DESPLIEGUE DE SERVIDOR WEB

Gonzalo de Saavedra, Iván Muñoz

Tras semanas de aprendizaje bajo la tutela del gran maestro Ismael Pérez Roldán, procedemos a aplicar nuestro recientemente adquirido conocimiento en un desafiante proyecto propuesto por el mismo. He aquí la documentación del proceso.

- ***Creación de Dockerfile.***

  Empezamos creando un Dockerfile, archivo a partir del cual se generará una imagen de contenedor con las características que deseemos.

  Ya que vamos a hacer que el contenedor use el puerto 80, vamos a deshabilitar las webs que teníamos asignadas en proyectos previos a este puerto preventivamente para evitar errores (a2dissite web1.conf web2.conf).

➔ Extraemos la estructura básica del Dockerfile que hicimos en días anteriores en clase:

FROM ubuntu

EXPOSE 81

RUN apt install apache2 -y

ADD index.php /var/www/html/index.html

CMD[“usr/sbin/apache2ctl”, “-D”, “FOREGROUND”]

➔ Añadimos estos comandos al archivo para instalar el mod de php:

RUN sudo apt install php

RUN sudo apt install php libapache2-mod-php

RUN sudo a2enmod php

RUN sudo systemctl restart apache2

➔ Añadimos el certificado SSL para habilitar HTTPS:

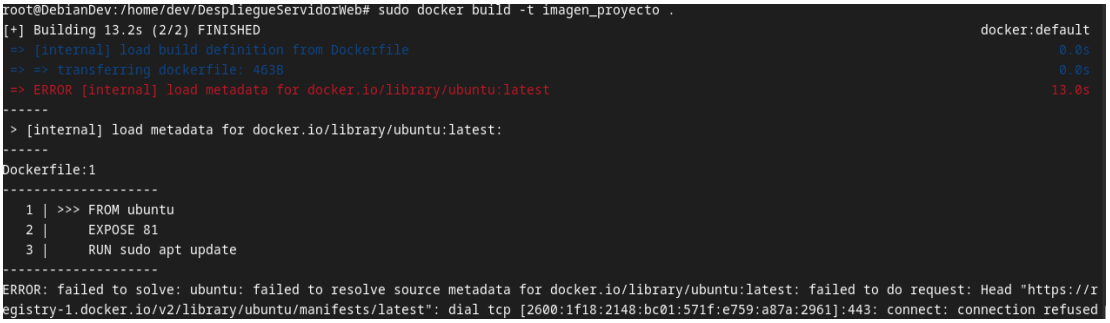
RUN openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -sha256 -days 365

- ***<u>Construir la imagen Docker a partir del Dockerfile.</u>***

  Procedemos a la creación de la imagen. Es en este momento cuando descubriremos si las instrucciones plasmadas en el Dockerfile son correctas y no generan ningún error. A lo largo de las dos siguientes páginas se detallan los errores que han surgido y las acciones que hemos tomado para solucionarlos.

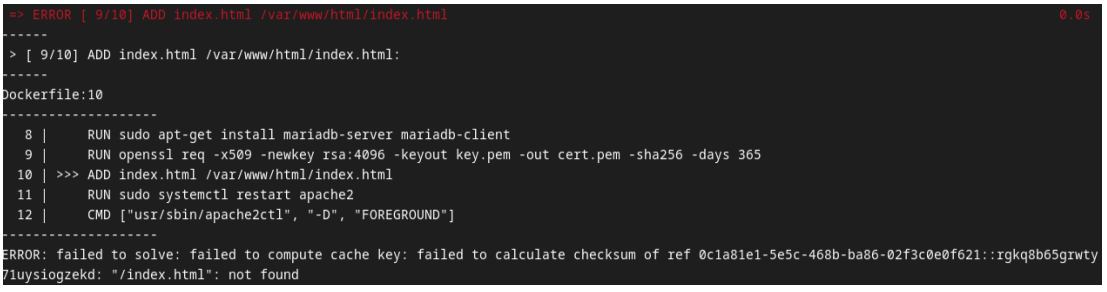
➔ **ERROR:** Al crear la imagen con el comando **docker build -t ImagenProyecto . ,** salta el error 'Invalid tag', ya que el nombre que le hemos querido contiene mayúsculas lo cual no está permitido.
SOLUCIÓN: Cambiamos el nombre a "imagen_proyecto".

➔ **ERROR:** Lo volvemos a intentar y la terminal indica que no se pueden cargar los metadatos de la imagen de Ubuntu.



Al principio creemos que docker no está pudiendo acceder al Dockerfile así que probamos a lanzar el comando desde diferentes directorios, a crear una nueva carpeta para el Dockerfile, a lanzar el comando con la ruta del Dockerfile en vez de con el punto que indica el archivo por defecto, reiniciamos el servicio de Docker…
SOLUCIÓN: No hemos encontrado solución. Este error parece un problema interno y en nuestra experiencia salta aproximadamente el 60% de las veces que intentamos crear una imagen. **Procedemos aprovechando el 40% de ocasiones.**

➔ **ERROR:** Salta un error cuando se intenta reemplazar el archivo index.html por defecto de apache por nuestro propio index.



Este error nos deja un tanto perplejos ya que en el Dockerfile que hemos creado indicamos que **nuestro index es un index.php** . Sin embargo, como se puede observar en la imagen, **Docker está buscando un index.html**, el cual no encuentra.
SOLUCIÓN (provisional): Creamos un index.html para que Docker encuentre lo que está buscando.

➔ **ERROR:** Cuando Docker intenta ejecutar comandos con sudo salta el error **"sudo: not found".** Investigando un poco aprendemos que los contenedores actúan como root por lo que no es necesario usar sudo.

```
 => ERROR [ 2/10] RUN sudo apt update
------
 > [ 2/10] RUN sudo apt update:
0.569 /bin/sh: 1: sudo: not found
------
Dockerfile:3
--------------------
   1 |     FROM ubuntu
   2 |     EXPOSE 81
   3 | >>> RUN sudo apt update
   4 |     RUN sudo apt install apache2 -y
   5 |     RUN sudo apt install php
--------------------
ERROR: failed to solve: process "/bin/sh -c sudo apt update" did not complete successfully: exit code: 127
```

SOLUCIÓN: Eliminamos todos los "sudo" del Dockerfile.

➔ **ERROR:** A pesar de haber eliminado los sudo, salta el mismo error y los sudo siguen apareciendo en la terminal. Nuestra hipótesis es simple, Docker está leyendo el Dockerfile desde el caché. Reiniciamos Docker, lo que no soluciona nada. Es entonces cuando nos damos cuenta de que habíamos estado editando el Dockerfile erróneo. Esto también explica el error del index.
SOLUCIÓN: Hacemos todos los cambios mencionados en el Dockerfile correcto.

➔ **ERROR:** php no puede ser instalado. Leyendo la terminal vemos que el proceso ha sido abortado tras no recibir un input necesario para continuar la instalación.

```
1.601 Do you want to continue? [Y/n] Abort.
------
Dockerfile:5
--------------------
   3 |     RUN apt update
   4 |     RUN apt install apache2 -y
   5 | >>> RUN apt install php
   6 |     RUN apt install libapache2-mod-php
   7 |     RUN a2enmod php
--------------------
ERROR: failed to solve: process "/bin/sh -c apt install php" did not complete successfully: exit code: 1
```

SOLUCIÓN: Ya que no se puede introducir el input manualmente durante la creación de la imagen, **añadimos -y al comando** de instalación de php para que lo pase como input (**RUN apt install php -y**).

➔ **ERROR:** El comando a2enmod php no funciona. Probamos a añadir **RUN systemctl reload apache2** en la línea anterior pero este nuevo comando tampoco se ejecuta.

```
 => ERROR [ 6/10] RUN a2enmod php
------
 > [ 6/10] RUN a2enmod php:
0.420 ERROR: Module php does not exist!
------
Dockerfile:7
--------------------
   5 |     RUN apt install php -y
   6 |     RUN apt install libapache2-mod-php
   7 | >>> RUN a2enmod php
   8 |     RUN apt-get install mariadb-server mariadb-client
   9 |     RUN openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -sha256 -days 365
--------------------
ERROR: failed to solve: process "/bin/sh -c a2enmod php" did not complete successfully: exit code: 1
```

SOLUCIÓN: Borramos la línea de a2enmod. Al parecer no era necesaria ya que el módulo se activa por defecto.

➔ **ERROR:** Docker no puede crear el certificado ssl ya que necesita parámetros que no podemos meter manualmente.
SOLUCIÓN: Editamos el comando del certificado añadiendo los parámetros:
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -sha256 -days 3650 -nodes -subj "/C=ES/ST=Madrid/L=Madrid/O=./OU=./CN=."

Por fin **la imagen se crea sin ningún error adicional.** Creamos un contenedor de prueba sin ningún problema (**docker run --name Test imagen_proyecto**).
Accedemos a la web mediante la IP indicada por Docker (tras un poquito de investigación hemos descubierto que es una ip que Docker asigna por defecto)

Rango de las IPs asignadas por Docker: (**172.17. 0.0/16**)

Se muestra correctamente nuestro index.

- ***Creando fibonacci100.com***

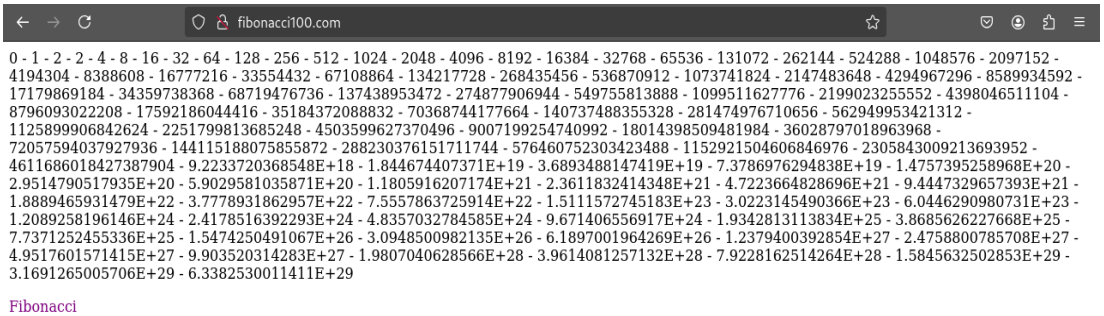
Para comprobar si funciona el php creamos la página de Fibonacci, construyendo una nueva imagen a partir de un nuevo index que incluya el código pertinente. Aprovechamos también para configurar los puertos modificando el comando (**docker run -d -p 80:80 -p 443:443 --name testpuertos fibonacci**) y modificando el archivo hosts añadiendo la línea:

127.0.0.1 fibonacci100.com

➔ **ERROR:** El puerto 80 está ocupado por lo que el comando no se ejecuta.
SOLUCIÓN: **Desactivamos apache por completo en la máquina virtual.** Ya habíamos desactivado todas las páginas que usaban el puerto 80 y eliminado todos los contenedores, pero aún así apache lo reserva para la página por defecto.

➔ **ERROR:** No se ejecuta el código php. Aunque accedemos sin problema a la página tanto desde localhost como desde fibonacci100.com, el bloque de código php no se ejecuta lo que nos hace pensar que falla algo en el Dockerfile.
SOLUCIÓN: En el dockerfile la línea:
ADD index.php /var/www/html/index.html
no hacía lo que creíamos (reemplazar el index.html por index.php), sino que solo reemplaza el contenido manteniendo la extensión html e ignorando el código php. Hemos borrado el index.html antes de reemplazarlo añadiendo una línea y modificando la mencionada:
RUN rm /var/www/html/index.html
ADD index.php /var/www/html/index.php

Probamos de nuevo y todo va como la seda (aunque está mal programada la sucesión).



- ***<u>Certificados digitales SSL</u>***

  A continuación **vamos a crear un certificado digital autofirmado**, lo que nos permitirá conectarnos a la página mediante el protocolo HTTPS a través del puerto 443.

➔ Lo primero será habilitar el módulo de SSL en el contenedor y activar la configuración HTTPS modificando el Dockerfile:

RUN a2enmod ssl
RUN a2ensite default-ssl

➔ La línea del Dockerfile que añadimos al principio crea dos archivos: **cert.pem, y key.pem** . Movemos los archivos de clave a la ubicación donde apache los buscará.

RUN mkdir -p /etc/ssl/certs /etc/ssl/private
RUN mv key.pem /etc/ssl/private/server.key
RUN mv cert.pem /etc/ssl/certs/server.crt

➔ **ERROR:** Salta el error **"AH01909: dawdb2.com:443:0 server certificate does NOT include an ID which matches the server name"**
SOLUCIÓN: Sustituimos el archivo 000-default.conf por uno con el mismo nombre que hemos creado y que usa el puerto predeterminado de HTTPS (:443):

<VirtualHost ***:443***>
ServerAdmin gonlelo04@gmail.com
DocumentRoot /var/www/html
ServerName fibonacci.com
ServerAlias www.fibonacci100.com

SSLEngine on
SSLCertificateFile /etc/ssl/certs/server.crt
SSLCertificateKeyFile /etc/ssl/private/server.key

ErrorLog ${APACHE_LOG_DIR}/fibonacci100.ssl.error.log
CustomLog ${APACHE_LOG_DIR}/fibonacci100.ssl.custom.log combined
</VirtualHost>

Lo añadimos mediante:
ADD Fibonacci/000-default.conf /etc/apache2/sites-available/000-default.conf

➔ Agregamos una redirección en el bloque `<VirtualHost *:80>` para redirigir todas las conexiones HTTP al puerto 443:

<VirtualHost *:80>
ServerAdmin gonlelo04@gmail.com
DocumentRoot /var/www/html
ServerName fibonacci100.com
ServerAlias www.fibonacci100.com

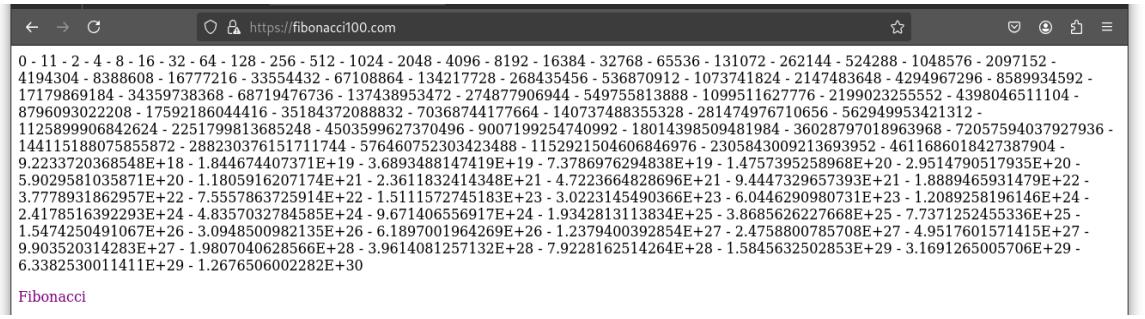
Redirect permanent / https:/fibonacci100.com/

ErrorLog ${APACHE_LOG_DIR}/fibonacci100.error.log
CustomLog ${APACHE_LOG_DIR}/fibonacci100.custom.log combined
</VirtualHost>

En la configuración del VirtualHost del puerto 80 no son imprescindibles tantos parámetros, pero como no sabemos cuáles son necesarios, lo copiamos todo.

➔ Sustituimos el archivo de configuración en el Dockerfile:
ADD Fibonacci/000-default.conf /etc/apache2/sites-available/000-default.conf
➔ **ERROR:** Salta el error: "**[ssl:warn] [pid 211] AH01906: dawdb2.com:443:0 server certificate is a CA certificate (BasicConstraints: CA == TRUE !?)**"
SOLUCIÓN: Reiniciamos todo.

Todo **funciona bien** y se puede acceder mediante ambos puertos.



- ***Instalación y configuración de MariaDB***

  Para empezar a configurar MariaDB creamos una nueva página siguiendo las instrucciones. Mediante el Dockerfile, añadimos los directorios y archivos necesarios (dawdb2.conf, index.php) siguiendo la misma estructura que con Fibonacci, solo que ahora no podemos usar la carpeta por defecto por lo que creamos una nueva (/var/www/dawdb2). Modificamos también /etc/hosts.

➔ Primero creamos un archivo sql (init.sql) que crea una tabla.
➔ Añadimos al dockerfile:
RUN apt-get install -y php-mysql
RUN apt-get install -y mariadb-server
COPY init.sql /docker-entrypoint-initdb.d/

Ejecutamos: **run -d -p 80:80 -p 443:443 --name pruebadb dawdb2.**
(dawdb2 es el nombre que le hemos dado a la imagen trabajando con mariadb)

➔ **ERROR:** Hay algún error con la base de datos y su configuración. Parece que no se está estableciendo una conexión. No hay mensaje de error.
Probamos a añadir el archivo init.sql a var/sql/ en el dockerfile con la línea

ADD dawdb2/init.sql /var/sql/

Probamos a cambiar el CMD para hacer referencia a archivo sql y a MariaDB.

Probamos a instalar librerías de MariaDB en el Dockerfile.

Probamos a instalar todos los mods de PHP habidos y por haber.

Probamos a usar el puerto 3306 (puerto por defecto de MariaDB).

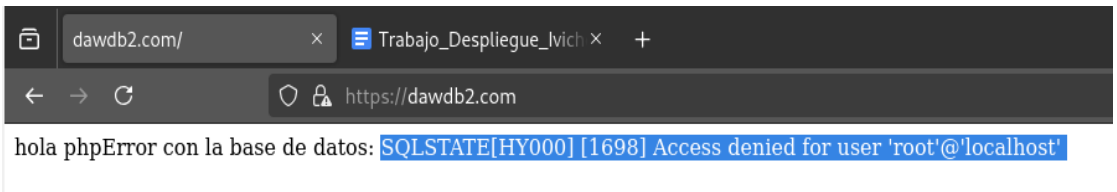
Probamos a sustituir el ports.conf haciendo que escuche el puerto 3306.

Probamos a cambiar la línea de conexión cambiando el nombre del host (probamos "localhost", la IP del contenedor, la IP local…).

Todo da error, nada funciona y tras horas de andar en la penumbra metemos la conexión a la base de datos en un try-catch, cosa que deberíamos haber hecho mucho antes.

➔ **ERROR: SQLSTATE[HY000] [2002] Connection refused**
SOLUCIÓN: Tras quitarnos horas de vida este error desapareció por sí solo de un día para otro. Agradecidos, continuamos.

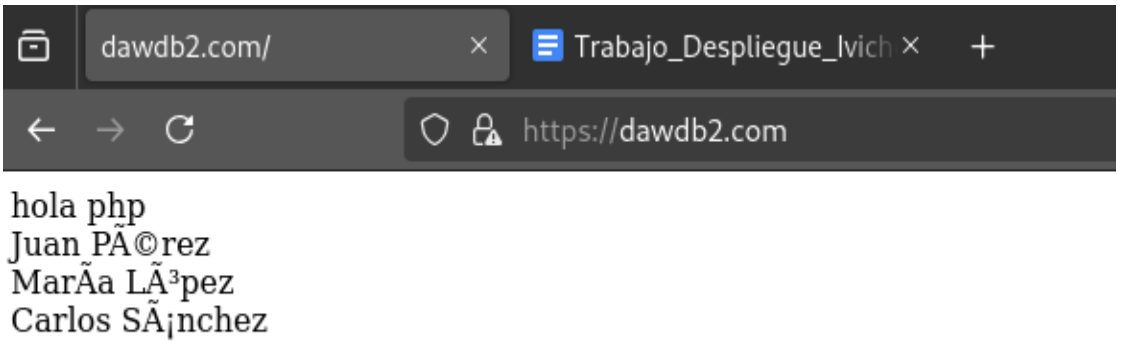
➔ **ERROR:** Salta el error: **SQLSTATE[HY000] [1698] Access denied for user 'root'@'localhost'**.



SOLUCIÓN: Configuramos los permisos de la base de datos añadiendo la siguiente línea al principio del archivo sql:
ALTER USER 'root'@'localhost' IDENTIFIED BY '';

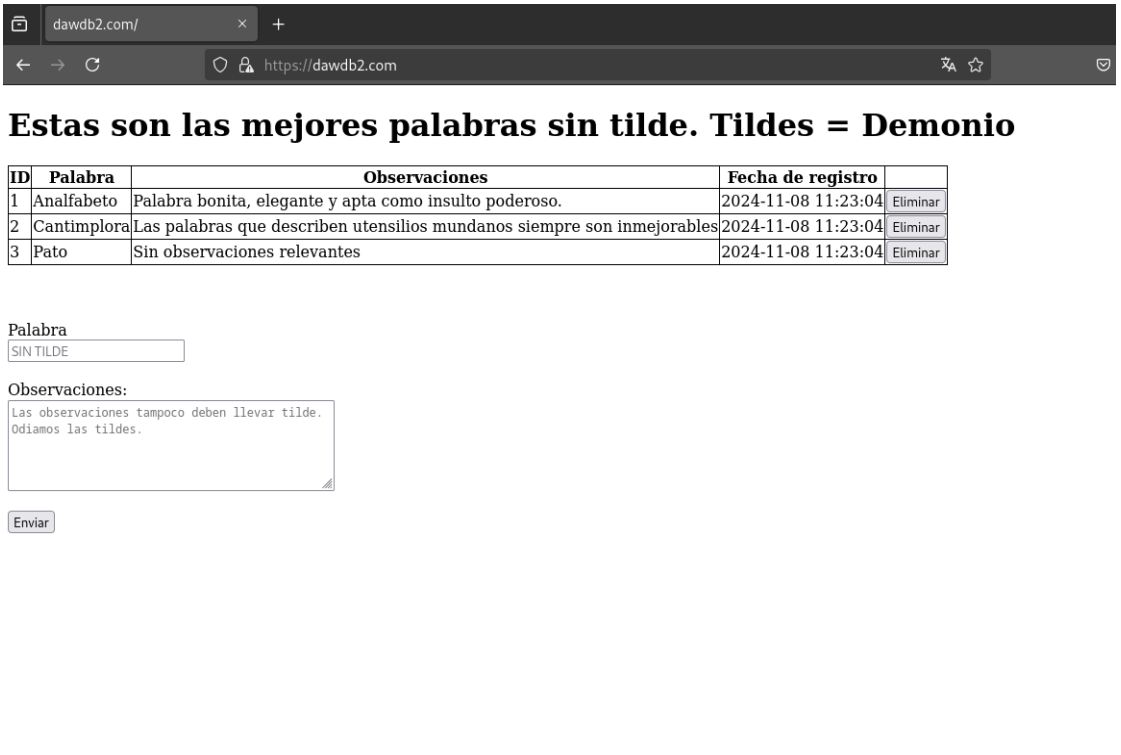
Lo cierto es que en este proceso nos olvidamos de documentar la mayoría de los errores que surgían, por no hablar de nuestros fútiles intentos de solucionarlos. Esto se debe en gran parte a un estado de enajenación y pesadumbre sin parangón. Al final **los cambios más importantes son los realizados en el Dockerfile y se explicarán en más detalle en el análisis línea por línea de este.**

Al final lo conseguimos. Las horas se han ido para siempre pero por lo menos tenemos algo a cambio. Una base de datos MariaDB funcional:



➔ **INCONVENIENTE:** Las tildes no se muestran de manera correcta.
SOLUCIÓN: Para probar las funcionalidades de MariaDB **creamos un programa que imprime las entradas de una base de datos** en la que se muestran las mejores palabras sin tilde. Esta temática tan específica viene dada por el fenómeno observado en la anterior captura, las tildes no se muestran correctamente. Tras un buen rato intentando configurar correctamente el html, el init.sql, la conexión a la base de datos… Decidimos que esta es una mejor solución.

El programa también permite eliminar filas y añadir nuevas palabras (siempre que no lleven tilde) al instante. Las pruebas son un éxito.



- ***<u>Mecanismo de autenticación</u>***

  Para crear un mecanismo de autenticación añadimos una nueva página (restringido.php) y la hacemos accesible mediante un link en la página de Fibonacci.

  Mediante un ajuste en la configuración (000-default.conf) conseguimos que la página se muestre restringida:

Sin embargo vamos a hacer que sea accesible mediante un usuario y contraseña:

➔ Añadimos la siguiente línea al Dockerfile que instala utilidades adicionales de apache: **RUN apt-get update && apt-get install -y apache2-utils**

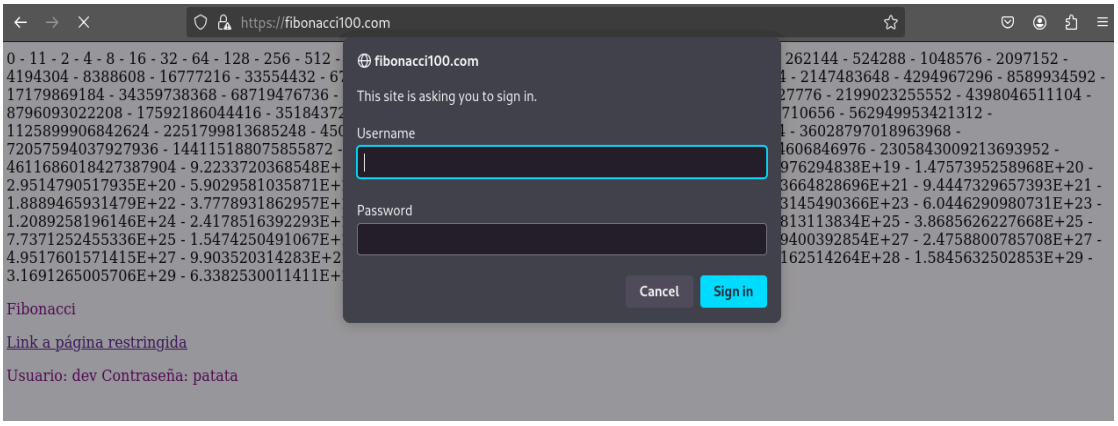
➔ Añadimos también la línea que establece el usuario y la contraseña para acceder al contenido restringido: **RUN htpasswd -cb /etc/apache2/.htpasswd dev patata**

➔ Por último modificamos el VirtualHost de Fibonacci para establecer los permisos mediante <Directory>:

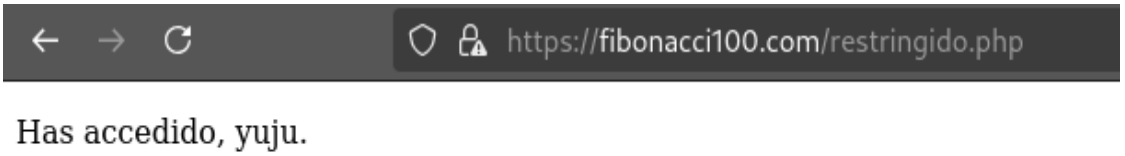
```
<Directory /etc/apache2>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/html/restringido.php>
        AuthType Basic
        AuthName "Restricted Content"
        AuthUserFile /etc/apache2/.htpasswd
        Require valid-user
</Directory>
```

Al presionar el link el navegador te pide un usuario y una contraseña:



Al meter los datos correctos, puedes acceder a la página restringida:



- ***Explicación línea a línea del Dockerfile***

```
#Indica que la imagen se va a crear a partir de Ubuntu
FROM ubuntu
#Documenta los puertos que se van a utilizar
EXPOSE 80 443
#Actualiza los paquetes y sus versiones de Ubuntu
RUN apt update
#Instala apache
RUN apt install apache2 -y
#Instala PHP
RUN apt-get install -y php libapache2-mod-php
#Instala utilidades adicionales para la autenticación
RUN apt-get update && apt-get install -y apache2-utils
#Instala muchas extensiones de PHP para la base de datos
RUN apt-get install -y php-pdo php-pdo-mysql php-mbstring php-xml php-curl
#Instala MariaDB
RUN apt-get install -y mariadb-server
#Genera el certificado y la clave SSL para HTTPS
```

```
RUN openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem
-sha256 -days 3650 -nodes -subj
"/C=ES/ST=Madrid/L=Madrid/O=./OU=./CN=dawdb2.com"
#Crea un directorio donde alojar la página dawdb2.com
RUN mkdir /var/www/dawdb2
#Sustituye el archivo de configuración por defecto que hemos usado para
#fibonacci.com  por uno personalizado que incluye los ajustes SSL
ADD Fibonacci/000-default.conf /etc/apache2/sites-available/000-default.conf
#Añade la página restringida accesible a través de fibonacci100.com
ADD Fibonacci/restringido.php /var/www/html
#Establece un usuario y contraseña para la página restringida
RUN htpasswd -cb /etc/apache2/.htpasswd dev patata
#Añade el index.php de dawdb2.com
ADD dawdb2/index.php /var/www/dawdb2
#Añade la configuración VirtualHost de dawdb2.com
ADD dawdb2/dawdb2.conf /etc/apache2/sites-available/dawdb2.conf
#Borra el index.html de la página por defecto de apache
RUN rm /var/www/html/index.html
#Añade el index.php de fibonacci100.com en su lugar
ADD Fibonacci/index.php /var/www/html/index.php
#Añade el archivo sql que crea la base de datos a la carpeta correspondiente
ADD dawdb2/init.sql /var/sql/
#Activa el mod SSL de apache (instalado por defecto)
RUN a2enmod ssl
#Activa la configuración de apache para el protocolo HTTPS
RUN a2ensite default-ssl
#Crea un directorio donde alojar la clave del certificado digital
RUN mkdir -p /etc/ssl/certs/etc/ssl/private
#Mueve la clave del certificado digital al directorio creado
RUN mv key.pem /etc/ssl/private/server.key
#Mueve el certificado digital a la carpeta de certificados
RUN mv cert.pem /etc/ssl/certs/server.crt
#Activa la página dawdb2.com
RUN a2ensite dawdb2
#Se asegura de iniciar mariadb desde nuestro archivo sql cuando se inicia el
#contenedor entre otras cosas
CMD [ "/bin/bash", "-c", "service mariadb start && mariadb < /var/sql/init.sql &&
/usr/sbin/apache2ctl -D FOREGROUND" ]
```