"Despliegue de Servidor Web con Docker: Instalación y Configuración de Aplicaciones PHP"

Alumnos:

Mary Almela

Sergio Ruiz

Pablo del Pozo

Índice

Contenido

Introducción

La documentación que presentamos tiene como objetivo mostrar el desarrollo de un servidor web completo y seguro utilizando Docker, una plataforma de software que permite crear, probar, e implementar aplicaciones rápidamente, empaquetando software en unidades estandarizadas llamadas contenedores; que incluyen todo lo necesario para que el software se ejecute.

Utilizando todo el contenido aprendido en clases, proseguiremos a presentarles paso por paso dicha implementación y sus respectivas pruebas de funcionamiento.

Creación de Dockerfile

Primero, crearemos un archivo llamado Dockerfile en el directorio raíz del proyecto, el cual se llama proyecto_apache, es desde donde estaremos trabajando. Este archivo contendrá las instrucciones para construir nuestra imagen Docker.

Actualizamos el sistema e instalamos el software de apache. Luego instalamos libapache2-mod-php que es un módulo de Apache que permite ejecutar código PHP directamente dentro del servidor web Apache.

```
FROM ubuntu

RUN apt update && apt install -y apache2

RUN apt install -y libapache2-mod-php

RUN rm /var/www/html/index.html

ADD index.php /var/www/html/

CMD [ "/usr/sbin/apache2ctl", "-D", "FOREGROUND" ]
```

Proseguimos con eliminar el fichero index.html que es la que se enseña por defecto. En nuestro directorio raíz tenemos un archivo index.php correspondiente a lo que será la página de fibonacci.com. La copiamos a la carpeta donde estaba el anterior index y probamos que funciona.

Para ahorrar tiempo creamos un archivo donde tenemos un script de shell que para es para el contenedor, si ya existe, lo borra. Y luego utilizamos el comando build para crear la imagen a partir del Docker file y con run construimos el contenedor

```
  GNU nano 7.2                          launch.sh
docker stop apachito
docker rm apachito
docker build -t img_apachito .
docker run -d -p 80:80 --name apachito img_apachito
```

Corremos el archivo launch y el contenedor se crea y vemos que funciona

```
root@vbox:/home/proyecto/proyecto_apache# ./launch.sh
Error response from daemon: No such container: apachito
Error response from daemon: No such container: apachito
[+] Building 7.0s (10/10) FINISHED                    docker:default
 => [internal] load build definition from Dockerfile            0.0s
 => => transferring dockerfile: 247B                            0.0s
 => [internal] load metadata for docker.io/library/ubuntu:late  6.7s
 => [internal] load .dockerignore                               0.0s
 => => transferring context: 2B                                 0.0s
 => [1/5] FROM docker.io/library/ubuntu:latest@sha256:99c35190  0.0s
 => [internal] load build context                               0.0s
 => => transferring context: 30B                                0.0s
 => CACHED [2/5] RUN apt update && apt install -y apache2       0.0s
 => CACHED [3/5] RUN apt install -y libapache2-mod-php          0.0s
 => CACHED [4/5] RUN rm /var/www/html/index.html                0.0s
 => CACHED [5/5] ADD index.php /var/www/html/                   0.0s
 => exporting to image                                          0.0s
 => => exporting layers                                         0.0s
 => => writing image sha256:4124e328d297c96f7549e97a1960fd1347  0.0s
 => => naming to docker.io/library/img_apachito                 0.0s
a123b11bf2baabbffeb111b4df5233d3210c99ded783914cc3e802485ee25fd0
root@vbox:/home/proyecto/proyecto_apache# S
```

Creación y funcionamiento del sitio Fibonacci

La página Fibonacci mostrará hasta el número 100 de la serie de Fibonacci en código PHP. Primero instalamos el módulo PHP libapache2-mod-php.

```
  GNU nano 7.2                                  Dockerfile *
FROM ubuntu

RUN apt update && apt install -y apache2

RUN apt install -y libapache2-mod-php
```

libapache2-mod-php es un módulo que permite al servidor web Apache interpretar y ejecutar scripts PHP para que podamos visualizar la página de forma correcta.

```
drwxr-xr-x 2 root docker 4096 Nov  7 09:42 fibonacci

-rw-r--r-- 1 root docker 737 Nov 11 11:03 fibonacci.conf
-rw-r--r-- 1 root docker 477 Oct 24 10:13 index.php
```

Tenemos el index.

```
  GNU nano 7.2                          index.php
<?php
function fibonacci($n) {
    $fib = [0, 1]; // Inicializamos la serie con los dos primeros números

    while (true) {
        $next = $fib[count($fib) - 1] + $fib[count($fib) - 2]; // Suma de los dos último>
        if ($next > $n) {
            break; // Salir si el siguiente número supera 100
        }
        $fib[] = $next; // Agregar el siguiente número a la serie
    }

    return $fib;
}

$result = fibonacci(100);
echo implode(', ', $result);
?>
```

Y el archivo de configuración donde configuramos un que escucha en el puerto 80 para las solicitudes HTTP dirigidas a fibonacci.com y www.fibonacci.com, especifica que los archivos se servirán desde el directorio /var/www/fibonacci, establece la dirección de correo electrónico del administrador del servidor, y configura registros de errores y accesos específicos para este dominio,

almacenando los logs
en fibonacci.error.log y fibonacci.custom.log respectivamente.

```
  GNU nano 7.2                fibonacci.conf
<VirtualHost *:80>
    ServerAdmin mary@fibonacci.com
    ServerName fibonacci.com
    ServerAlias www.fibonacci.com
    DocumentRoot /var/www/fibonacci
    ErrorLog ${APACHE_LOG_DIR}/fibonacci.error.log
    CustomLog ${APACHE_LOG_DIR}fibonacci.custom.log combined
</VirtualHost>
```

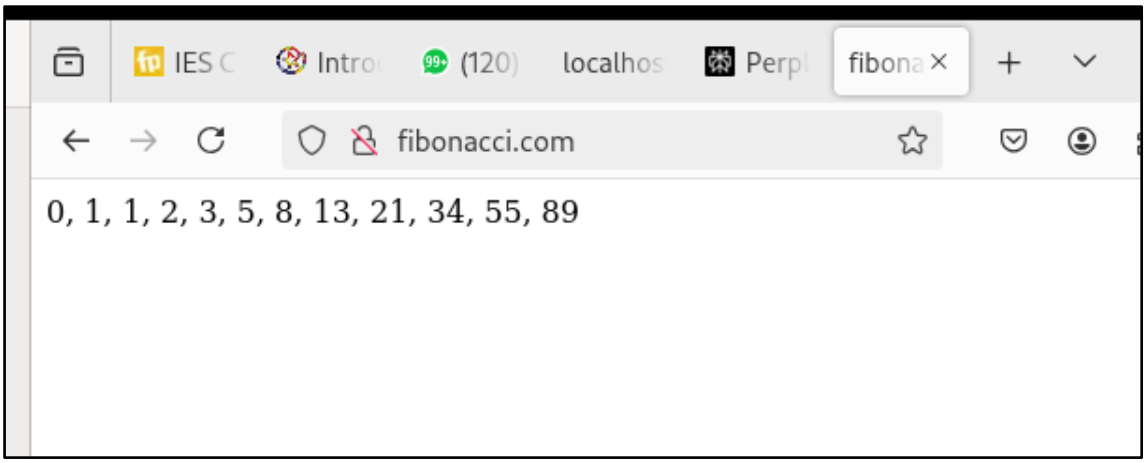
En el dockerfile hacemos algunas modificaciones. Movemos el archivo de configuración de Fibonacci a la carpeta de sites-available dentro del contedor, movemos el index que está en la carpeta de Fibonacci a una carpeta que se creará automáticamente en var/www/, agregamos los permisos y corremos el comando a2ensite para el archivo de configuración de Fibonacci para habilitar la configuración del host virtual y podamos acceder a él.

```
RUN apt install -y libapache2-mod-php

RUN rm /var/www/html/index.html

ADD /fibonacci/fibonacci.conf /etc/apache2/sites-available

ADD /fibonacci/index.php /var/www/fibonacci/

RUN chmod 777 /var/www/fibonacci/index.php

RUN a2ensite fibonacci.conf
```

Agregamos el nombre de dominio para mapearlo a la dirección IP.

```
 GNU nano 7.2                          hosts
127.0.0.1       localhost
127.0.0.1       fibonacci.com           www.fibonacci.com
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2      01e1cc5f2ab3
```

Vemos que funciona



Instalación y funcionamiento de mariaDB

```
ENV DEBIAN_FRONTEND=noninteractive

RUN apt update && apt install -y apache2

RUN apt-get update && apt-get install -y mariadb-server

RUN apt-get update && apt-get install -y \
    php \
    php-mysql \
    php-pdo

RUN sed -i 's/bind-address\s*=\s*127.0.0.1/bind-address = 0.0.0.0/' /etc/mysql/mariadb.conf.d/50-server.cnf

EXPOSE 3306

COPY setup.sql setup.sql
COPY init.sh init.sh
RUN chmod +x init.sh
RUN ./init.sh
```

Actualizamos nuevamente el indice de paquetes y luego iinstalamos el servidor de base de datos MariaDB

-Explicación basica de MariaDB:

Es un sistema de gestión de base de datos compatible con Mysql y es el backend de la base de datos.

Luego instalamos php y las extensiones necesarias para que php pueda conectarse a MariaDB.

Después runeamos el comando siguiente para que modifique el archivo de configuración de MariaDB para cambiar la dirección de enlace de 127.0.0.1 a 0.0.0.0, esto permite que mariadb acepte conexiones desde cualquier dirección ip.

Luego hacemos un EXPOSE 3306 que lo que hace es abrir el puerto donde mariadb escucha.

Luego copias el archivo sql desde el directorio local al contenedor. Ya que este archivo contiene comandos para crear una base de datos y tablas.

Copiamos el archivo de script init.sh al contenedor, el cual contiene comandos para inicializar mariadb.

Runeamos el siguiente comando para dar permisos de ejecución al arhivo init.sh permitiendo que el script pueda ejecutarse el contenedor.

Y por ultimo ejecuta el script en el contenedor.

```
CMD ["/bin/sh", "-c", "service mariadb start && /usr/sbin/apache2ctl -D FOREGROUND"]
```

Inicia el servicio de mariadb en el contenedor e inicia el servidor apache para que continúe ejecutándose mientras el contenedor esté activo.

```
  GNU nano 7.2                                setup.sql
-- setup.sql

-- Crea la base de datos "empresa"
CREATE DATABASE IF NOT EXISTS empresa;

-- Usa la base de datos "empresa"
USE empresa;

-- Crea la tabla "trabajador"
CREATE TABLE IF NOT EXISTS trabajador (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL
);

INSERT INTO trabajador(nombre) VALUES('Sergio');
INSERT INTO trabajador(nombre) VALUES('Mary');
INSERT INTO trabajador(nombre) VALUES('Pablo');

ALTER USER 'root'@'localhost' IDENTIFIED BY '1234';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

Creamos una base ded datos llamada empresa.

Seleccionamos la base de datos empresa para que los comandos siguientes se ejecuten en el contexto de la base de datos.

Creamos una tabla llamada trabajador dentro de la base de datos empresa.

Dentro de la tabla introducimos dos columnas una llamada id con tipo int y primary key, luego una llamada nombre de tipo varchar.

Introducimos los registros que queremos en la tabla.

Despues cambiamos la contraseña del usuario para lass conexiones desde localhost, esto asegura que el usuario tenga contraseña específica.

Y le damos los privilegios.

```
#!/bin/sh

# Inicia el servicio de MariaDB
service mariadb start

# Configura la contraseña de root si no está configurada
mysql -u root < setup.sql
```

Iniciamos el servicio.

```
root@vbox:/home/proyecto/proyecto_apache# docker exec -it apachito mariadb -u root -p



Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.11.8-MariaDB-0ubuntu0.24.04.1 Ubuntu 24.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| empresa            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.003 sec)
```

Iniciamos el comando para entrar en la base de datos, metemos la contraseña y luego hacemos un show database para verificar que la base de datos empresa esta creada.

```
root@vbox:/home/proyecto/proyecto_apache# docker exec -it apachito mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 7
Server version: 10.11.8-MariaDB-0ubuntu0.24.04.1 Ubuntu 24.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> USE empresa
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [empresa]> SELECT * FROM trabajador
    -> ;
+----+--------+
| id | nombre |
+----+--------+
|  1 | Sergio |
|  2 | Mary   |
|  3 | Pablo  |
+----+--------+
3 rows in set (0.033 sec)
```

Ponemos el comando SELECT… para verificar que esta todo lo creado anteriormente.

Creación y funcionamiento del sitio dawdb2

Esta línea actualiza el sistema e instala PHP junto con las extensiones necesarias para trabajar con bases de datos MySQL y usar la interfaz PDO.

```
RUN apt-get update && apt-get install -y mariadb-server

RUN apt-get update && apt-get install -y \
    php \
    php-mysql \
    php-pdo
```

Creamos una carpeta llamada dawdb2 donde añadimos los archivos dawdb2.conf e index.php, de esta forma podemos tener dos index.php diferentes, uno para fibonacci y otro para dawdb2.

```
-rw-r--r-- 1 root docker 262 Nov  4 11:40 dawdb2.conf
-rw-r--r-- 1 root docker 473 Nov  4 12:24 index.php
```

Modificamos el index.php de dawdb2 añadiendo la cadena de conexión, el usuario y la clave con la que poder acceder a la base de datos, y escribimos código PHP para recoger todos los datos de nuestra tabla trabajador y que los escriba.

```
 GNU nano 7.2                                                  index.php
<?php
    $cadena_conexion = 'mysql:dbname=empresa;host=127.0.0.1';
    $usuario = 'root';
    $clave = '1234';

    // El try-catch es opcional
    try {
        $bd = new PDO($cadena_conexion, $usuario, $clave);
        $sql = "SELECT *  FROM trabajador";
        $usuarios = $bd->query($sql);

        foreach ($usuarios as $usu) {
                echo $usu['id'];
                echo $usu['nombre'];
                echo "<br>";
        }
    } catch (PDOException $e) {
        echo 'Error con la base de datos: ' . $e->getMessage();
    }
?>
```

Configuramos el archivo dawdb2.conf para que escuche en el puerto 80 para las solicitudes HTTP dirigidas a dawdb2.com y www.dawdb2.com. También le especificamos la ruta de los documentos la cual es /var/www/dawdb2, y le añadimos un registro de errores y accesos que se guardarán en dawdb2.error.log y dawdb2.custom.log sucesivamente.

```
<VirtualHost *:80>
    ServerAdmin mary@dawdb2.com
    ServerName dawdb2.com
    ServerAlias www.dawdb2.com
    DocumentRoot /var/www/dawdb2
    ErrorLog ${APACHE_LOG_DIR}/dawdb2.error.log
    CustomLog ${APACHE_LOG_DIR}dawdb2.custom.log combined
</VirtualHost>
```

Añadimos dawdb2.conf a sites-available para poder acceder a la página, añadimos el archivo index.php a la carpeta /var/www/dawdb2, le damos todos los permisos y activamos el archivo dawdb2.conf con a2ensite para ya por fín poder acceder a la página y que nos muestre el index.php.

```
ADD /dawdb2/dawdb2.conf /etc/apache2/sites-available

ADD /dawdb2/index.php /var/www/dawdb2/

RUN chmod 777 /var/www/dawdb2/index.php

RUN a2ensite dawdb2.conf
```

Aquí la prueba de acceso a la página dawdb2.com donde nos muestra el index.php, y podemos ver los datos que están en la base de datos.

Certificados y HTTPS

Antes de hacer la configuración para los certificados ejecutamos el contenedor, accedimos a él con Docker exec y ejecutamos el comando openssl versión para verificar si tenía alguna versión de ssl instalada, lo cual comprobamos que si

```
root@e03636a0207b:/# openssl version
OpenSSL 3.0.13 30 Jan 2024 (Library: OpenSSL 3.0.13 30 Jan 2024)
```

En el dockerfile actualizamos el índice de paquetes e instalamos openssl por si acaso, que es una herramienta para manejar certificados y configuraciones de cifrado. Se usa para generar certificados SSL/TLS.

Con a2enmod ssl habilitamos el módulo ssl, permitiendo que el servidor soporte conexiones HTTPS. Luego corremos el comando que genera un certificado SSL autofirmado. Especificamos que la clave privada se guarde en etc/ssl/private y el certificado en etc/ssl/certs.

Con -subj especificamos detalles del certificado como país, estado, localidad, organización y nombre común (CN). Lo especificamos en el Docker ya que si no se especifica -subj, openssl pedirá al usuario que ingrese los detalles uno por uno, lo que detendría el proceso de construcción del contenedor.

Luego añadimos manualmente la carga del módulo ssl_module en el archivo de configuración de Apache (apache2.conf). Esto asegura que el módulo SSL esté cargado en caso de que no esté habilitado de forma automática.

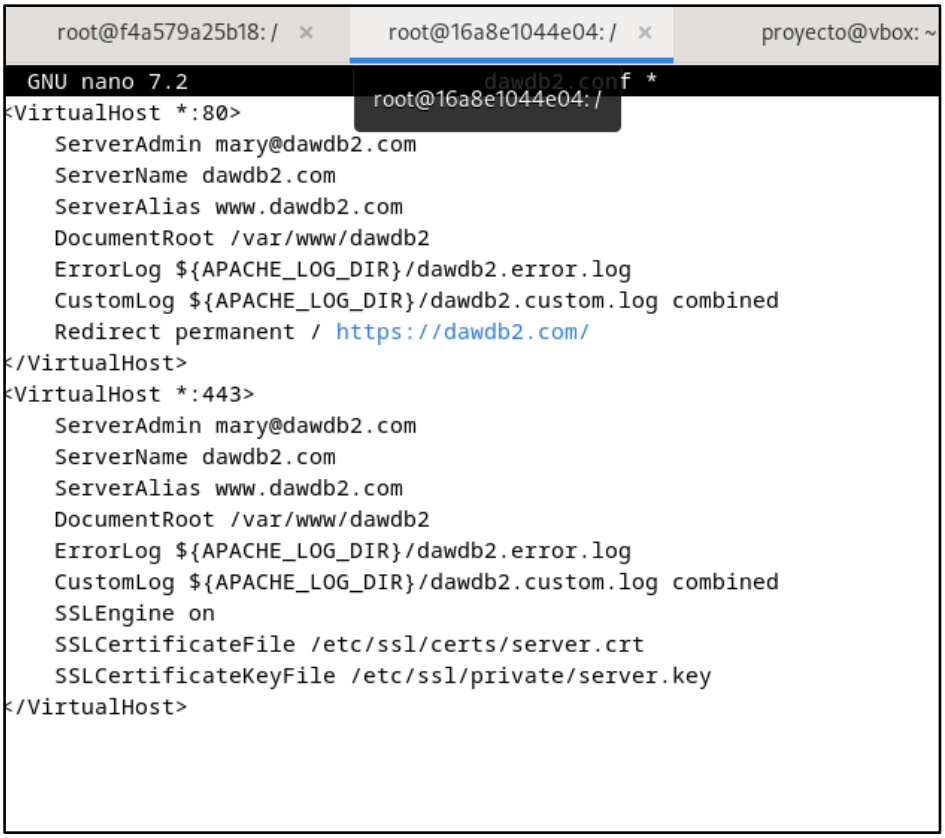
Exponemos el puerto 443, que es el puerto estándar para conexiones HTTPS. Esto permite que las conexiones externas accedan al servidor Apache en HTTPS.

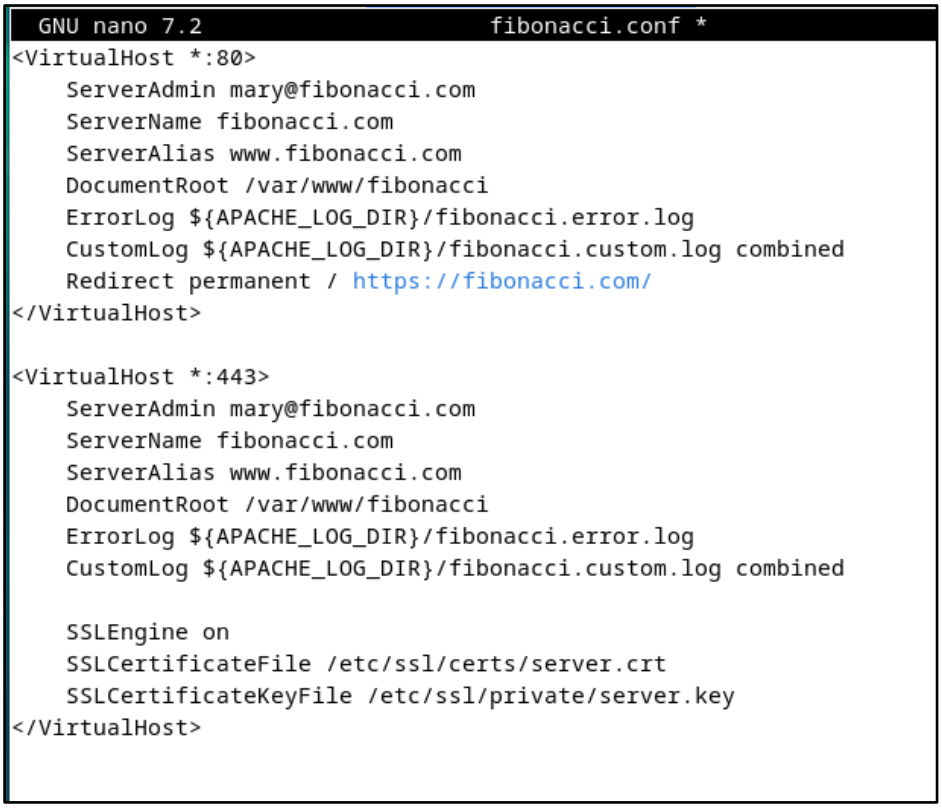
```
RUN apt-get update && apt-get install -y openssl

RUN a2enmod ssl

RUN openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
-keyout /etc/ssl/private/server.key -out /etc/ssl/certs/server.crt \
-subj "/C=ES/ST=MAD/L=MAD/0=Company/OU=CLARADELREY/CN=localhost"

RUN echo "LoadModule ssl_module modules/mod_ssl.so" >> /etc/apache2/apache2.conf

EXPOSE 443

CMD ["/bin/sh", "-c", "service mariadb start && /usr/sbin/apache2ctl -D FOREGROUND"]
```

Comprobamos que el contenedor se crea perfectamente.

```
=> => transferring context: 2B                                                                                                  0.0s
=> CACHED [ 1/23] FROM docker.io/library/ubuntu:latest@sha256:99c35190e22d294cdace2783ac55effc69d32896daaa265f0bbedbcde4fbe3e5  0.0s
=> [internal] load build context                                                                                                0.1s
=> => transferring context: 270B                                                                                                0.0s
=> [ 2/23] RUN apt update && apt install -y apache2                                                                            26.2s
=> [ 3/23] RUN apt-get update && apt-get install -y mariadb-server                                                             22.2s
=> [ 4/23] RUN apt-get update && apt-get install -y     php     php-mysql     php-pdo                                      22.0s
=> [ 5/23] RUN sed -i 's/bind-address\s*=\s*127.0.0.1/bind-address = 0.0.0.0/' /etc/mysql/mariadb.conf.d/50-server.cnf          0.4s
=> [ 6/23] COPY setup.sql setup.sql                                                                                             0.2s
=> [ 7/23] COPY init.sh init.sh                                                                                                 0.1s
=> [ 8/23] RUN chmod +x init.sh                                                                                                 0.4s
=> [ 9/23] RUN ./init.sh                                                                                                        1.9s
=> [10/23] RUN apt install -y libapache2-mod-php                                                                                2.5s
=> [11/23] RUN rm /var/www/html/index.html                                                                                      0.4s
=> [12/23] ADD /fibonacci/fibonacci.conf /etc/apache2/sites-available                                                           0.2s
=> [13/23] ADD /fibonacci/index.php /var/www/fibonacci/                                                                         0.1s
=> [14/23] RUN chmod 777 /var/www/fibonacci/index.php                                                                           0.4s
=> [15/23] RUN a2ensite fibonacci.conf                                                                                          0.5s
=> [16/23] ADD /dawdb2/dawdb2.conf /etc/apache2/sites-available                                                                 0.2s
=> [17/23] ADD /dawdb2/index.php /var/www/dawdb2/                                                                               0.1s
=> [18/23] RUN chmod 777 /var/www/dawdb2/index.php                                                                              0.4s
=> [19/23] RUN a2ensite dawdb2.conf                                                                                             0.5s
=> [20/23] RUN apt-get update && apt-get install -y openssl                                                                     7.0s
=> [21/23] RUN a2enmod ssl                                                                                                      0.5s
=> [22/23] RUN openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/server.key -out /etc/ssl/certs/server.cr  0.8s
=> [23/23] RUN echo "LoadModule ssl_module modules/mod_ssl.so" >> /etc/apache2/apache2.conf                                    0.4s
=> exporting to image                                                                                                           4.5s
=> => exporting layers                                                                                                          4.4s
=> => writing image sha256:769484f2f623a9f9bd92bbd16ea9b65cae69b7e5c3280f004d0bfae70d5a13f7                                     0.0s
=> => naming to docker.io/library/img_apachito                                                                                  0.0s
3beff3dbfede8e143a02376e043f7b5b53faf3a0c6a57091a4b5fed7d7fb3ef6
```

Cambiamos la configuración de ambas páginas, la de Fibonacci y la de dawdb2, con esta nueva configuración definimos un VirtualHost para el dominio dawdb2.com y uno para Fibonacci, manejando tanto conexiones HTTP (*:80) como HTTPS (*:443). Este mapeo redirige todo el tráfico HTTP a HTTPS para garantizar conexiones seguras. Incluimos las configuraciones para directorios y logs como las que enseñamos anteriormente y además agregamos la configuración de seguridad SSL, especificando los archivos de certificado y clave para cifrar las comunicaciones.



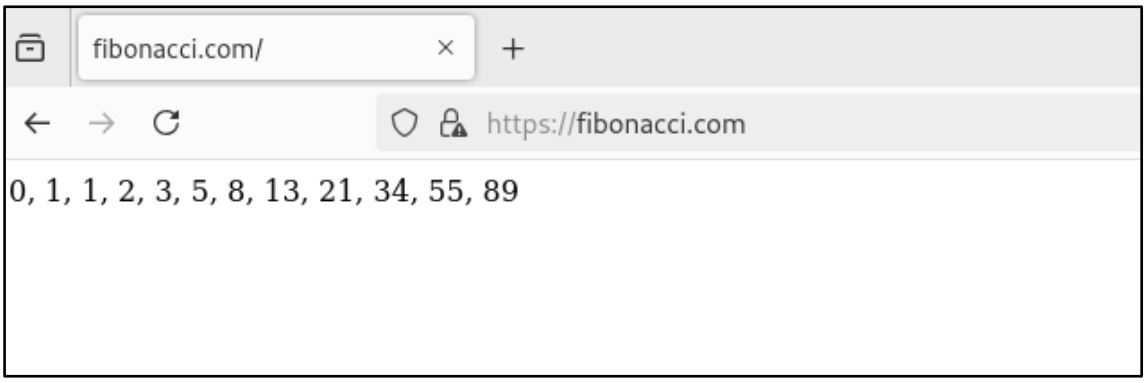


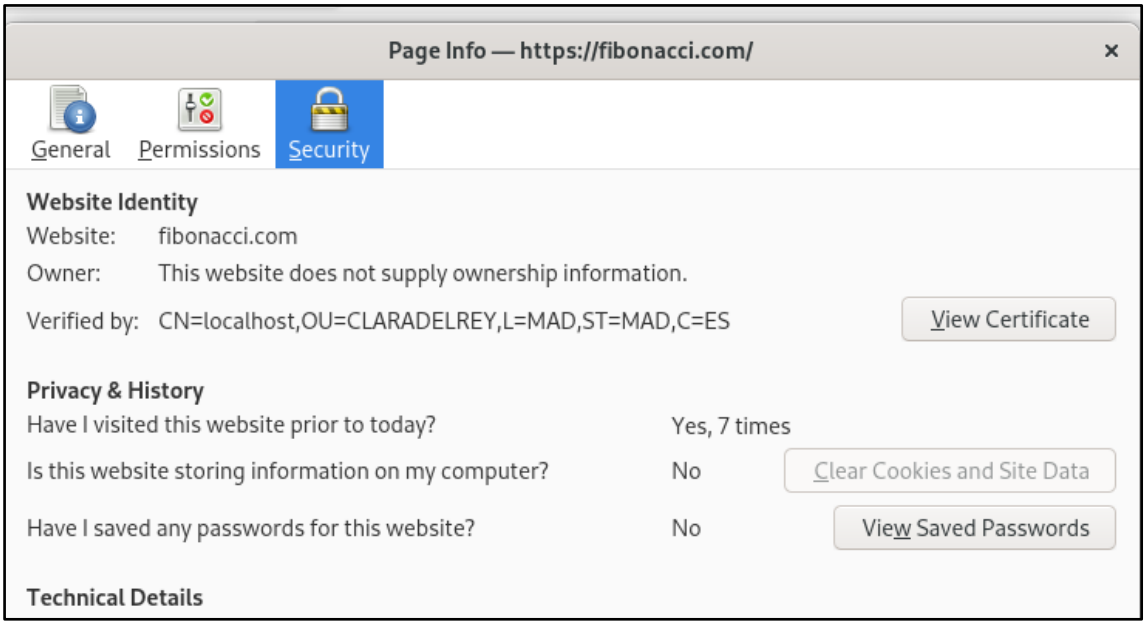
Cambiamos el archivo launch que configuramos al principio agregandole "-p 443:443". Estos mapeos hacen posible que el servidor Apache que se ejecuta dentro del contenedor Docker sea accesible desde fuera del contenedor usando los puertos estándar de HTTP (80) y HTTPS (443).

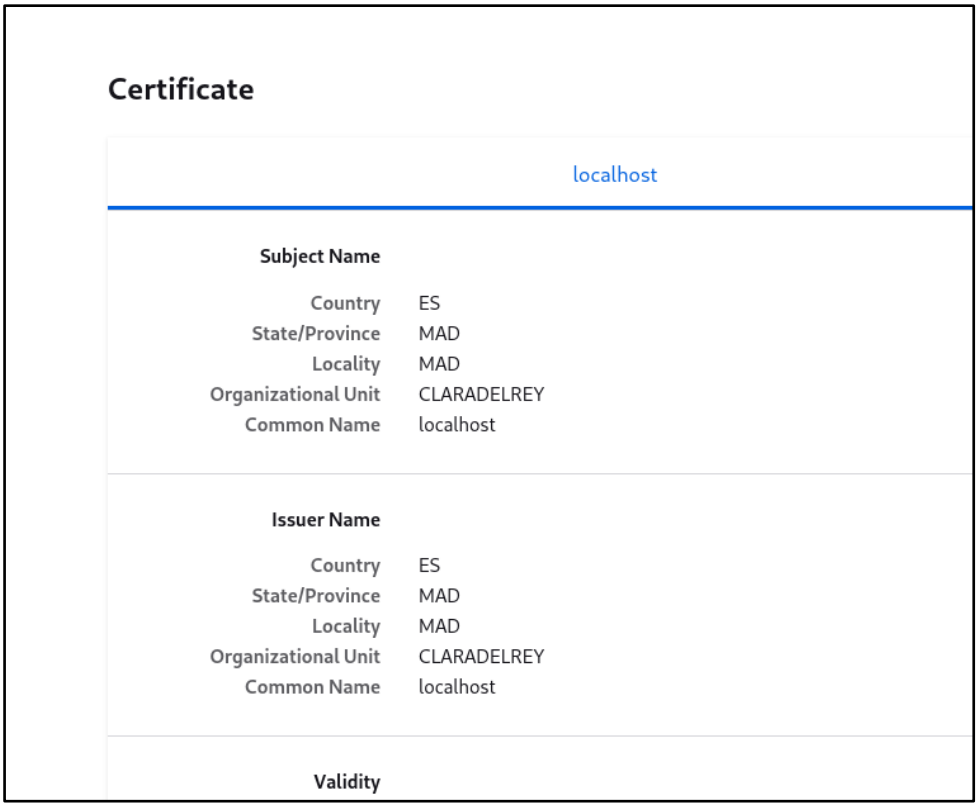
```
docker stop apachito
docker rm apachito
docker build -t img_apachito .
docker run -d -p 80:80 -p 443:443 apachito img_apachito
```

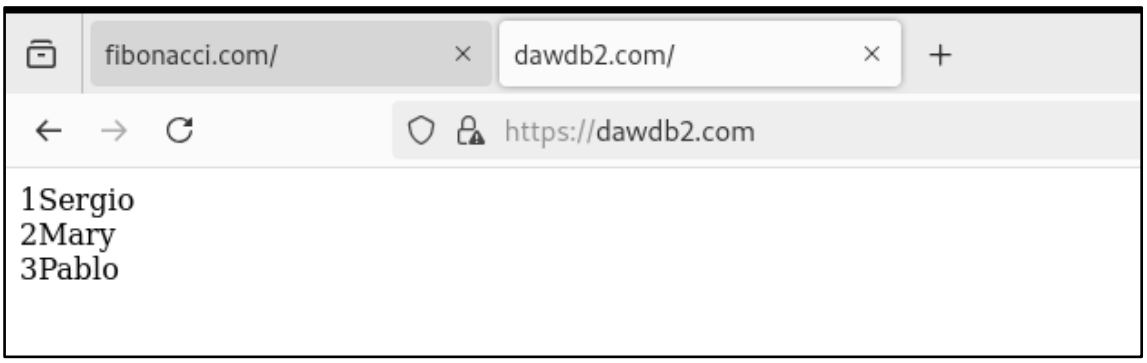
Prueba de que funciona.

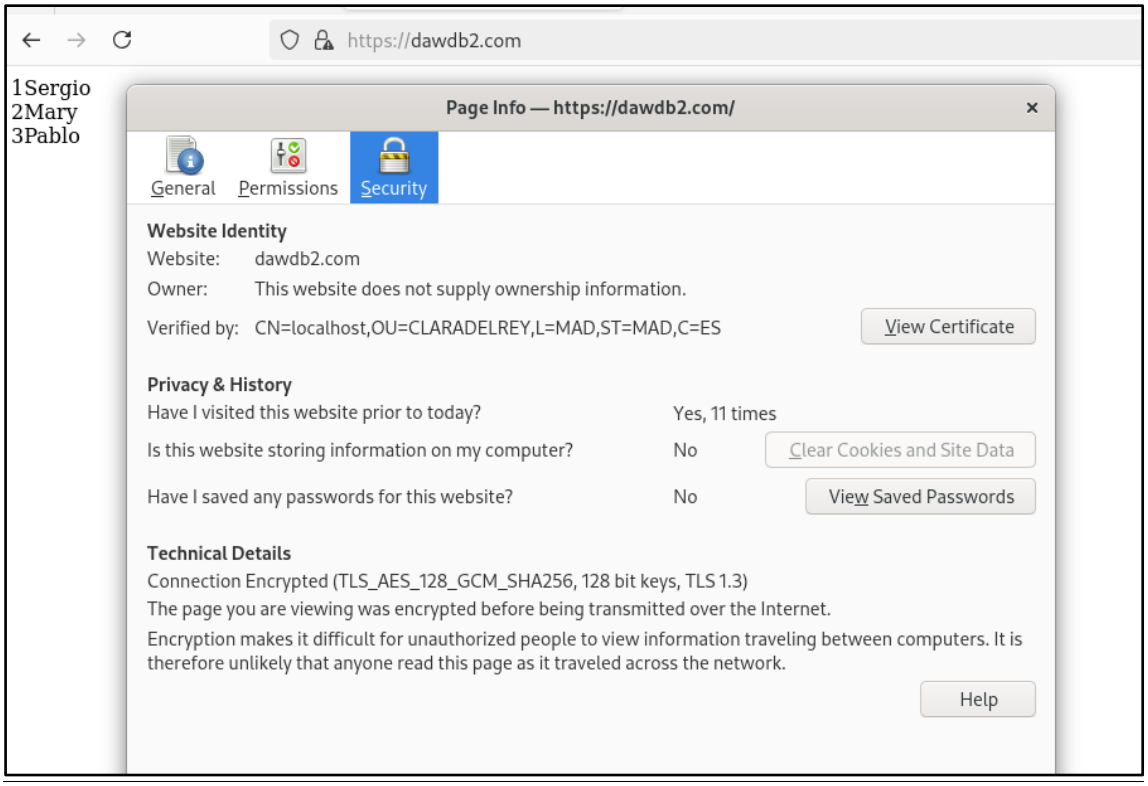
Accedemos ingresando con el nombre del dominio y vemos que nos redirige a la página utilizando el protocolo HTTPS.

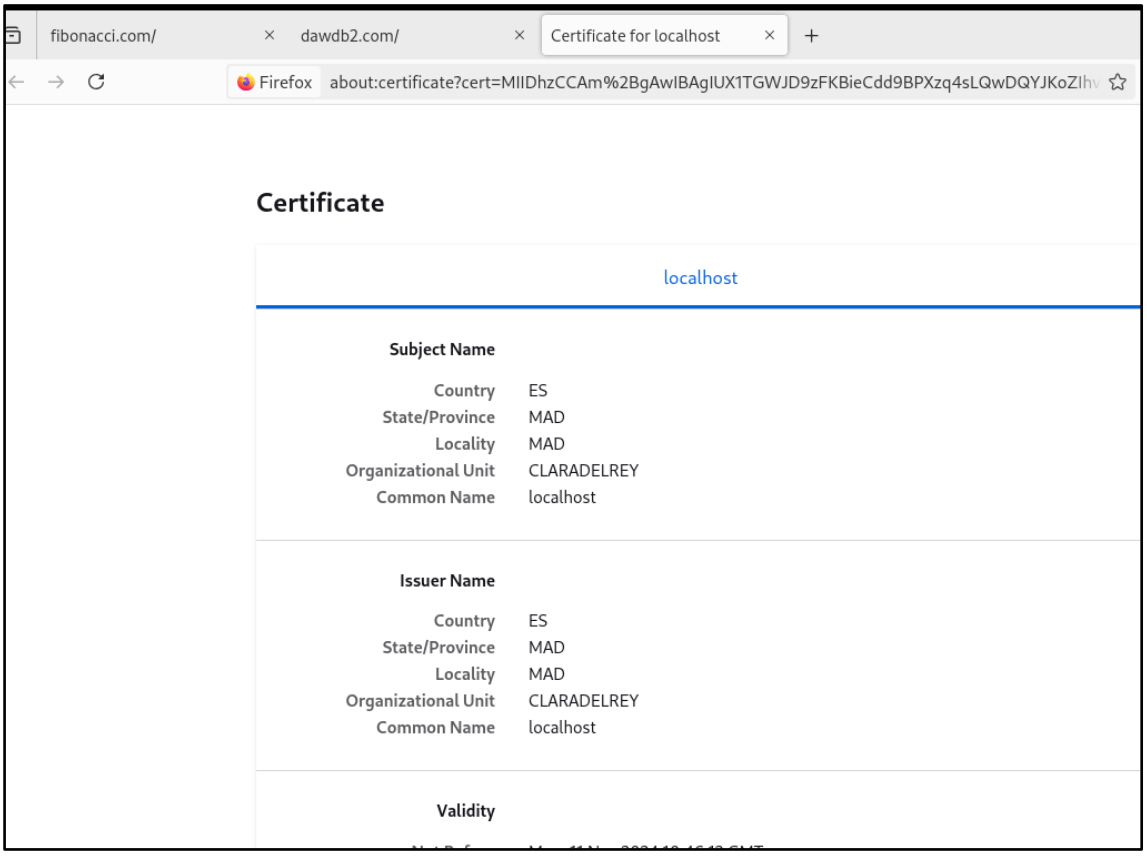





Certificate
localhost
Subject Name
Country ES
State/Province MAD
Locality MAD
Organizational Unit CLARADELREY
Common Name localhost
Issuer Name
Country ES
State/Province MAD
Locality MAD
Organizational Unit CLARADELREY
Common Name localhost
Validity


fibonacci.com/
dawdb2.com/
https://dawdb2.com
1Sergio
2Mary
3Pablo





Podemos ver que se puede acceder a ambas páginas de forma independiente, que ambas utilizan el protocolo HTTPS y que podemos acceder al certificado y visualizar los datos que ingresamos al crearlo.

Seguridad y acceso

El siguiente paso lo que va a hacer es configurar la contraseña para cuando queramos acceder a la página pida usuario y contraseña para tener más seguridad.

```
RUN apt-get update
RUN apt-get install apache2-utils
RUN a2enmod auth_basic
RUN htpasswd -bc /etc/apache2/.htpasswd proyecto 1234
```

Entonces lo que hacemos es primero actualizar los paquetes para ver que todo vaya bien y luego instalar las utilidades de apache2.

Después Habilitamos el módulo auth_basic en el servidor Apache. Esto permite que el servidor utilice el mecanismo de autenticación básica HTTP.

Y por último creamos el archivo .htpasswd con credeciales para usuario y contraseña.

-bc lo que hace es: b permite pasar la contraseña directamente en el comando en lugar de perdila de manera interactiva y c crea un nuevo archivo .htpasswd.

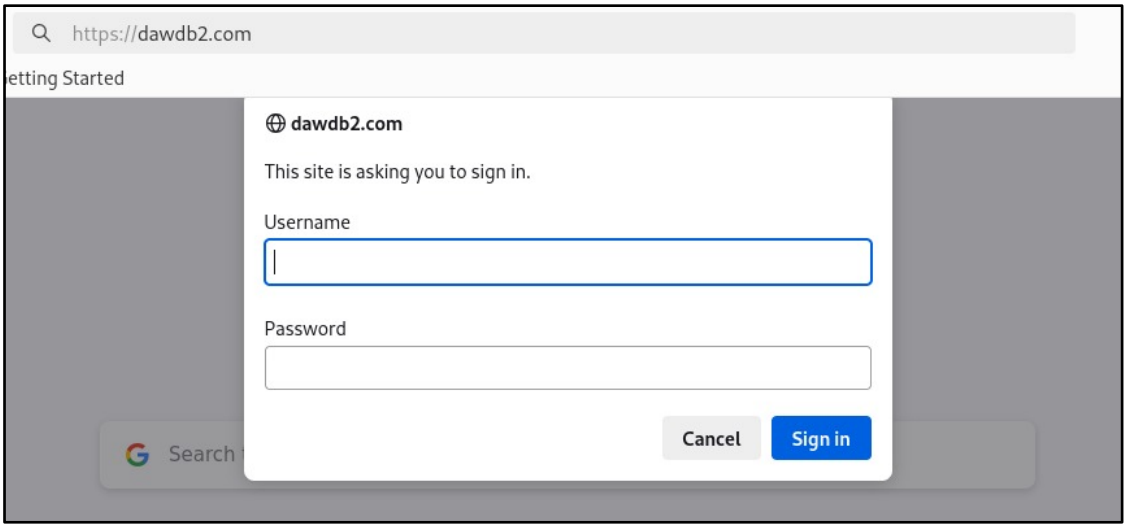
```
<Directory /var/www/dawdb2>
    AuthType Basic
    AuthName "Restricted Area"
    AuthUserFile /etc/apache2/.htpasswd
    Require valid-user
</Directory>
```

Por último, editamos el archivo de configuración dawdb2.conf para restringirlo usando autenticación básica.

Entonces hacemos un directorio con la dirección que sale y dentro de él, AuthType Basic y AuthName "Restricted Area" especifican el tipo de autenticación y el mensaje de la ventana de autenticación.

Luego AuthUserFile /etc/apache2/.htpasswd define el archivo donde se almacenan las credenciales de los usuarios autorizados.

Require valid-user permite el acceso solo a usuarios válidos.



Después de hacer todo eso, actualizamos el launch.sh y si todo va bien ponemos en el navegador la dirección https://dawdb2.com y al entrar se abrirá una ventana donde te pedirá el nombre de usuario y la contraseña.