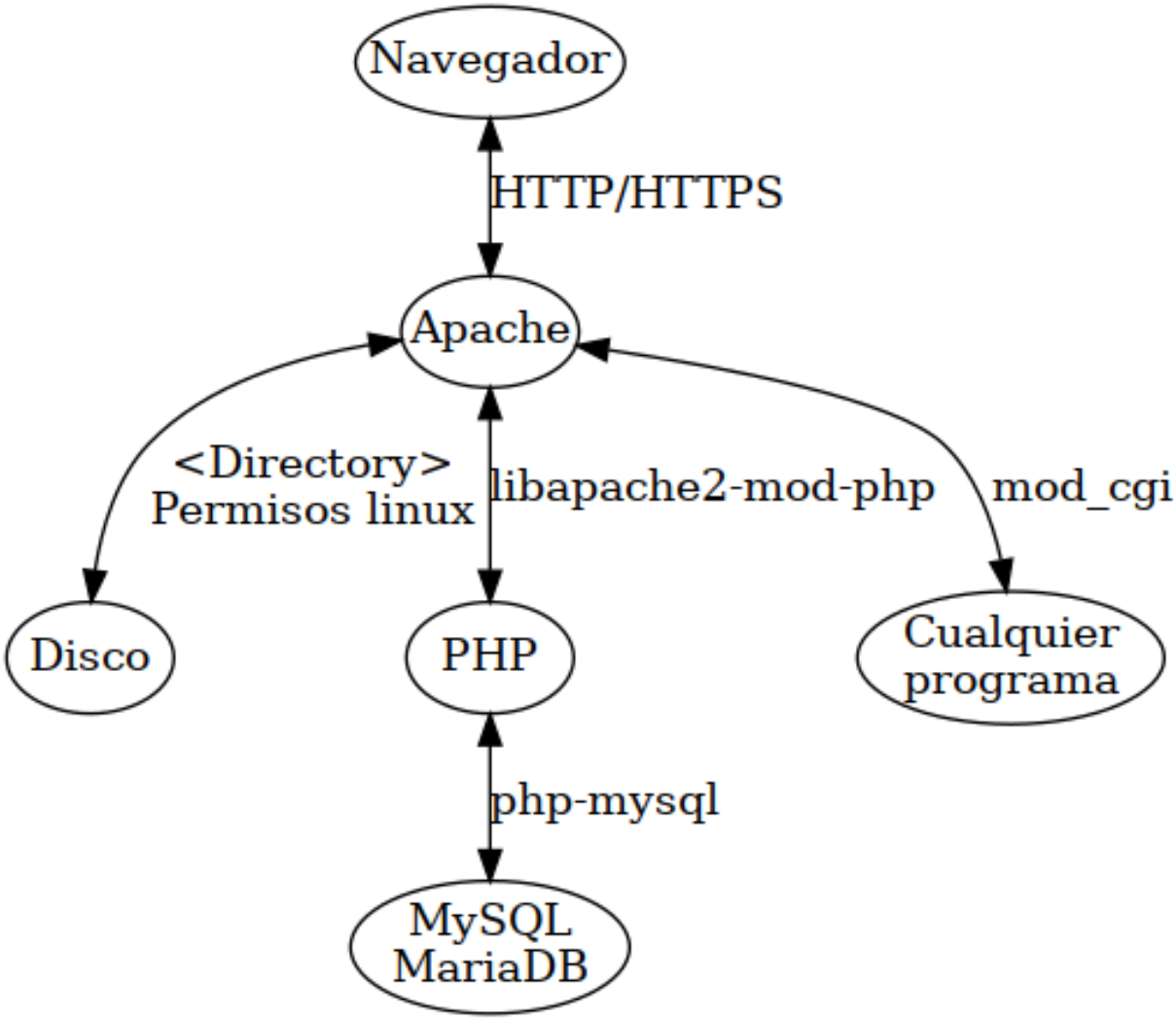
7. LAMP

- Linux, Apache, MySQL (o MariaDB), PHP
- Es un estándar *de facto* para aplicaciones web
  - Libre
  - Gratis
  - Muy probado
  - Rendimiento aceptable para proyectos pequeños y medianos



Fuente: Wikipedia

8. Instalación LAMP



9. Linux

- Impartido en otros módulos
- Usaremos Debian

10. Apache

```
sudo apt install apache2 -y
```

Tras instalar

```
sudo systemctl start apache2
sudo systemctl stop apache2
sudo systemctl restart apache2
sudo systemctl reload apache2
sudo systemctl status apache2
```

10.1. Configuración de Apache

- Directorio `/etc/apache2/`

- `apache2.conf`: Es el archivo de configuración principal. En este archivo se incluyen todos los archivos de configuración adicionales.
- `envvars`: Este archivo se definen las variables de entorno que hacen referencia al servidor web Apache y se utilizan en el archivo `apache2.conf`.
- `magic`: Este archivo contiene instrucciones para determinar el tipo de contenido o tipo MIME (MUltipurpose Internet Mail Extensions) de un archivo en función de los primeros bytes de un archivo. Los navegadores a menudo usan el tipo MIME (y no la extensión de archivo) para determinar cómo procesará un documento; por lo tanto, es importante que los servidores estén configurados correctamente para adjuntar el tipo MIME correcto al encabezado del objeto de respuesta. Puede encontrar más información sobre los tipo MIME aquí.
- `ports.conf`: En este archivo se definen los puertos TCP donde el servidor Apache estará escuchando peticiones.
- `conf-available`: Este directorio contiene archivos de configuración que se aplican a todos los hosts virtuales de forma global.
- `conf-enabled`: Este directorio contiene enlaces simbólicos a los archivos de configuración del directorio `conf-available` que están activos.
- `mods-available`: Este directorio contiene los archivos de configuración de los módulos que se pueden utilizar para añadir nuevas funcionalidades al servidor.
- `mods-enabled`: Este directorio contiene enlaces simbólicos a los archivos de configuración del directorio `mod-available` que están activos.
- `sites-available`: Este directorio contiene los archivos de configuración de los hosts virtuales.
- `sites-enabled`: Este directorio contiene enlaces simbólicos a los archivos de configuración del directorio `sites-available` que están activos.

10.2. *VirtualHost*

- Apache puede comportarse como diferentes servidores, según el nombre de servidor que ve el cliente
  - Cabecera HTTP Host `ServerName`
  - Hay otras formas de hacer *VirtualHosts*

```
<VirtualHost *:80>
    DocumentRoot /var/www/html
</VirtualHost>

<VirtualHost *:80>
    ServerName miservidor.local
    DocumentRoot /var/www/miservidor
</VirtualHost>
```

10.3. *Directory*

- Configura un directorio del disco (no del servidor web)
- Ejemplo que
  - Permite usar *links* a ficheros
  - Permite configurar cada directorio con un fichero .htaccess
  - Permite cualquier usuario de Apache
- Nota: El proceso de Apache debe poder acceder a ese directorio

```
<Directory /var/www/miservidor>
   Options Indexes FollowSymLinks
   AllowOverride All
   Require all granted
</Directory>
```

10.4. CGI

- *Common Gateway Interface*
- Configuración: `serve-cgi-bin.conf`
- Al recibir una petición:
  - Apache no devuelve el fichero, sino que lo ejecuta
  - El programa recibe como **entorno** información de la petición
  - El programa se encarga de generar la respuesta, incluidas las cabeceras

```
#!/usr/bin/bash

echo "Content-type: text/plain"
echo
echo

echo $*
env
```

10.5. Ejercicio CGI

- Consigue que el programa anterior funcione en Apache
- Examina las variables recibidas y su relación con las cabeceras HTTP de *request*

11. MariaDB/MySQL

```
apt install mariadb-server # o mysql-server
```

11.1. Hacer segura la instalación

- Comando `mysql_secure_installation`
- Elegir las opciones más seguras:

```
Enter current password for root (enter for none):
Switch to unix_socket authentication [Y/n] n
Change the root password? [Y/n] Y
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y
```

11.2. Usuarios

- El usuario de linux `root` puede acceder, con el usuario `root` de MariaDB
- Otros usuarios necesitan contraseña

```
create user alumno identified by 'XXXXXXXX';
```

```
mysql -u alumno -p
```

11.3. Comandos útiles MariaDB

```
SHOW DATABASES;
CREATE DATABASE <database>;
DROP DATABASE <database>;
USE <database>;
SHOW TABLES;
DESCRIBE <table>;
```

11.4. Privilegios MariaDB

- Sentencia estándar `grant` o `revoke`
- Se puede dar permisos sobre todas las tablas de una base de datos

```
grant all privileges on unabasededatos.*;
```

- Se pueden dar permisos sobre todas las bases de datos
- Se pueden dar permisos a usuarios dependiendo del origen de la conexión

```
GRANT ALL PRIVILEGES ON *.* TO 'nombre_usuario'@'localhost';
```

12. PHP

```
sudo apt install php
```

- La configuración de php está dividida según desde donde se use:
  - `/etc/php/7.4/cli`: Configuración de php para `php7.4-cli`, cuando se utiliza php desde la línea de comandos.
  - `/etc/php/7.4/apache2`: Configuración de php para apache2 cuando utiliza el módulo.
  - `/etc/php/7.4/mods-available`: Módulos disponibles de php que puedes estar configurados en cualquiera de los escenarios anteriores.
  - `/etc/php/7.4/apache2/conf.d`: Módulos instalados en esta configuración de php (enlaces simbólicos a `/etc/php/7.4/mods-available`).
  - `/etc/php/7.4/apache2/php.ini`: Configuración de php para este escenario.

12.1. Ejercicio CGI

- Consigue un programa en php que funcione como CGI
- El probrama debe usar la función `phpinfo()`

13. Interconexión entre componentes

```
sudo apt install libapache2-mod-php php-mysql
```

14. Ejercicio: phpmyadmin

- Instala phpmyadmin desde un paquete de la página de descargas https://www.phpmyadmin.net/downloads/
- Crea una base de datos de nombre `pruebamyadmin`, donde un usuario `myadmin` tenga todos los privilegios.
  - ¿qué es `php-mbstring`?

15. Qué había que saber antes de instalar LAMP

- `tail -F`
- `PATH`
- `locate`
- `ssh`
- `tmux`, o más de una ventana
- `/etc/hosts`

16. Alternativas a la instalación local

- https://profreehost.com
- infinityfree

17. aws :noexport

aws cli https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html#cliv2-linux-install

```
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install --bin-dir /usr/local/bin --install-dir /usr/local/aws-cli --update
```

18. Referencias

- Formatos:
  - Transparencias
  - PDF
  - Página web
  - EPUB
- Creado con:
  - Emacs
  - org-re-reveal
  - Latex
- Alojado en Github