PRÁCTICAS

1. Introducción

1.1 Ifconfig

```
ifconfig
```

Ver la IP de la interfaz wifi (algo similar a *wlp4s0*). Diferente para cada persona.

1.2 https://miip.es

Entrar a https://miip.es y ver la IP (será la misma para todos).

2. HTTP

2.1 DevTools

Entrar a https://www.w3schools.com/ y ver las DevTools (botón derecho + “Inspeccionar” + ir a la pestaña de “Red”), solo para ver las llamadas HTTP. Conforme vayamos haciendo scroll en la web, veremos cómo se hacen peticiones para los nuevos recursos.

Nota: Se puede hacer con cualquier navegador.

3. Postman & HTTPie

3.1 Navegadores

Entrar a https://www.w3schools.com/ y ver las DevTools en profundidad (pasando por *w3codecolor.js*, hasta llegar a la imagen de *w3css_templates.webp*).

Entrar a https://profile.w3schools.com/log-in y subir los datos (y así vemos el *OPTIONS* y el *POST*).

Nota: Se puede hacer con cualquier navegador.

3.2 Postman

```
sudo snap install postman
```

```
postman
```

Hacer peticiones *GET* a varias URLs de w3schools:
- https://www.w3schools.com
- https://www.w3schools.com/lib/w3codecolor.js
- https://www.w3schools.com/w3css_templates.webp
- *Cualquier otra de la web*

Comparar con el navegador.

3.3 HTTPie

```
sudo apt update
sudo apt install httpie
```

```
http https://www.w3schools.com
```

```
```

```
http https://www.w3schools.com/lib/w3codecolor.js
```

```
http https://www.w3schools.com/w3css_templates.webp
```

```
http https://www.w3schools.com/w3css_templates.webp >
imagenW3Schools1.webp
```

Comparar con el navegador.

3.4 cURL

```
# En teoría está ya instalado
sudo apt update
sudo apt install curl
```

```
curl https://www.w3schools.com
curl -v https://www.w3schools.com
```

```
curl https://www.w3schools.com/lib/w3codecolor.js
curl -v https://www.w3schools.com/lib/w3codecolor.js
```

```
curl https://www.w3schools.com/w3css_templates.webp
curl -v https://www.w3schools.com/w3css_templates.webp
```

```
curl https://www.w3schools.com/w3css_templates.webp >
imagenW3Schools2.webp
curl -v https://www.w3schools.com/w3css_templates.webp >
imagenW3Schools2.webp
```

Comparar con el navegador.

4. APIs

4.1 https://httpbin.org

Entrar a https://httpbin.org/#/ para ver su API.

```
# Petición simple
http GET https://httpbin.org/get
curl -v -X GET https://httpbin.org/get
```

Hacer la misma llamada con Postman.

```
# Con cabeceras
http GET https://httpbin.org/get accept:application/json
curl -v -X GET https://httpbin.org/get -H "accept:
application/json"
```

Hacer la misma llamada con Postman.

```
# Con el método POST
http POST https://httpbin.org/post
curl -v -X POST https://httpbin.org/post
```

Hacer la misma llamada con Postman.

```
# Para ver un 404
http GET https://httpbin.org/meloinvento
curl -v -X GET https://httpbin.org/meloinvento
```

Hacer la misma llamada con Postman.

```
# Para ver un 500 (y método PUT)
http PUT https://httpbin.org/status/500
curl -v -X PUT https://httpbin.org/status/500
```

Hacer la misma llamada con Postman.

```
# Para descargar contenido HTML (página web)
http GET https://httpbin.org/html
curl -v -X GET https://httpbin.org/html
```

Hacer la misma llamada con Postman.

```
# Para descargar contenido JSON
http GET https://httpbin.org/json
curl -v -X GET https://httpbin.org/json
```

Hacer la misma llamada con Postman.

```
# Para descargar una imagen
http GET https://httpbin.org/image accept:image/png >
imagenHttpbin1.png
curl -X GET https://httpbin.org/image -H "accept: image/png" >
imagenHttpbin2.png
```

Hacer la misma llamada con Postman.

```
# Para hacer una redirección a google.com (la cabecera
"location" devuelve a dónde hay que redireccionar)
http GET
"https://httpbin.org/redirect-to?url=https://google.com&status_
code=302"
curl -v -X GET
"https://httpbin.org/redirect-to?url=https://google.com&status_
code=302"
```

Hacer la misma llamada con Postman (vemos que Postman es inteligente, y hace la redirección automáticamente).

```
# Para enviar datos JSON al servidor (con un POST)
http POST https://httpbin.org/anything accept:application/json
nombre=ivan ciudad=pamplona
curl -v -X POST https://httpbin.org/anything -H "Content-Type:
application/json" -d '{"nombre":"ivan","ciudad":"pamplona"}'
```

Hacer la misma llamada con Postman.

```
# Para enviar datos con un GET
http GET
"https://httpbin.org/anything?nombre=ivan&ciudad=pamplona"
curl -v -X GET
"https://httpbin.org/anything?nombre=ivan&ciudad=pamplona"
```

Hacer la misma llamada con Postman.

```
# Para enviar un token de autenticación existen muchas
opciones, hay que ver cómo está configurado el servidor.
Estas son solo algunas opciones.

http GET "https://httpbin.org/anything?access-token=0123456789"
curl -v -X GET
"https://httpbin.org/anything?access-token=0123456789"

http GET https://httpbin.org/anything "Authorization:Bearer
0123456789"
curl -v -X GET https://httpbin.org/anything -H
"Authorization:Bearer 0123456789"
```

Hacer la misma llamada con Postman.

```
# Para más información sobre HTTPie y cURL
http --help
curl --help
```

```

Probar la API todo lo que se quiera.

## 4.2 https://gorest.co.in/

*A menos que nos sobre el tiempo, saltamos esto, porque es muy similar al 4.1.*
```

5. Nginx

5.1 Servidor Nginx

```
sudo apt update
sudo apt install nginx
nginx -v
```

```
sudo systemctl start nginx
sudo systemctl status nginx
```

Entrar a http://0.0.0.0 (recordad que esto es lo mismo que poner http://0.0.0.0:80, solo que, al poner “http”, no hace falta escribir el puerto 80, ya que es el puerto por defecto). Vemos ahí el servidor levantado. El “0.0.0.0” hace referencia a nuestro propio ordenador, pero en su lugar podemos poner nuestra dirección IP.

La configuración de Nginx que vamos a editar está en el archivo “/etc/nginx/sites-available/default”. Para poder editar este archivo con “Visual Studio Code” vamos a ejecutar estos comandos (si no tenemos instalado Visual Studio Code, hay que ejecutar “sudo snap install --classic code” antes):

```
sudo code /etc/nginx/sites-available/default --no-sandbox
--user-data-dir /home
```

Ahora ya podemos modificar la configuración de Nginx y guardarla. Recordad que, cada vez que modifiquemos este fichero, hay que ejecutar “sudo systemctl reload nginx” para actualizar Nginx con la nueva configuración.

Todos los archivos estáticos (HTML, JavaScript, CSS, imágenes…) que se sirven en Nginx están por defecto en el directorio “/var/www/html”. Para poder editar los archivos de ese directorio con “Visual Studio Code”, hay que ejecutar estos comandos:

```
sudo chmod -R 777 /var/www/html/
code /var/www/html/
# Ahora podemos cambiar cosas de este escritorio, por
ejemplo, el contenido del fichero index.html
```

Entrar a http://0.0.0.0 en el navegador y vemos el contenido que hemos cambiado del index.html. Entrar a http://0.0.0.0 realmente es lo mismo que entrar a http://0.0.0.0/index.html pero, como en la configuración de Nginx está escrita la línea “index index.html", eso significa que, si no se pone nada, se muestra el archivo “index.html” por defecto.

Podemos añadir nuevos archivos al directorio “/var/www/html”. Si añadimos una imagen, por ejemplo, “perro.jpg”, la podremos ver en el navegador en http://0.0.0.0/perro.jpg. Si añadimos otro archivo HTML, por ejemplo, prueba.html, lo podremos ver en el navegador en http://0.0.0.0/prueba.html.

Igualmente, podemos añadir archivos CSS, JavaScript o incluso otras carpetas, y se verán todas en nuestro servidor web (http://0.0.0.0/*ruta_al_archivo*). Si al acceder a un archivo que hemos llevado a “/var/www/html” (por ejemplo, una imagen), vemos un error 403, puede que sea por los permisos del archivo, y si eso pasa hay que volver a ejecutar “sudo chmod 777 -R /var/www/html”. También podemos crear una carpeta dentro de “/var/www/html” y meter archivos ahí.

Si accedemos a un archivo que no existe, el servidor nos devolverá un error 404, porque así está definido en el archivo “/etc/nginx/sites-available/default”.

Ahora vamos a probar a modificar la configuración de Nginx, así que haremos cambios en “/etc/nginx/sites-available/default” y luego ejecutaremos “sudo systemctl reload nginx” para actualizar Nginx. Para hacer más fácil el seguimiento, he subido mi archivo “default” a Moodle, para que podáis ver qué he ido metiendo y para qué sirve cada comando.

Recordad que, al modificar el archivo “/etc/nginx/sites-available/default”, necesitamos recargar Nginx para que coja la nueva configuración. Para eso, usaremos este comando:

```
sudo systemctl reload nginx
```

Además, al estar todos en la misma red local, sabemos que podemos acceder al servidor de otro alumno accediendo con su dirección IP. Por ejemplo, si queremos acceder al servidor de alguien que está en la IP 192.168.10.6, accederé a http://192.168.10.6.

Bonus: a modo de curiosidad, en Linux hay un pequeño “servidor DNS” en “/etc/hosts”, y podemos ir ahí y añadir por ejemplo esta línea para hacer que cuando pongamos test.com, se vaya a la dirección IP 0.0.0.0:

```
sudo code /etc/hosts --no-sandbox --user-data-dir /home
# Escribimos en la última línea de archivo "0.0.0.0 test.com" y guardamos.
```

```
# Ahora, al acceder a http://test.com es como si fuésemos a
http://0.0.0.0
```

6. Certificados SSL

6.1 Certificado en el navegador

Entrar a https://www.w3schools.com/ y ver el certificado.

Entrar a https://www.google.com/ y ver el certificado.

Entrar a https://httpie.io/ y ver el certificado.

Entrar a http://www.columbia.edu/~fdc/sample.html y ver que, al ser HTTP, no tiene certificado.

6.2 Servidor Nginx con HTTPS

Instalamos openssl para crear la clave privada y el certificado público.

```
sudo apt update
sudo apt install openssl
```

Creamos la clave privada (clave1.pem) y el certificado público (certificado1.pem).

```
openssl req -nodes -new -x509 -keyout clave1.pem -out
certificado1.pem
```

Creamos un directorio en “/etc/nginx/certs” y metemos ahí la clave y el certificado.

```
sudo mkdir /etc/nginx/certs
sudo cp clave1.pem certificado1.pem /etc/nginx/certs
```

Añadimos la conifguración de HTTPS en “/etc/nginx/sites-available/default”.

```
# Vamos a /etc/nginx/sites-available/default y añadimos esto al
final
server {
```

```
    listen              443;
    ssl                 on;
    server_name         0.0.0.0;
    ssl_certificate     /etc/nginx/certs/certificado1.pem;
    ssl_certificate_key /etc/nginx/certs/clave1.pem;
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;
    location / {
        try_files $uri $uri/ =404;
    }
}
```

Recargamos Nginx para que coja la nueva configuración.

```
sudo systemctl reload nginx
sudo systemctl status nginx
```

Podemos entrar a nuestro servidor HTTP en https://0.0.0.0 (recordad que esto es lo mismo que poner https://0.0.0.0:443, solo que, al poner “https”, no hace falta escribir el puerto 443, ya que es el puerto por defecto).

Como tenemos un certificado autofirmado (lo hemos creado nosotros), cuando entramos a https://0.0.0.0 el navegador nos avisa de que puede ser inseguro. En este caso nos saltamos el chequeo de seguridad porque ya sabemos que no es un servidor peligroso.

Podemos ver el certificado autofirmado en el navegador y compararlo con el de google.com.

Por último, podemos jugar libremente con Nginx.