



UNIVERSIDAD DE GRANADA

SWAP

Práctica 3

Trabajo individual

Nuria Manzano Mata

Profesor: José Manuel Soto Hidalgo



Tiempo dedicado: 7 horas

ÍNDICE

1. Introducción.....	2
2. Desarrollo tareas básicas.....	2
2.1. B1. Preparación del Entorno de Trabajo.....	2
2.2. B2. Creación de Certificados SSL.....	2
2.3. B3. Configuración de Servidores Web Apache con SSL.....	3
2.4. B4. Configuración del Balanceador de Carga Nginx con SSL.....	5
2.5. B5. Docker Compose para la Granja Web con SSL.....	7
2.6. B6. Verificación y Pruebas del Escenario con SSL.....	9
2.7. Análisis propuesta IA.....	12
3. Desarrollo tareas avanzadas.....	12
3.1. A1. Exploraciones Avanzadas de creación de certificados SSL.....	13
3.2. A2. Optimización de la configuración SSL en los servidores web.....	15
3.3. A3. Configuración de Caché de Sesiones SSL y Tickets de Sesión en el balanceador.....	17
3.4. A4. Optimización de conexiones HTTPS y cifrado en el balanceador.....	18
3.5. Análisis propuesta IA.....	20
Bibliografía:.....	20

1. Introducción

Esta práctica tiene como objetivo reforzar la seguridad de la infraestructura web utilizando contenedores Docker y aplicando conceptos clave de cifrado y autenticación. El **desarrollo de este documento se estructura de manera progresiva**, abordando cada tarea con un **doble propósito**: explicar detalladamente los procedimientos y soluciones implementadas, y justificar el razonamiento detrás de cada decisión. Para facilitar la comprensión, se incluyen capturas de pantalla que muestran el proceso. Debido a la extensión del documento, algunos apartados se presentan de manera concisa, mostrando los resultados más significativos o los aspectos más importantes.

El .zip entregado contiene dos carpetas, una llamada “P3” donde está todo el desarrollo de las tareas básicas y otra carpeta llamada “P3-ADVANCED” donde está todo el desarrollo de las tareas avanzadas.

2. Desarrollo tareas básicas

2.1. B1. Preparación del Entorno de Trabajo

- Crear directorios específicos para los archivos de configuración:
 - **P3-mmnuria-certificados** para crear los certificados SSL.
 - **P3-mmnuria-apache** para trabajar con los archivos de configuración de apache.
 - **P3-mmnuria-nginx** para trabajar con los archivos de configuración de nginx.

```
2025-05-02 20:16:58 ✘ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3
|o → ls
P3-mmnuria-apache P3-mmnuria-nginx web_mmnuria
P3-mmnuria-certificados docker-compose.yml
```

2.2. B2. Creación de Certificados SSL

- Generación correcta del certificado SSL y la clave privada con OpenSSL siguiendo las especificaciones dadas.

Para resolver este apartado, primero debemos acceder desde la terminal a nuestro directorio P3-mmnuria-certificados

```
2025-05-02 20:17:00 ✘ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3
|o → cd P3-mmnuria-certificados/
```

Después ejecuto el siguiente comando para la creación del certificado SSL y la clave privada con OpenSSL

```
openssl req -x509 -nodes -days 365 -newkey
rsa:2048 \
-keyout certificado_mmnuria.key \
-out certificado_mmnuria.crt \
-subj
"/C=ES/ST=Granada/L=Granada/O=SWAP/OU=Prácti
ca            3/CN=Nuria           Manzano
Mata/emailAddress=mmnuria@correo.uqr.es"
```

A continuación, explicaré cada parte del comando que tienen que ver con las especificaciones de la práctica:

- **x509**: crea un certificado autofirmado (no una CSR).
 - **nodes**: evita que se solicite passphrase.
 - **days 365**: duración de 1 año.
 - **newkey rsa:2048**: genera una nueva clave RSA de 2048 bits.
 - **keyout**: archivo donde se guardará la clave privada.
 - **out**: archivo del certificado generado.
 - **subj**: todos los datos del certificado (nombre, email, etc.)

Cómo se ve en la última captura de pantalla, se ha generado correctamente la clave privada certificado_mmnuria.key y el certificado de autofirmado certificado_mmnuria.crt

2.3. B3. Configuración de Servidores Web Apache con SSL

- Redacción adecuada del Dockerfile para los servidores Apache con soporte SSL.

Para que el Dockerfile añada correctamente el soporte SSL, en primer lugar he reutilizado el Dockerfile de la práctica 1 y le he añadido lo siguiente:

```

2025-05-02 20:18:54 📁 Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUAT
RI/SWAP/ManzanoMataNuriaP3/P3/P3-mmnnuria-apache
[O ➔ cat DockerfileApacheP3
#versión 11 de debian
FROM debian:bullseye-slim
# Instalación Apache, PHP y herramientas de red
RUN apt-get update && apt-get install -y \
    apache2 \
    php \
    libapache2-mod-php \
openssl \
    iputils-ping \
    net-tools \
    curl \
    iproute2 \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Habilita el módulo Apache para PHP (necesario para que Apache procese archivos
# PHP) y SSL
RUN a2enmod php7.4 ssl

# Crear directorio para certificados
RUN mkdir -p /etc/apache2/ssl

# Copiar los certificados SSL y clave privada
COPY certificados_mmnnuria/certificado_mmnnuria.crt /etc/apache2/ssl/certificado_m
mnuria.crt
COPY certificados_mmnnuria/certificado_mmnnuria.key /etc/apache2/ssl/certificado_m
mnuria.key

# Establecer permisos seguros
RUN chmod 600 /etc/apache2/ssl/certificado_mmnnuria.*

# Copiar archivo de configuración
COPY mmnnuria-apache-ssl.conf /etc/apache2/sites-available/mmnnuria-apache-ssl.con
f

# Habilitar el sitio SSL personalizado
RUN a2ensite mmnnuria-apache-ssl

# Establecer directorio de trabajo
WORKDIR /var/www/html

# Exponer puertos HTTP y HTTPS
EXPOSE 80
EXPOSE 443

# Mantener Apache en ejecución
CMD ["apache2ctl", "-D", "FOREGROUND"]

```

Todo lo que se ve resaltado en rosa, son las partes que he tenido que añadir en el Dockerfile para que soporte SSL.

- Configuración precisa del archivo de host virtual de Apache para atender peticiones HTTPS.

```

2025-05-02 20:19:00 ✘ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUAT
RI/SWAP/ManzanoMataNuriaP3/P3/P3-mmnia-apache
o → cat mmnia-apache-ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/html

    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/certificado_mmnia.crt
    SSLCertificateKeyFile /etc/apache2/ssl/certificado_mmnia.key

    <Directory "/var/www/html">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error_ssl.log
    CustomLog ${APACHE_LOG_DIR}/access_ssl.log combined
</VirtualHost>

```

Me he apoyado en el ejemplo básico de las diapositivas de la práctica para completar este apartado, añadiendo además lo necesario para atender peticiones HTTPS dando acceso al contenido web.

2.4. B4. Configuración del Balanceador de Carga Nginx con SSL

- Elaboración correcta del Dockerfile y la configuración de Nginx para gestionar conexiones SSL.

Para la elaboración del archivo mmnia-nginx-ssl.conf he usado como base el entregado en la práctica 2, por lo que, he tenido que modificar algunas partes y añadir otras para gestionar las conexiones SSL:

```

2025-05-02 20:20:41 ✘ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATRI/SWAP/Ma
nzanoMataNuriaP3/P3/P3-mmurria-nginx
o ➔ cat mmurria-nginx-ssl.conf
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    include      mime.types;
    default_type application/octet-stream;

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    # Grupo de servidores backend
    upstream backend_mmurria {
        server 192.168.10.2;
        server 192.168.10.3;
        server 192.168.10.4;
        server 192.168.10.5;
        server 192.168.10.6;
        server 192.168.10.7;
        server 192.168.10.8;
        server 192.168.10.9;
    }

    # Bloque HTTP: redirige a HTTPS
    server {
        listen 80;
        server_name nginx_mmurria;
        return 301 https://$host$request_uri;
    }

    # Bloque para manejar tráfico HTTPS
    server {
        listen 443 ssl;
        server_name nginx_mmurria;

        # Rutas al certificado y clave privada
        ssl_certificate      /etc/nginx/ssl/certificado_mmurria.crt;
        ssl_certificate_key  /etc/nginx/ssl/certificado_mmurria.key;

        access_log /var/log/nginx/nginx_mmurria.access.log main;
        error_log  /var/log/nginx/nginx_mmurria.error.log;

        location / {
            proxy_pass http://backend_mmurria;
            proxy_set_header Cookie $http_cookie;
            proxy_hide_header Set-Cookie;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }

        location /estadisticas_mmurria {
            stub_status on;
        }
    }
}

```

Lo que aparece resaltado en rosa son las partes que he añadido/modificado para el correcto funcionamiento del SSL. Por un lado, se redirigen todas las peticiones HTTP a HTTPS y por otro lado se maneja ese tráfico HTTPS correctamente.

Lo siguiente a realizar es la correcta configuración del Dockerfile:

```
2025-05-02 20:20:47 🐳  Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATRI/SWAP/Ma
nzanoMataNuriaP3/P3/P3-mmnnuria-nginx
[O → cat DockerFileNginxP3
FROM nginx:latest

# Crear directorio para certificados SSL
RUN mkdir -p /etc/nginx/ssl

# Copiar certificados SSL al contenedor
COPY certificados_mmnnuria/certificado_mmnnuria.crt /etc/nginx/ssl/certificado_mmnnuria.crt
COPY certificados_mmnnuria/certificado_mmnnuria.key /etc/nginx/ssl/certificado_mmnnuria.key

# Copiar archivo de configuración SSL
COPY mmnnuria-nginx-ssl.conf /etc/nginx/nginx.conf

# Exponer el puerto HTTP y HTTPS
EXPOSE 80
EXPOSE 443
```

Cómo se puede observar, he seguido las instrucciones de las diapositivas de la práctica para realizarlo, además de apoyarme en el ejemplo que también se ofrece.

2.5. B5. Docker Compose para la Granja Web con SSL

- Desarrollo de un archivo docker-compose.yml funcional que despliegue todos los servicios y configuraciones definidos en la práctica.

Para el desarrollo del docker-compose.yml he usado como base el de la práctica 1, cambiando todo lo necesario para asegurar las configuraciones definidas en las diapositivas de la práctica. Para el balanceador de carga Nginx, me he apoyado del creado en la práctica 2, cambiando y añadiendo todo lo definido en la práctica para su correcto funcionamiento.

Para cada instancia de Apache hay que añadir las siguientes configuraciones para garantizar su correcto despliegue, éstas se pueden ver resaltadas en rosa en la siguiente captura de pantalla:

```

2025-05-02 20:22:39 Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATRI/SWAP/Ma
nzanoMataNuriaP3/P3
○ → cat docker-compose.yml
version: '3'

services:
  web1:
    build:
      context: ./P3-mmnuria-apache
      dockerfile: DockerfileApacheP3
      image: mmnuria-apache-image:p3
      container_name: web1
    volumes:
      - ./web_mmnuria:/var/www/html
      - ./P3-mmnuria-certificados:/etc/apache2/ssl/
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2
    ports:
      - "8080:80"
      - "8081:443"
    restart: unless-stopped

  web2:
    image: mmnuria-apache-image:p3
    container_name: web2
    volumes:
      - ./web_mmnuria:/var/www/html
      - ./P3-mmnuria-certificados:/etc/apache2/ssl/
    networks:
      red_web:
        ipv4_address: 192.168.10.3
      red_servicios:
        ipv4_address: 192.168.20.3
    ports:
      - "8082:80"
      - "8083:443"
    restart: unless-stopped

  web3:
    image: mmnuria-apache-image:p3
    container_name: web3
    volumes:
      - ./web_mmnuria:/var/www/html
      - ./P3-mmnuria-certificados:/etc/apache2/ssl/
    networks:
      red_web:
        ipv4_address: 192.168.10.4
      red_servicios:
        ipv4_address: 192.168.20.4
    ports:
      - "8084:80"
      - "8085:443"
    restart: unless-stopped

```

Cómo se puede observar, además de la configuración requerida por la práctica, he añadido a cada instancia dos puertos, uno para las peticiones HTTP y otro para las HTTPS.

Y para el balanceador de carga:

```

# Balanceador
nginx-ssl:
  build:
    context: ./P3-mmnuria-nginx
    dockerfile: DockerFileNginxP3
    image: mmnuria-nginx-image:p3
    container_name: balanceador-nginx-ssl
  volumes:
    - ./P3-mmnuria-certificados:/etc/nginx/ssl/
    - ./P3-mmnuria-nginx/mmnuria-nginx-ssl.conf:/etc/nginx/nginx.conf
  depends_on:
    - web1
    - web2
    - web3
    - web4
    - web5
    - web6
    - web7
    - web8
  ports:
    - "8096:80"
    - "8097:443"
  networks:
    red_web:
      ipv4_address: 192.168.10.50

```

Vemos la configuración necesaria para que actúe el servidor Nginx como balanceador de carga construyéndose a partir del DockerFileNginxP3 creado anteriormente y montando los certificados y archivo de configuración. Por otro lado, he habilitado dos puertos para peticiones HTTP y HTTPS, donde el archivo de configuración de nginx (explicado anteriormente) redirige las peticiones siempre a HTTPS. El resto es igual que el balanceador de carga Nginx de la práctica 2.

2.6. B6. Verificación y Pruebas del Escenario con SSL

- Ejecución efectiva del despliegue usando Docker Compose y verificación del correcto funcionamiento del entorno SSL.

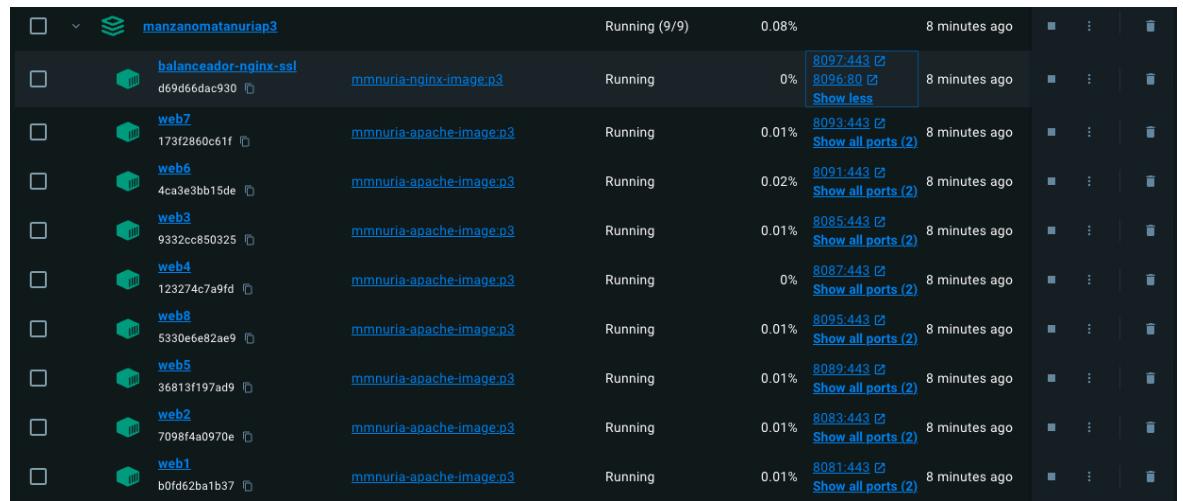
Para comprobar la ejecución efectiva del despliegue, en primer lugar voy a ejecutar el comando `docker-compose up -d` y obtenemos:

[+] Running 11/11						
✓	Network manzanomatanuriap3_red_web			Created 0.1s		
✓	Network manzanomatanuriap3_red_servicios			Created 0.1s		
✓	Container web4			Started 0.2s		
✓	Container web2			Started 0.2s		
✓	Container web1			Started 0.2s	r-nginx-ssl	
✓	Container web7			Started 0.2s		
✓	Container web8			Started 0.2s		
✓	Container web5			Started 0.2s		
✓	Container web6			Started 0.2s		
✓	Container web3			Started 0.2s		
✓	Container balanceador-nginx-ssl			Started 0.1s		

En segundo lugar ejecuto el comando docker-compose ps:

2025-04-29 14:45:34 ⌂ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATRI/SWAP/ManzanoMataNuriaP3							
● o ➔	NAME	IMAGE	COMMAND	SERVICE	CREATED	STATUS	PORTS
●	balanceador-nginx-ssl	mnnuria-nginx-image:p3	"/docker-entrypoint..."	nginx-ssl	8 minutes ago	Up 8 minutes	0.0.0.0:8096->80/tcp, 0
.	.0.0.0:8097->443/tcp						
●	web1	mnnuria-apache-image:p3	"apache2ctl -D FOREG..."	web1	8 minutes ago	Up 8 minutes	0.0.0.0:8080->80/tcp, 0
.	.0.0.0:8081->443/tcp						
●	web2	mnnuria-apache-image:p3	"apache2ctl -D FOREG..."	web2	8 minutes ago	Up 8 minutes	0.0.0.0:8082->80/tcp, 0
.	.0.0.0:8083->443/tcp						
●	web3	mnnuria-apache-image:p3	"apache2ctl -D FOREG..."	web3	8 minutes ago	Up 8 minutes	0.0.0.0:8084->80/tcp, 0
.	.0.0.0:8085->443/tcp						
●	web4	mnnuria-apache-image:p3	"apache2ctl -D FOREG..."	web4	8 minutes ago	Up 8 minutes	0.0.0.0:8086->80/tcp, 0
.	.0.0.0:8087->443/tcp						
●	web5	mnnuria-apache-image:p3	"apache2ctl -D FOREG..."	web5	8 minutes ago	Up 8 minutes	0.0.0.0:8088->80/tcp, 0
.	.0.0.0:8089->443/tcp						
●	web6	mnnuria-apache-image:p3	"apache2ctl -D FOREG..."	web6	8 minutes ago	Up 8 minutes	0.0.0.0:8090->80/tcp, 0
.	.0.0.0:8091->443/tcp						
●	web7	mnnuria-apache-image:p3	"apache2ctl -D FOREG..."	web7	8 minutes ago	Up 8 minutes	0.0.0.0:8092->80/tcp, 0
.	.0.0.0:8093->443/tcp						
●	web8	mnnuria-apache-image:p3	"apache2ctl -D FOREG..."	web8	8 minutes ago	Up 8 minutes	0.0.0.0:8094->80/tcp, 0
.	.0.0.0:8095->443/tcp						

Y por último, muestro una captura de pantalla de todo funcionando correctamente:



Cómo se puede apreciar, el despliegue del contenedor y las instancias ha sido correcto. Ahora voy a verificar el correcto funcionamiento del entorno SSL haciendo peticiones en el navegador al balanceador de carga:



Práctica 3 SWAP - Nuria Manzano Mata

La dirección IP del servidor es: 192.168.10.2

General

Issued To

Common Name (CN)	Nuria Manzano Mata
Organisation (O)	SWAP
Organisational Unit (OU)	Práctica 3

Issued By

Common Name (CN)	Nuria Manzano Mata
Organisation (O)	SWAP
Organisational Unit (OU)	Práctica 3

Validity Period

Issued On	Tuesday 29 April 2025 at 12:12:59
Expires On	Wednesday 29 April 2026 at 12:12:59

SHA-256 Fingerprints

Certificate	f106f5fe85f97f172aaeed5a238589745044c4d034292a5b416cc ae430563823 6ae410e78718285a6af72c0d1f4be1df4f3b9dca04e4d702098f1a 70c7a0f723
Public key	

Details

Certificate Hierarchy

- Nuria Manzano Mata

Certificate Fields

▼ Certificate

- Version
- Serial Number
- Certificate Signature Algorithm

Issuer

► Validity

Subject

► Subject Public Key Info

Field Value

emailAddress = mmnuria@correo.ugr.es
CN = Nuria Manzano Mata
OU = Práctica 3
O = SWAP
L = Granada
ST = Granada

Export...

2.7. Análisis propuesta IA

Con este apartado voy a tratar de **analizar los resultados propuestos por la IA** para este primer apartado de tareas básicas. He preguntado a la IA escribiendo directamente el enunciado de los diferentes apartados para que me mostrara paso a paso lo que debe de incluir cada uno, [pulsa aquí para acceder al chat](#)

De forma resumida, la IA me ha proporcionado soluciones muy básicas en las que me he apoyado para generar las mías:

- **Tarea B2:** La IA me ha proporcionado varios comandos que, tras adaptarlos a mi usuario, me permitieron generar correctamente tanto la clave privada como el certificado autofirmado.
- **Tarea B3:** Para este apartado, lo que me ha sugerido la IA me ha servido como base para configurar el servidor web Apache con SSL. No obstante, opté por incluir directivas adicionales de seguridad, permisos y registros que no estaban presentes en la configuración básica propuesta por la IA. Además, no utilicé el Dockerfile sugerido, ya que preferí reutilizar el que había creado en prácticas anteriores, siguiendo las indicaciones de las diapositivas de la práctica.
- **Tarea B4:** Al igual que en el anterior apartado, la IA me ha ofrecido una solución muy básica que me ha servido para “ubicarme” aunque finalmente no la utilicé directamente. En su lugar, reutilicé la configuración de una práctica anterior, realizando las modificaciones necesarias para adaptarla, combinando así lo aprendido previamente con las sugerencias de la IA y las instrucciones de las diapositivas.
- **Tarea B5:** En esta tarea, la IA fue útil para entender cómo añadir correctamente el volumen de los certificados y comprender el contexto de las instancias. Sin embargo, aspectos como las direcciones IP, puertos y otras configuraciones no fueron correctamente especificados. Aun así, contar con un ejemplo de referencia resultó útil para identificar lo que está “mal”, “incompleto” o “correcto” y de esa forma, tomar una decisión más apropiada.

3. Desarrollo tareas avanzadas

Para resolver este apartado, he creado una carpeta nueva llamada "P3-ADVANCED", en la que **cada sección tiene su propia subcarpeta/archivo** para facilitar la verificación del funcionamiento y mantener todo organizado. En cada apartado

se especifica el nombre de la subcarpeta/archivo creado y el paso a paso seguido para su resolución.

A diferencia de anteriores prácticas, en esta parte no se levantan nuevos contenedores, ya que, únicamente se piden cambios concretos en archivos de configuración y creación de certificados SSL, por lo que, para verificar el correcto funcionamiento, es necesario **reemplazar los archivos de configuración existentes por los nuevos que se encuentran en esta carpeta, agregar los certificados correspondientes y actualizar las rutas** a los mismos para asegurar que todo opere correctamente.

3.1. A1. Exploraciones Avanzadas de creación de certificados SSL

- Generar un certificado raíz (CA) y uno o más certificados intermedios (subCA) para entender cómo se construye una cadena de confianza. Utilizar el certificado intermedio para firmar los certificados de los servidores, simulando una estructura más realista y segura que se encuentra en entornos de producción.
 - Recomendaciones: Primero, genera clave privada de la CA y úsala para crear un certificado autofirmado que servirá como CA raíz. Segundo, crea una Autoridad Certificadora Intermedia (subCA) similar a la CA, crea una clave privada para la subCA y utilízala para generar una solicitud de certificado (CSR) de la subCA que será firmada por la CA raíz, estableciendo una cadena de confianza.

La resolución de este apartado la he realizado en la carpeta P3-ADVANCED/P3-mmnuria-certificados donde se puede encontrar tres carpetas con las diferentes claves y certificados necesarios para la construcción de una cadena de confianza. Los pasos que he seguido son:

1. Creación directorio CA raíz: en esta carpeta, se crea la clave privada y con ella el certificado autofirmado CA

```
2025-05-02 11:22:28 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmnuria-certificados
[o ➔ cd ca]

2025-05-02 11:22:31 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmnuria-certificados/ca
[o ➔ openssl genrsa -out rootCA.key 4096]

2025-05-02 11:22:47 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmnuria-certificados/ca
[o ➔ openssl req -x509 -new -key rootCA.key -sha256 -days 3650 -out rootCA.crt \
[> -subj "/C=ES/ST=Granada/L=Granada/O=SWAP/OU=Practica3/CN=RootCA"
```

- Creación directorio subCA: en esta nueva carpeta creamos una clave privada que la usamos para generar una solicitud de certificado (CSR) de la subCA para que sea firmada por la CA raíz.

```
2025-05-02 11:23:22 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmniauria-certificados/subca
o → openssl genrsa -out subCA.key 4096

2025-05-02 11:23:28 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmniauria-certificados/subca
o → openssl req -new -key subCA.key -out subCA.csr \
[> -subj "/C=ES/ST=Granada/L=Granada/O=SWAP/OU=Practica3/CN=SubCA"

2025-05-02 11:23:35 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmniauria-certificados/subca
o → openssl x509 -req -in subCA.csr -CA ../ca/rootCA.crt -CAkey ../ca/rootCA.key \
\
[> -CAcreateserial -out subCA.crt -days 1825 -sha256
Certificate request self-signature ok
subject=C=ES, ST=Granada, L=Granada, O=SWAP, OU=Practica3, CN=SubCA
```

- Creación del certificado de servidor firmado por la SubCA:

```
2025-05-02 11:24:06 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmniauria-certificados/servidores
o → openssl genrsa -out servidor.key 2048

2025-05-02 11:24:12 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmniauria-certificados/servidores
o → openssl req -new -key servidor.key -out servidor.csr \
[> -subj "/C=ES/ST=Granada/L=Granada/O=SWAP/OU=Practica3/CN=miweb.local"

2025-05-02 11:24:28 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmniauria-certificados/servidores
o → openssl x509 -req -in servidor.csr -CA ../subca/subCA.crt -CAkey ../subca/sub
CA.key \
[> -CAcreateserial -out servidor.crt -days 825 -sha256
Certificate request self-signature ok
subject=C=ES, ST=Granada, L=Granada, O=SWAP, OU=Practica3, CN=miweb.local
```

- Creación de la cadena de confianza:

```
2025-05-02 11:24:48 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmniauria-certificados/servidores
o → cat servidor.crt ../subca/subCA.crt ../ca/rootCA.crt > servidor-chain.crt
```

- Creación de un archivo de configuración de extensiones para la subCA: este paso es necesario para que la cadena de confianza funcione correctamente, por ello definimos un archivo subca_ext.cnf para que la subCA puede actuar como autoridad certificadora y firmar certificados:

```

2025-05-02 11:51:45 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmnuria-certificados/subca
[o → cat subca_ext.cnf
[ v3_ca ]
basicConstraints = critical,CA:TRUE
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer

```

6. Regeneración del certificado de la subCA con las extensiones adecuadas: en esta parte se crea un certificado intermedio válido, que ahora incluye las extensiones necesarias para ser reconocido como una CA válida.

```

2025-05-02 11:35:26 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmnuria-certificados/subca
[o → openssl x509 -req -in subCA.csr \
> -CA ../ca/rootCA.crt -CAkey ../ca/rootCA.key -CAcreateserial \
> -out subCA.crt -days 1825 -sha256 -extfile subca_ext.cnf -extensions v3_ca
Certificate request self-signature ok
subject=C=ES, ST=Granada, L=Granada, O=SWAP, OU=Practica3, CN=SubCA

```

7. Comprobación de la cadena de confianza:

```

2025-05-02 11:36:24 ☺ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGUNDO CUATR
I/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmnuria-certificados/servidores
[o → openssl verify -CAfile ../ca/rootCA.crt -untrusted ../subca/subCA.crt servido]
r.crt
servidor.crt: OK

```

Finalmente, el último paso, confirma que el certificado del servidor fue correctamente firmado por la subCA, y a su vez por la CA raíz, estableciendo una **cadena de confianza válida**.

3.2. A2. Optimización de la configuración SSL en los servidores web

- Optimizar la configuración SSL en Apache para mejorar tanto la seguridad como el rendimiento de las conexiones seguras. Para ello, deshabilitar protocolos inseguros y cifrados débiles que pueden ser explotados por ataques. Por ejemplo, permitir o denegar protocolo TLS v1, TLS v1.1 y TLS v1.2 así como cifrados MD5, RC4 y 3DES. Justifica la elección de cada uno.

La resolución de este apartado se puede encontrar en P3-ADVANCED/mmnuria-apache-ssl.conf en donde he añadido lo siguiente:

```

2025-05-02 12:04:04 ✉ Nurias-MacBook-Pro in ~/Desktop/UNI/QUINTO/SEGU
NDO CUATRI/SWAP/ManzanoMataNuriaP3/P3-ADVANCED/P3-mmnia-apache
● o → cat mmnia-apache-ssl.conf
<VirtualHost *:443>
    DocumentRoot /var/www/html

    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/certificado_mmnia.crt
    SSLCertificateKeyFile /etc/apache2/ssl/certificado_mmnia.key

    SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
    SSLCipherSuite HIGH:!aNULL:!MD5:!RC4:!3DES
    SSLHonorCipherOrder on

    <Directory "/var/www/html">
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error_ssl.log
    CustomLog ${APACHE_LOG_DIR}/access_ssl.log combined
</VirtualHost>

```

Lo que aparece resaltado en rosa, refuerza la seguridad del servidor Apache y el rendimiento de las conexiones HTTPS. La elección ha sido:

- **SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1:** se deshabilitan los protocolos SSLv3, TLSv1.0 y TLSv1.1 debido a que son considerados inseguros. SSLv3 es vulnerable al ataque POODLE, mientras que TLSv1 y TLSv1.1 carecen de soporte moderno y no ofrecen mecanismos de seguridad avanzados como el Forward Secrecy. Se permite únicamente TLSv1.2 y TLSv1.3 (este último si está habilitado por defecto en el servidor).
- **SSLCipherSuite HIGH:!aNULL:!MD5:!RC4:!3DES:** se fuerza el uso de suites de cifrado "fuertes" (HIGH) y se excluyen:
 - aNULL: suites sin autenticación, altamente inseguras.
 - MD5: algoritmo criptográficamente roto y vulnerable a colisiones
 - RC4: obsoleto y vulnerable a múltiples tipos de ataques de canal lateral.
 - 3DES: aunque aún algo usado, ya no se considera seguro por su baja longitud efectiva de clave (112 bits) y posibilidad de ataques por colisión.

- SSLHonorCipherOrder on: obliga al servidor a priorizar las suites de cifrado que considere más seguras, en lugar de seguir las que prefiera el cliente. Esto garantiza que se usen los cifrados más robustos disponibles.

3.3. A3. Configuración de Caché de Sesiones SSL y Tickets de Sesión en el balanceador

- Configurar Nginx para utilizar caché de sesiones SSL y tickets de sesión para mejorar la velocidad de las conexiones seguras repetidas, reduciendo el tiempo necesario para negociar la seguridad de la conexión.

En la ruta P3-ADVANCED/mmnuria-nginx-ssl.conf se encuentra el archivo modificado para que el servidor Nginx utilice caché de sesiones SSL y tickets de sesión:

```
# Bloque para manejar tráfico HTTPS
server {
    listen 443 ssl;
    server_name nginx_mmnuria;

    # Rutas al certificado y clave privada
    ssl_certificate      /etc/nginx/ssl/certificado_mmnuria.crt;
    ssl_certificate_key  /etc/nginx/ssl/certificado_mmnuria.key;

    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;
    ssl_session_tickets on;

    access_log /var/log/nginx/nginx_mmnuria.access.log main;
    error_log  /var/log/nginx/nginx_mmnuria.error.log;

    location / {
        proxy_pass http://backend_mmnuria;
        proxy_set_header Cookie $http_cookie;
        proxy_hide_header Set-Cookie;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

Lo resaltado en rosa, es la parte que he añadido en el bloque de server donde se maneja el tráfico HTTPS, de forma que:

- `ssl_session_cache shared:SSL:10m;`: permite que Nginx almacene en memoria sesiones TLS para que, si un cliente vuelve a conectarse, se reutilice la sesión sin renegociar desde cero. Mejora tiempos de carga.
- `ssl_session_timeout 10m;`: define que esas sesiones almacenadas serán válidas durante 10 minutos.
- `ssl_session_tickets on;`: activa los tickets TLS (tickets de sesión almacenados en el cliente). Reduce aún más el tiempo de handshake al permitir que el cliente reutilice una sesión sin que el servidor mantenga su estado (ideal para balanceadores).

Con esta configuración se consigue mejorar la velocidad de las conexiones seguras repetidas, reduciendo el tiempo necesario para negociar la seguridad de la conexión.

3.4. A4. Optimización de conexiones HTTPS y cifrado en el balanceador

- Personalizar los protocolos SSL/TLS y suites de cifrado para equilibrar seguridad y rendimiento y activar HTTP/2 para mejorar la eficiencia de las conexiones HTTPS. Por ejemplo, protocolos TLSv1.2 y TLSv1.3 así como cifrado ECDH, AESGCM, AES256, etc. Justifica la elección de cada uno.

Este apartado ha sido realizado en P3-ADVANCED/mmnuria-nginx-ssl-EJ4.conf de forma que, se ha personalizado los protocolos y activado HTTP/2 de la siguiente forma:

```

# Bloque para manejar tráfico HTTPS
server {
    listen 443 ssl http2;
    server_name nginx_mmnuria;

    # Rutas al certificado y clave privada
    ssl_certificate      /etc/nginx/ssl/certificado_mmnuria.crt;
    ssl_certificate_key  /etc/nginx/ssl/certificado_mmnuria.key;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305';
    ssl_prefer_server_ciphers on;

    access_log /var/log/nginx/nginx_mmnuria.access.log main;
    error_log /var/log/nginx/nginx_mmnuria.error.log;

    location / {
        proxy_pass http://backend_mmnuria;
        proxy_set_header Cookie $http_cookie;
        proxy_hide_header Set-Cookie;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /estadisticas_mmnuria {
        stub_status on;
    }
}

```

Cómo se puede apreciar en la captura de pantalla, se ha añadido:

- `listen 443 ssl http2;` habilita HTTP/2, que mejora notablemente el rendimiento de las conexiones HTTPS:
 - Multiplexación de peticiones/resuestas sobre una misma conexión.
 - Compresión de cabeceras.
 - Reducción de la latencia.
- `ssl_protocols TLSv1.2 TLSv1.3;` se restringen los protocolos SSL a TLS 1.2 y TLS 1.3, eliminando versiones anteriores y vulnerables como TLS 1.0 y 1.1. TLS 1.3, en particular, permite un handshake más rápido y seguro, con mejores algoritmos criptográficos y sin necesidad de renegociación.
- `ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:... .';` se definen suites de cifrado modernas con:
 - ECDHE (Elliptic Curve Diffie-Hellman Ephemeral): garantiza perfect forward secrecy, evitando que la clave de sesión sea reutilizada o recuperada aunque se comprometa la clave del servidor.
 - AES-GCM y CHACHA20-POLY1305: cifrados autenticados eficientes, resistentes a ataques de canal lateral y recomendados por la mayoría de guías de seguridad actuales.

- `ssl_prefer_server_ciphers on;` esta directiva fuerza que sea el servidor quien defina el orden de preferencia de los algoritmos de cifrado, garantizando el uso de los más seguros, en lugar de dejar esa decisión al navegador cliente.

3.5. Análisis propuesta IA

En este apartado trato de analizar los resultados propuestos por la IA de este segundo apartado para las tareas avanzadas. He preguntado a la IA escribiendo directamente el enunciado de las diferentes tareas para que me mostrara paso a paso lo que debe de incluir cada una, [pulsa aquí para acceder al chat](#)

De forma resumida, la IA me ha proporcionado soluciones bastante útiles:

- Tarea A1: Para esta tarea, la IA me ha sugerido una serie de pasos y comandos algo confusos para mí, es por ello que, investigando en internet y revisando el temario de la asignatura, he conseguido obtener los comandos necesarios para la realización correcta de este apartado, además de que, he creado carpetas diferentes que las que me ha sugerido la IA, simplemente porque me parece menos confuso. En definitiva, lo que me sugiere la IA es correcto pero, no lo he usado “directamente” ya que me resultó algo confuso y tuve que realizar previamente una labor de búsqueda y entendimiento.
- Tarea A2, A3 y A4: Para estos apartados si que me ha proporcionado una buena solución la IA, es cierto que, simplemente me ha dicho que hay que poner sin explicarme dónde, pero no ha sido algo complicado de entender por mí misma.

Bibliografía:

- Mozilla. (s.f.). Server Side TLS - Mozilla SSL Configuration Generator. Mozilla Foundation. <https://ssl-config.mozilla.org/>
- NGINX, Inc. (s.f.). Configuring HTTPS servers. NGINX Documentation. https://nginx.org/en/docs/http/configuring_https_servers.html
- Soto Hidalgo, J. M. (2024). *Guía Práctica 3 – Seguridad (certificados SSL)* [Documento académico]. Universidad de Granada.
- Apache Software Foundation. (2024). Apache HTTP Server Documentation – SSL/TLS Encryption. <https://httpd.apache.org/docs/current/ssl/>
- Nginx Inc. (2024). NGINX Documentation – Configure HTTPS Servers. https://nginx.org/en/docs/http/configuring_https_servers.html