Actividades. Implantación de arquitecturas web

Actividad 1: Análisis y fundamentos de arquitecturas web. (Criterios a, b, g, h)

Práctica 1: "Análisis Comparativo de Arquitecturas Web"

Descripción: Investigar y comparar diferentes arquitecturas (Monolítica, Cliente-Servidor, Microservicios, Serverless, SPA vs MPA).

Tareas:

- Crear una tabla comparativa con las características, ventajas e inconvenientes (Criterio a) de cada arquitectura.
- Investigar y explicar los protocolos fundamentales: HTTP/HTTPS, métodos (GET, POST), códigos de estado, cookies/sessions. Incluir el papel del DNS (Criterio b).
- Analizar la estructura de una aplicación web simple (por ejemplo, un WordPress o una app hecha en HTML, CSS, JS y PHP). Identificar los recursos: frontend, backend, base de datos, archivos estáticos (Criterio g).
- Elaborar una lista de requerimientos para implantar la aplicación analizada (necesidades de hardware, software, dependencias, permisos, dominio) (Criterio h).
- Entrega: Documento de análisis.

Actividad 2: Instalación, configuración básica y funcionamiento de servidores web. (Criterios c f)

Práctica 2.1: "Servidor Web con NGINX"

Objetivo

Familiarizarse con la instalación, configuración y uso básico de **NGINX** como servidor web para el despliegue de aplicaciones.

Tareas

1.- Comprueba la **disponibilidad del puerto 80**.
Antes de instalar NGINX, debemos asegurarnos de que el **puerto 80** no esté ocupado por otro servicio (por ejemplo, por **Apache**).

bash
sudo lsof -i :80

Si no aparece nada, significa que el puerto está libre.
Si aparece un servicio como `apache2` o `httpd`, significa que ya tienes el puerto ocupado y deberás detenerlo temporalmente para trabajar con NGINX o utilizar un puerto diferente:

bash
sudo systemctl stop apache2

2.- Instalación de NGINX.
En sistemas basados en **Debian/Ubuntu**:

bash
sudo apt update
sudo apt install nginx -y

3.- Verificación del servicio.
Comprueba que NGINX está en ejecución:

bash
systemctl status nginx

Abre un navegador y accede a `localhost` o a la dirección IP del servidor.

bash
http://localhost

Debería aparecer la página de bienvenida de NGINX.

4.- Estructura de archivos y directorios
Localiza los siguientes archivos de configuración:

- `/etc/nginx/nginx.conf` (configuración principal)
- `/etc/nginx/sites-available/` (sitios disponibles)
- `/etc/nginx/sites-enabled/` (sitios habilitados)

El contenido web por defecto se encuentra en:

- `/var/www/html`

5.- Crear una página web simple

Edita y personaliza el archivo de inicio.

html
<!DOCTYPE html>
<html>
<head>
  <title>Mi primer sitio con NGINX</title>
</head>
<body>

```
    <h1>¡Hola, mundo desde NGINX!</h1>
</body>
</html>
```

Refresca la página en el navegador para comprobar los cambios.

6.- Configurar un nuevo sitio web personal

Crea un directorio para el nuevo sitio:

bash
```
sudo mkdir -p /var/www/misitio
sudo nano /var/www/misitio/index.html
```

Inserta un HTML sencillo de bienvenida.

Crea un archivo de configuración:

bash
```
sudo nano /etc/nginx/sites-available/sitio-personal
```

Contenido básico:

bash
```
server {
    listen 80;
    server_name misitio.local;
    root /var/www/misitio;
    index index.html;
}
```

Activa el sitio:

bash
```
sudo ln -s /etc/nginx/sites-available/misitio /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

Añade la entrada en `/etc/hosts`:

bash
```
127.0.0.1 misitio.local
```

Accede al nuevo sitio.

bash
```
http://misitio.local
```

Tareas propuestas.

1.- Desplegar varias aplicaciones web
Crea al menos dos sitios adicionales, cada uno con su propia carpeta y configuración en NGINX.

2.- Configurar un redireccionamiento
Haz que las visitas a `http://misitio.local/blog` redirijan a otro sitio web externo (por ejemplo, `https://www.wikipedia.org`).

3.- Habilitar HTTPS con un certificado autofirmado
Genera un certificado SSL/TLS y configura NGINX para que el acceso al sitio se realice también en `https://`.

4.- Logs y monitoreo
Localiza los archivos de registro de NGINX y comprueba qué sucede cuando accedes al sitio.

- `/var/log/nginx/access.log`
- `/var/log/nginx/error.log`

Entrega.

Redacta y publica memoria del desarrollo de las prácticas. Incluye dificultades encontradas y un análisis de las aplicaciones reales que podrías desplegar usando NGINX.

- Qué aprendiste en esta práctica.
- Las dificultades encontradas.
- Qué aplicaciones reales podrías desplegar usando NGINX.

Práctica 2.2: "Servidor Web con Apache HTTP Server"

Objetivo

Familiarizarse con la instalación, configuración y uso básico de **Apache HTTP Server** como servidor web para el despliegue de aplicaciones.

Tareas

1.- Comprueba la disponibilidad del puerto 80.
Antes de instalar Apache, debemos asegurarnos de que el **puerto 80** no esté ocupado por otro servicio (por ejemplo, por **NGINX**).

bash
sudo lsof -i :80

Si no aparece nada, significa que el puerto está libre.
Si aparece un servicio como nginx u otro, significa que ya tienes el puerto ocupado y deberás detenerlo temporalmente para trabajar con Apache o utilizar un puerto diferente:

bash

sudo systemctl stop nginx

2.- Instalación de Apache HTTP Server.
En sistemas basados en Debian/Ubuntu:

bash
sudo apt update
sudo apt install apache2 -y

3.- Verificación del servicio.
Comprueba que Apache está en ejecución:

bash
systemctl status apache2

Abre un navegador y accede a localhost o a la dirección IP del servidor.

bash
http://localhost

Debería aparecer la página de bienvenida de Apache por defecto.

4.- Estructura de archivos y directorios.
Localiza los siguientes archivos de configuración:

- `/etc/apache2/apache2.conf` (configuración principal)
- `/etc/apache2/sites-available/` (sitios disponibles)
- `/etc/apache2/sites-enabled/` (sitios habilitados)
- `/etc/apache2/ports.conf` (configuración de puertos)

El contenido web por defecto se encuentra en:

- `/var/www/html`

5.- Crear una página web simple

Edita y personaliza el archivo de inicio por defecto.

bash
sudo nano /var/www/html/index.html

Inserta un HTML sencillo:

html
<!DOCTYPE html>
<html>
<head>
  <title>Mi primer sitio con Apache</title>
</head>
<body>

```
    <h1>¡Hola, mundo desde Apache HTTP Server!</h1>
    <p>Servidor funcionando correctamente</p>
</body>
</html>
```

Refresca la página en el navegador para comprobar los cambios.

6.- Configurar un nuevo sitio web virtual

Crea un directorio para el nuevo sitio:

bash
```
sudo mkdir -p /var/www/misitio
sudo nano /var/www/misitio/index.html
```

Inserta un HTML de bienvenida personalizado.

Crea un archivo de configuración de sitio virtual:

bash
```
sudo nano /etc/apache2/sites-available/misitio.conf
```

Contenido básico:

bash
```
<VirtualHost *:80>
    ServerName misitio.local
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/misitio
    ErrorLog ${APACHE_LOG_DIR}/misitio_error.log
    CustomLog ${APACHE_LOG_DIR}/misitio_access.log combined
</VirtualHost>
```

Activa el sitio y recarga la configuración:

bash
```
sudo a2ensite misitio.conf
sudo apache2ctl configtest
sudo systemctl reload apache2
```

Añade la entrada en /etc/hosts:

bash
```
127.0.0.1 misitio.local
```

Accede al nuevo sitio:

bash
```
http://misitio.local
```

Tareas propuestas

1.- Desplegar varios sitios virtuales.
Crea al menos dos sitios virtuales adicionales, cada uno con su propia carpeta, configuración y nombre de dominio diferente.

2.- Configurar redireccionamientos.
Haz que las visitas a `http://misitio.local/noticias` redirijan a otro sitio web externo (por ejemplo, `https://www.bbc.com`).

3.- Habilitar módulos de Apache.
Investiga y habilita al menos dos módulos de Apache (como rewrite, ssl, o headers) y explica su funcionalidad.

4.- Configurar directorios protegidos con .htaccess.
Crea un directorio protegido con autenticación básica usando archivos .htaccess y .htpasswd.

5.- Logs y monitoreo.
Localiza los archivos de registro de Apache y analiza las entradas cuando accedes a los diferentes sitios:

- `/var/log/apache2/access.log`
- `/var/log/apache2/error.log`
- `/var/log/apache2/other_vhosts_access.log`

Entrega

Redacta y publica memoria del desarrollo de las prácticas. Incluye dificultades encontradas y un análisis de las aplicaciones reales que podrías desplegar usando Apache.

- Qué aprendiste en esta práctica.
- Las dificultades encontradas.
- Qué aplicaciones reales podrías desplegar usando Apache.
- Comparativa con otros servidores web como NGINX

Ahora puedes copiar y pegar este código HTML en Moodle usando el modo de edición HTML (`<>`) en una etiqueta o página.