Infraestructura Web Contenerizada con Docker: Despliegue, Automatización y Mantenimiento para PYMES.

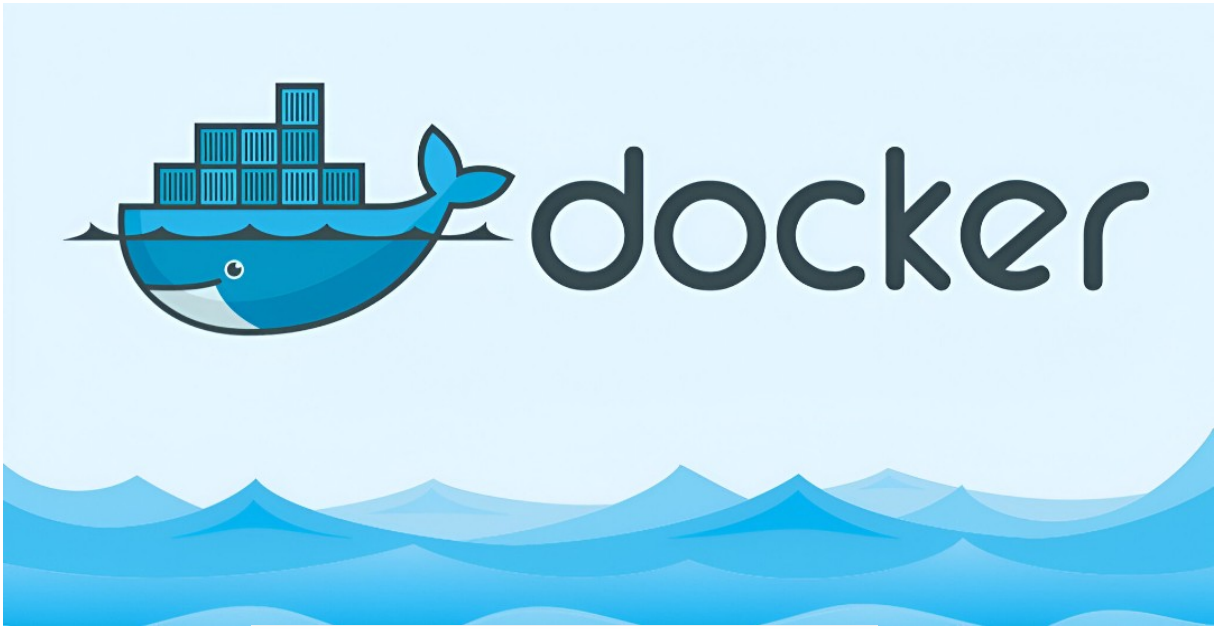


Figura 1: Portada

LaVEREDA | ESO BATXILLERAT CICLES

Autor: Jose Gavilán Tortajada **Grado:** Administración de Sistemas Informáticos en Red
Curso: 2024-2025 **Centro:** IES La Vereda **Tutor:** Marc Crespo

Resumen

Castellano:

Este Trabajo de Fin de Grado (TFG) presenta una infraestructura informática diseñada para cubrir de forma sencilla, segura y eficiente las necesidades básicas de pequeñas y medianas empresas (PYMES). El sistema está construido con Docker y Docker Compose, e integra un servidor web con una aplicación de gestión de stock, un servidor DNS privado, una base de datos persistente y un sistema de copias de seguridad automáticas en la nube (Dropbox), todo gestionado de forma remota y segura mediante acceso SSH.

El objetivo es ofrecer una solución completa, fácil de poner en marcha desde el primer momento, que ayude a las PYMES a evitar los problemas típicos que suelen tener con sus sistemas: desorganización, falta de seguridad o herramientas poco claras. Con esta infraestructura se busca mejorar el control del inventario de forma visual y sencilla, sin que la empresa necesite tener conocimientos técnicos avanzados. Además, este proyecto está planteado como una base real para poder ofrecer servicios informáticos como autónomo a empresas que busquen sistemas prácticos y adaptados a sus necesidades reales.

RESUM

Valenciano:

Este Treball de Fi de Grau (TFG) presenta una infraestructura informàtica dissenyada per cobrir de manera senzilla, segura i eficient les necessitats bàsiques de petites i mitjanes empreses (PIMES). El sistema està construït amb Docker i Docker Compose, i integra un servidor web amb una aplicació de gestió d'estoc, un servidor DNS privat, una base de dades persistent i un sistema de còpies de seguretat automàtiques al núvol (Dropbox), tot gestionable de manera remota i segura mitjançant accés SSH.

L'objectiu és oferir una solució completa, fàcil de posar en marxa des del primer moment, que ajude les PIMES a evitar els problemes típics que solen tindre amb els seus sistemes: desorganització, manca de seguretat o eines poc clares. Amb esta infraestructura es busca millorar el control de l'inventari d'una manera visual i senzilla, sense que l'empresa haja de tindre coneixements tècnics avançats. A més, este projecte està plantejat com una base real per poder oferir serveis informàtics com a autònom a empreses que busquen sistemes pràctics i adaptats a les seues necessitats reals.

ABSTRACT
Inglés:

This Final Degree Project (TFG) presents an IT infrastructure designed to cover the basic needs of small and medium-sized businesses (SMEs) in a simple, secure, and efficient way. The system is built using Docker and Docker Compose, and includes a web server with a stock management app, a private DNS server, a persistent database, and an automatic cloud backup system (Dropbox), all manageable remotely and securely through SSH access.

The goal is to offer a complete solution that is easy to set up from the start, helping SMEs avoid common problems in their systems, such as disorganization, lack of security, or unclear tools. This infrastructure aims to improve inventory control in a visual and simple way, without needing advanced technical knowledge. Additionally, this project is designed as a real base to offer IT services as a freelancer to companies looking for practical systems tailored to their real needs.

Índice

1 Introducción

1.1 Descripción del proyecto

Empecemos hablando del contexto del proyecto, las razones del desarrollo de este y los fines que quiero conseguir.

- **Contextualización:** Las PYMES son un pilar fundamental de la economía, pero a menudo presentan carencias en la implementación de tecnologías y infraestructuras informáticas robustas y seguras. Es frecuente encontrar gestiones de inventariado basados en sistemas manuales o herramientas básicas como hojas de calculo, lo que genera in-eficiencias operativas aumentando la probabilidad de errores ademas de vulnerabilidades de seguridad. La limitación de recursos especializados y presupuestos ajustados hace complicado la inversión en soluciones profesionales, dejando a muchas PYMES expuestas a riesgos que pueden comprometer su viabilidad.
- **Justificación:** El proyecto nace al observar la necesidad de las PYMES de soluciones tecnológicas accesibles y eficientes. La razón principal de este trabajo es proporcionar una alternativa integral, diseñada para resolver desafíos técnicos actuales y, al mismo tiempo, potenciar la eficiencia operativa y la seguridad digital.

Durante mi día a día, veía como en pleno 2024 aun hay tiendas que tienen su sistema de gestión a papel, así pues se me ocurrió una manera sencilla de modernizar su sistema de gestión con la ayuda de contenedores Docker, permitiendo así que cualquier persona lo tenga accesible sin necesidad de tener unos recursos demasiado buenos. Además decidí automatizar este proceso para poder implantarlo a diferentes empresas modificando algunos simples campos.

2 Estructura del proyecto

2.1 Plan de trabajo

El plan de trabajo será indicado en un diagrama de Gantt (diagrama de gantt, s. f.), ya que esta es una herramienta que permite gestionar un proyecto ilustrando el trabajo que se va a realizar durante un periodo de tiempo, incluyendo las fechas de iniciación el 08 de abril y de finalización el 1 de Junio, además de la persona encargada de realizar cada tarea entre muchas otras funciones.

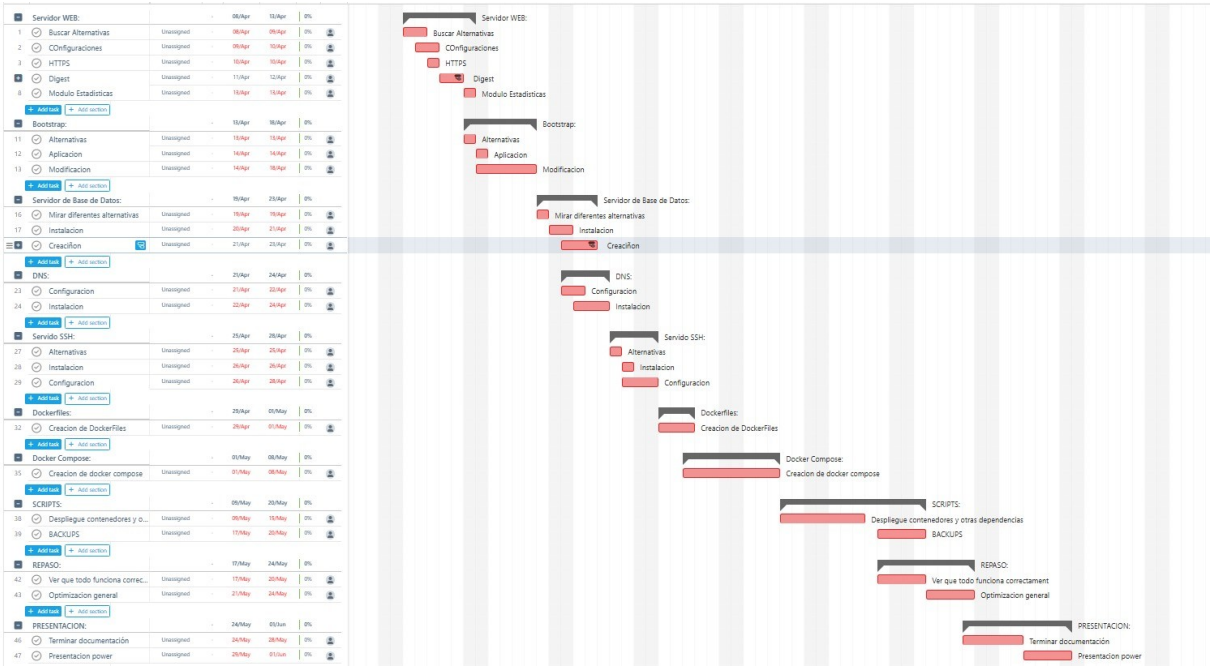


Figura 1: Diagrama de GANTT

2.2 Diseño de infraestructura de red

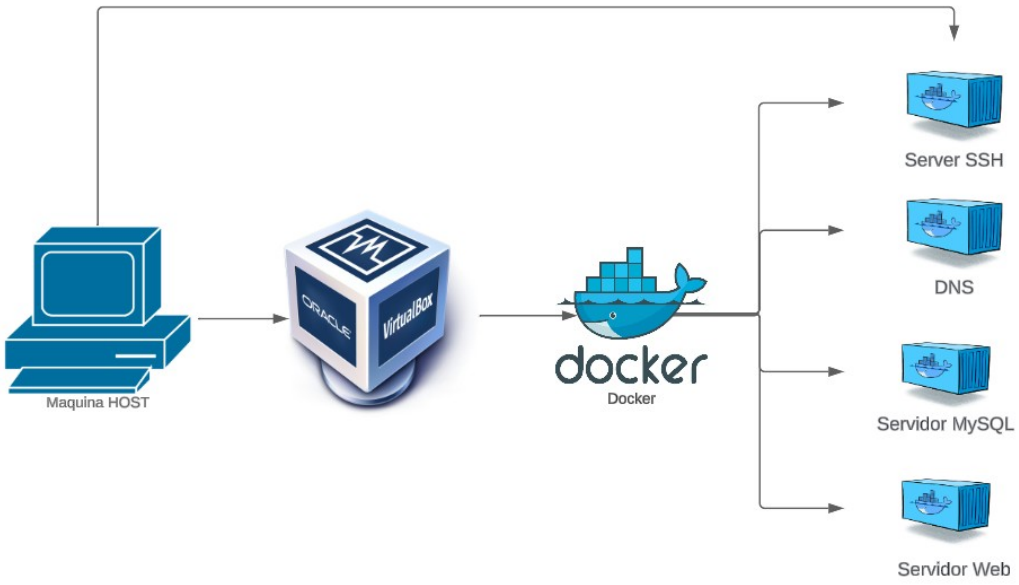


Figura 2: Infraestructura de red

La estructura de red que tiene el proyecto se compone de una red creada específicamente en docker, esta permitirá añadir nuevos contenedores con nuevas IP fijas si fuera necesario en un futuro, pero para el proyecto partiremos de las 4 siguientes.

TABLA DE IP	
DNS	192.168.100.2
Apache	192.168.100.3
Base de datos	192.168.100.4
Manager	192.168.100.5

2.3 Viabilidad y proyección del proyecto

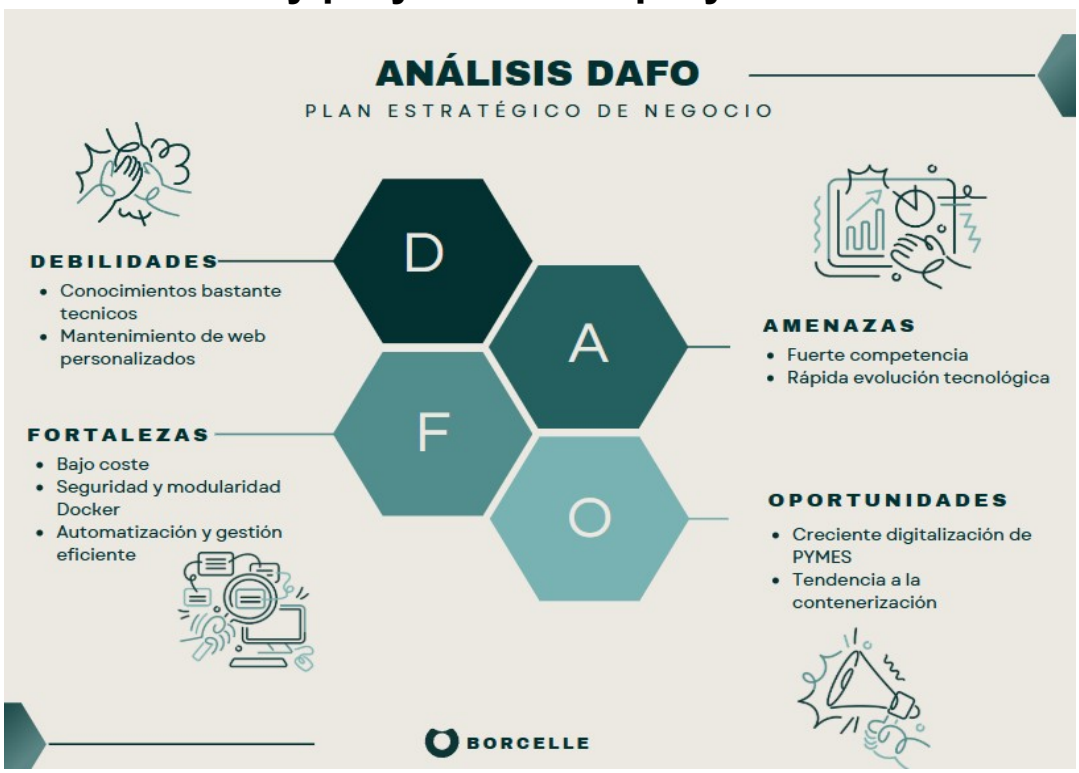


Figura 3: Análisis Dafo

En cuanto al estudio de costes, este proyecto no necesita una inversión inicial elevada ni recursos complicados, ya que la mayor parte del trabajo se lleva a cabo de forma autónoma utilizando herramientas accesibles. La principal fuente de ingresos provendrá de la gestión y el mantenimiento continuo del servicio, asegurando su buen funcionamiento y la implementación de mejoras progresivas con el tiempo.

El análisis DAFO de este proyecto pone en evidencia fortalezas como el desarrollo autónomo y el bajo coste de mantenimiento, junto con oportunidades como la creciente demanda de herramientas accesibles. Entre las debilidades destacan los recursos limitados para promoción y la necesidad de que el proyecto crezca de forma natural. Como amenazas, se identifican la competencia ya establecida y posibles obstáculos tecnológicos o legales.

Además, una de las cosas que me gustaría en un futuro, si todo funcionase bien y el proyecto creciera como debería, sería poder mejorar el servicio para que también pueda ser utilizado por personas con alguna discapacidad, como por ejemplo la ceguera. No lo enfoco tanto como “impacto social”, sino más bien como un paso lógico hacia hacer la herramienta más accesible para todo el mundo.

2.4 Hardware y Software

Este apartado es fundamental para saber cuáles son las características mínimas necesarias para poder ejecutar estos servicios en las máquinas.

En principio, para instalar un sistema de Docker, las características mínimas que debe tener cualquier sistema son las siguientes:

- Una **arquitectura de 64bits** en el sistema operativo
- Una capacidad de **memoria RAM de 2GB** como mínimo para que sea efectivo y **4 GB** para sacar un rendimiento más optimo.
- Respecto al **procesador** no hay mínimo, solamente se **recomienda que sea multinúcleo** ya que es preferible para las multitareas.
- No hay un mínimo tampoco en el **espacio de disco,** pero es recomendable tener unos 20GB de memoria para almacenar imágenes, volúmenes y otros tipos de datos.
- Respecto a la compatibilidad con los diferentes SO, es recomendable utilizar versiones nuevas y estables de sistemas linux

Como podemos comprobar las características necesarias para poder utilizar un Docker en una máquina son insignificantes, ya que en la actualidad esos requisitos los cumple cualquier pc y no sería necesario hacer una gran inversión para poder adquirir un equipo en caso de no tener uno.

3 DOCKER

¿Qué es Docker?

Docker es una plataforma de código abierto que permite crear y gestionar contenedores, los cuales se utilizan habitualmente para ejecutar microservicios o procesos específicos dentro de una aplicación mayor. Estos contenedores son una forma de virtualización a nivel de sistema operativo.

Lo que los distingue es que incluyen únicamente lo necesario para funcionar, compartiendo el kernel del sistema anfitrión. Esto reduce su tamaño a unos pocos megabytes y permite un arranque mucho más rápido, lo que los hace más ligeros y portables que una máquina virtual.

Para este proyecto decidí utilizar Docker en lugar de máquinas virtuales, ya que ambas opciones permiten alcanzar objetivos similares, pero Docker ofrece ventajas más acordes con el enfoque planteado. Por ejemplo, su menor consumo de recursos permite ejecutar varios contenedores a la vez sin sobrecargar el sistema.

También destaqué su portabilidad y eficiencia, ya que los contenedores garantizan el mismo comportamiento sin importar el sistema operativo. A diferencia de las máquinas virtuales, que requieren un sistema operativo completo, Docker mejora el rendimiento general.

Por último, herramientas como **Docker Compose** o **Dockerfile** facilitan notablemente la creación y despliegue de contenedores, lo cual se detallará más adelante.

En conclusión, aunque contenedores y máquinas virtuales pueden parecer similares, es fundamental entender sus diferencias, ya que su funcionamiento y uso son distintos.

3.1 Dockerfiles

Los dockerfile son una herramienta que el propio Docker brinda para crear imágenes de un contenedor. A pesar de que podemos crear una imagen guardando el contenedor que hemos editado, podemos crear una imagen mediante dockerfile. Esto nos permite que podamos añadirle diferentes características como que cuando lancemos dicha imagen, esta ejecute un servicio nada más iniciarlo.

Además, nos permite almacenar las imágenes como código lo que hace que su peso sea mucho menor hasta el momento de su lanzamiento.

En resumen, los Dockerfile son archivos de texto que tienen una serie de instrucciones para crear la imagen de un contenedor.

Así pues, en el desarrollo de este he necesitado crear 3 dockerfile que mas tarde se montaran con docker compose.

La manera en que hago estos archivos es la siguiente. Primero uso una imagen por defecto, hago todas las instalaciones y configuraciones de archivos necesarios para cada servicio, y una vez este contenedor funciona correctamente, selecciono los archivos que he utilizado, y los guardo en una carpeta para crear este dockerfile, así pues, más adelante explicare el proceso de creación de estos contenedores.

Para realizar este proceso donde seleccionamos los archivos del contenedor y los guardamos en nuestro equipo, sera con la orden:

docker cp <id_contenedor>:/dirección/archivo /dirección/a/copiar

De esta manera, yo saque los archivos de configuración necesarios para crear el dockerfile de cada servicio.

```
FROM debian:bullseye
RUN apt update
RUN apt install -y apache2 php libapache2-mod-php php-mysql
RUN apt install -y nano iproute2 iputils-ping openssl openssh-server rsyslog

COPY bootstrap /var/www/html/bootstrap
RUN chmod -R 755 /var/www/html/bootstrap

COPY intranetHTTPS.conf /etc/apache2/sites-available/intranetHTTPS.conf
COPY ports.conf /etc/apache2/ports.conf
COPY .htdigest /etc/apache2/.htdigest
RUN echo "ServerName PROYECTO" >> /etc/apache2/apache2.conf

RUN chmod 600 /etc/apache2/.htdigest
RUN chown www-data:www-data /etc/apache2/.htdigest
RUN a2ensite intranetHTTPS
RUN a2enmod auth_digest

RUN a2enmod lua
COPY status.conf /etc/apache2/mods-available/status.conf
COPY server-status.lua /etc/apache2/server-status.lua

COPY server.crt /etc/ssl/certs/server.crt
COPY server.key /etc/ssl/certs/server.key
RUN a2enmod ssl

RUN mkdir /var/run/sshd
RUN echo 'root:1234' | chpasswd
RUN echo "PermitRootLogin yes" > /etc/ssh/sshd_config.d/custom-login.conf && \
    echo "PasswordAuthentication yes" >> /etc/ssh/sshd_config.d/custom-login.conf && \
    echo "AllowUsers root@192.168.100.5" >> /etc/ssh/sshd_config.d/custom-login.conf
RUN chmod 644 /etc/ssh/sshd_config.d/custom-login.conf
RUN touch /var/log/auth.log && chmod 640 /var/log/auth.log
EXPOSE 22

CMD ["sh", "-c", "/usr/sbin/rsyslogd -n & /usr/sbin/sshd -D & apachectl -D FOREGROUND & tail
-F /var/log/auth.log"]

EXPOSE 443
```

Unos de los comandos básicos de este lenguaje que podemos ver, son los siguiente:

FROM – sirve para definir la imagen, debian 12 bullseye, a utilizar

RUN – le indicamos las ordenes de instalación, ejecución, cambios de permisos, ...

CMD – como run, pero se ejecuta después de inicializar el contenedor, se ejecuta por defecto en /bin/sh -c

COPY – con este podemos copiar los archivos que hemos seleccionado antes para añadirlos en la ubicación deseada al crear la imagen

EXPOSE - habilitamos el puerto que deseemos en este caso yo he habilitado el 443 para https

El comando CMD está programado para que nada más el contenedor Docker arranque se inicialice el servicio de apache gracias a apachectl, que se utiliza para gestionar el servidor apache y con –D FOREGROUND se indica que el contenedor arranque el servicio de apache manteniendo este proceso en primer plano.

Por otro lado hay que añadir que también instalamos directamente el servidor ssh y hacemos la configuración “manualmente” en vez de ofrecerle un archivo ya configurado para ver diferentes formas. En este caso es permitir que se conecte un tercero como root, si pudiese, pero solo podría desde la dirección de la maquina 192.169.100.5 (la manager que se utiliza para conectarse a ella y gestionar el resto de contenedores). Además le damos los permisos necesarios entre otros.

3.2 Docker Compose

Docker compose (Docker compose, s. f.) al igual que dockerfile, es una herramienta complementaria de Docker, pero este a diferencia de los dockerfile, este permite definir las redes y los volúmenes a utilizar, entre otros como también le podemos decir que lance un dockerfile indicando a este como imagen con lo que creara el contenedor. Básicamente sirve para especificar como queremos lanzar una imagen.

Cabe resaltar que esta herramienta hay que instalarla, ya que su formato de archivo YAML si que podremos crearlo y modificarlo, pero no podremos ejecutarlo hasta que se instale. Con un simple sudo apt install docker-compse ya nos permitiría lanzar comandos relacionados con esta herramienta.

Una vez tengamos el archivo YAML configurado y queramos ejecutarlo sería tan fácil como utilizar el comando sudo docker-compose up –d, el –d es opcional ya que realmente sirve para que se desplieguen con el Docker compose se ejecuten en segundo plano permitiendo liberar la terminal y el servicio quede activo.

```
services:
  apache:
    build:
      context: ./apache
      dockerfile: dockerfile
    networks:
      redbridge:
        ipv4_address: 192.168.100.3
```

Esto es parte del Docker compose que he utilizado. Como se aprecia, al principio se define la versión que se utilizará, y luego se definen los servicios a utilizar, cada uno con sus características.

4 Configuración de servicios

4.1 SERVIDOR WEB

Un servidor web (*Servidor Web*, s. f.) tiene como finalidad almacenar y transmitir el contenido solicitado de un sitio web al navegador de los clientes.

Para configurar el servidor web, estuve entre dos opciones, **NGINX y APACHE**, ambos son servidores web con funcionalidades similares pero cada uno tiene sus diferencias, ya que en términos generales NGINX es algo mejor que Apache si de rendimiento y eficiencia se habla, pero al final me decidí por escoger Apache.

NGINX tiene buenas características como su alto rendimiento frente a una gran capacidad de peticiones a la vez y su bajo consumo de recursos de la memoria RAM o de la CPU frente a servidores al realizar las mismas tareas.

Pero en contra, la configuración de NGINX puede ser más compleja, además de que NGINX puede tener más errores de compatibilidad. Es por ello por lo que me decidí a usar Apache en vez NGINX, ya que al hacer la configuración sobre Docker me aseguro de no tener problemas de compatibilidad. Pero sin duda lo que me hizo decantarme por Apache fue porque este es más popular por lo que tiene mucha más documentación e información que me facilita la creación de este sobre Docker.

En conclusión, que optase por apache en vez de otro fue la familiaridad personal, el amplio soporte y documentación, la gran adaptabilidad a diferentes necesidades y su reputación como opción segura y confiable.

4.1.1 Configuración

(Antes de nada, quiero aclarar que las contraseñas utilizadas son contraseñas simples ya que es un entorno de pruebas, en el momento de ponerse en ejecución las contraseñas se deberían de modificar).

Para comenzar la creación de este contenedor lo primero que hay que realizar es su arranque, ya que al ser en HTTPS y en docker, hay que definir un puerto para poder acceder a el hasta tener un DNS configurado:

docker run -it –name apache2 –p 9999:443 debian

Al lanzar este comando arrancaremos una imagen de manera que nos permitirá interactuar directamente con el contenedor y acceder al sitio desde el navegador con **https://localhost:9999.**

Una vez ya este arrancado lo primero que hay que instalar son las dependencias siguiente:

apt install apache2 php libapache2-mod-php php-mysql -y

apt install nano iproute2 iputils-ping openssl openssh-server rsyslog -y

El siguiente paso es crear el sitio web, pero este lo iré configurando a medida que los demás servicios se añadan. Este se debe de crear en /etc/apache2/sites-availables, yo decidí llamarlo **intranetHTTPS.conf.** Luego de esto, he de habilitarlo con **a2ensite intranetHTTPS.conf** y reiniciar el servicio apache para guardar y aplicar los cambios.

4.1.2 HTTPS

HTTPS (*HTTPS*, s. f.) es la versión segura de HTTP, la característica principal de este es que encripta las comunicaciones y hoy en día es el principal protocolo para la transferencia de datos.

Para hacer la configuración de HTTPS en el servidor, hay que activar el módulo SSL en apache con un **a2enmod openssl**.

Una vez activado, se debe crear el certificado con el que se autenticara nuestro sitio.

Ahora sí, para crear este certificado hay que hacer lo siguiente:

- **openssl genrsa –out server.key 2048**
- **openssl req –new –key server.key -out server.csr**
- **openssl x509 –req –days 365 –in server.csr -signkey sever.key -out server.crt**

Primero de todo crearemos una key (llave) de 2048 bits, con la que crearemos un csr (solicitud de firma de certificado a una CA). Estos dos archivos servirán para crear un certificado auto-firmado.

Ahora una vez esto hecho correctamente, hay que editar el sitio web para que se apliquen los cambios de la siguiente manera. Además hay que recordar modificar el puerto por el que este VirtualHost va estar accesible, siendo este 443.

```
<VirtualHost *:443
    SSLEngine on
    SSLCertificateFile      /etc/ssl/certs/server.crt
    SSLCertificateKeyFile   /etc/ssl/certs/server.key
```

Así pues, el servicio https ya estaría disponible y accesible a través del puerto 9999 en nuestro equipo local

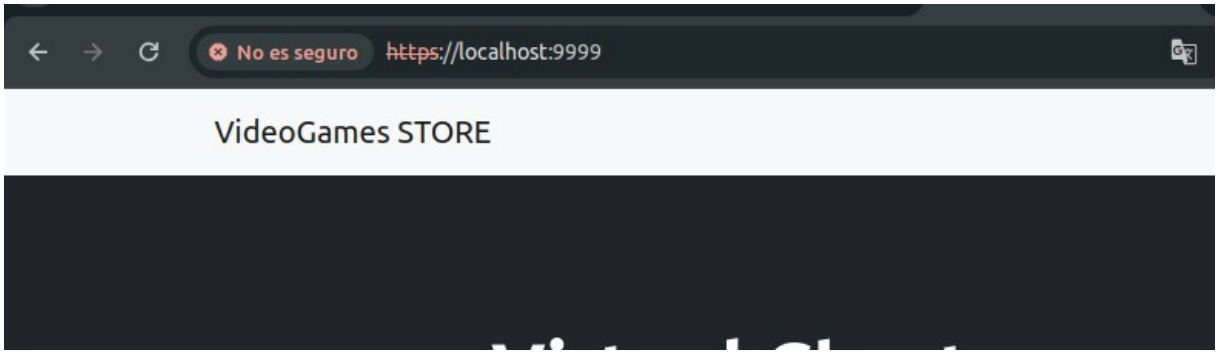


Figura 4: Conexión a la web por el puerto 9999

4.1.3 Autenticación Digest

Esta decisión la tome tome ya que este es un mecanismo simple de autenticación que permite saber si el usuario tiene acceso o no al servidor y así pues quien no tenga acceso no podrá acceder a este.

Esta medida la tomé ya que decidí usar este tipo de “autenticación doble”, ya que al acceder a nuestra web primero comprueba que tengamos acceso al servidor, y luego es la web la que comprueba otra vez si el usuario esta registrado diferenciando por tipo de empleado.

Para poner en uso este rse debe activar el módulo **auth_digest con el comando a2enmod auth_digest** y seguido de esto hay que reiniciar el servicio de apache.

Ahora en mi caso, utilice un archivo .htdigest para almacenar los usuarios que tengan acceso, estos se crearan dentro de este archivo con el siguiente comando **htdigest –c /etc/apache2/.htdigest “UsersDigest” nombre**

En caso de que se desee añadir más usuarios mas tarde de añadir el primero, el comando a realizar es el mismo, pero se debe de quitar el -c ya que esto sirve para crear un nuevo archivo de autenticación nuevos o sobrescribir los archivos ya existentes.

Una vez añadidos todos lo usuario que se deseen, es muy importante darle los permisos correctos para que se pueda utilizar, pero manteniendo la seguridad de los datos.

Los permisos serian **chmod 600** al archivo .htdigest así dejando solamente al propietario del archivo tenga permisos de lectura y escritura . Seguido de esto modificaremos al propietario **chown www-data:www-data /etc/apache2/.htdiges** para cambiar el propietario a apache y pueda acceder a este documento cuando nos autentiquemos.

Por ultimo se deberá modificar el sitio par indicarle que cada vez que quiera acceder alguien le pida esta autenticación, y recordando actualizar el servicio de apache para aplicar los cambios.

```
<Directory /var/www/html/bootstrap>
    AuthType Digest
    AuthName "UsersDigest"
    AuthDigestDomain /var/www/html/bootstrap
    AuthDigestProvider file
    AuthUserFile /etc/apache2/.htdigest
    Require valid-user
</Directory>
```

4.1.4 Módulo Lua

El módulo Lua en Apache tiene diferentes funcionalidades, pero en mi caso lo voy a utilizar para monitorizar el servidor web y ver que el rendimiento y estado de este sea correcto.

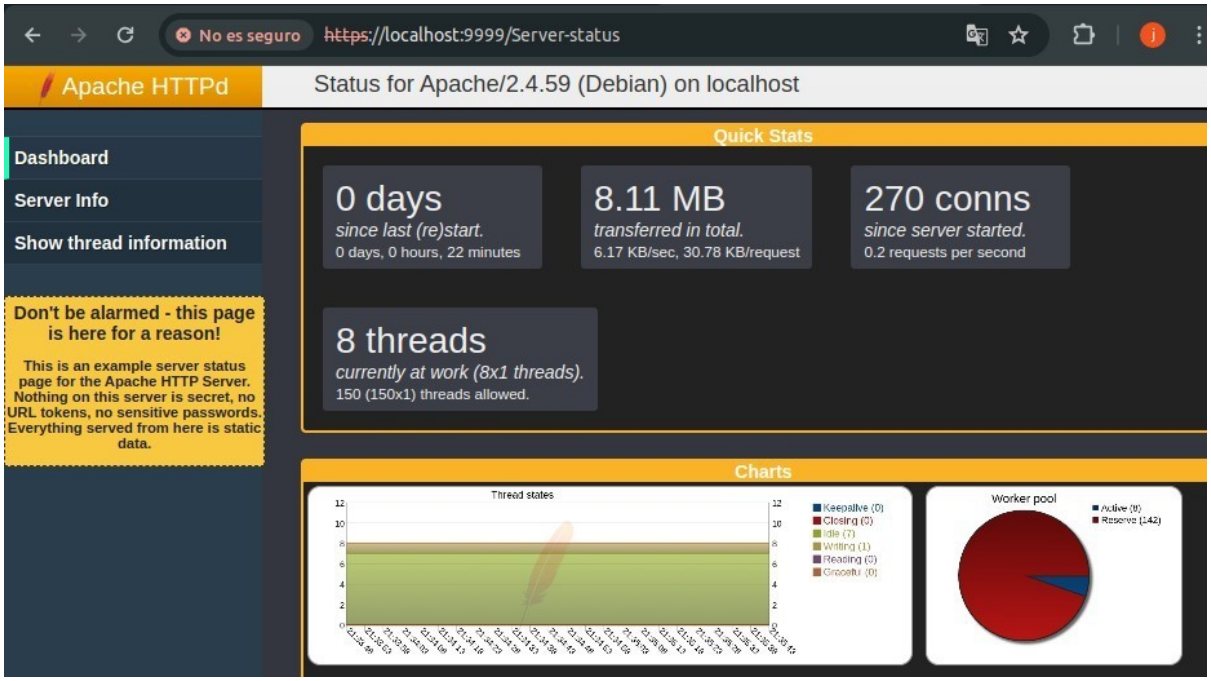
Para la instalación de este seguiremos una serie breve de pasos, empezando por añadirlas siguientes líneas de comando en el archivo de configuración **/etc/apache2/mods-availables/status.conf**

```
<IfModule mod_status.c>
    <Location /Server-staus>
        SetHandler server-status
        Require all granted
    </Location>
</IfModule>
```

Ahora con un archivo llamado **server-status.lua** que se puede encontrar en GitHub se debe copiar el archivo al contenedor con la ayuda de **docker cp**, este archivo ayudara a la hora de gestionar el servidor ya que dota de un estilo a la página y la información se deja de ser solo en formato de texto. Seguido de esto le diremos al sitio que lea este archivo con la siguiente linea.

```
LuaMapHandler /Server-status /etc/apache2/server-status.lua
```

Ahora solo quedaría activar el módulo lua con **a2enmod lua** y reiniciar el servicio apache. Para comprobar ver que funciona correctamente debemos acceder a la web y añadir **/server-status**



4.1.5 Bootstrap

¿Qué es Bootstrap?

Bootstrap es un framework de código abierto para desarrollo web que facilita la creación de sitios responsivos. Al presentar la idea de mi proyecto, se me recomendó usar un sistema como **PrestaShop**, pero mi tutor me sugirió emplear **Bootstrap**, por lo que investigué sus características y decidí utilizarlo, ya que se ajustaba mejor a mis necesidades.

Bootstrap es una biblioteca multiplataforma con herramientas basadas en HTML, CSS y JavaScript, centradas exclusivamente en el *front-end*, es decir, en el diseño y apariencia del sitio desde el punto de vista del usuario.

Opté por Bootstrap frente a PrestaShop porque me ofrece mayor libertad creativa. Mientras que PrestaShop está más enfocado al desarrollo de tiendas online de forma estructurada, Bootstrap permite diseñar sitios personalizados desde cero, adaptados a lo que realmente quiero construir.

¿Cómo lo pongo en uso?

Aplicar una web basada en **Bootstrap** es bastante sencillo. El primer paso fue buscar una plantilla ya diseñada que me resultara atractiva y, una vez elegida, descargar todos los archivos asociados. Tras esto, copié la carpeta al servidor web usando el comando docker cp, colocándola en la ruta /var/www/html bajo el nombre bootstrap, y le asigné permisos 755 de forma recursiva, para que fuese accesible pero solo modificable por el propietario.

Después, modifiqué la configuración del sitio web para que usara esa carpeta como principal, editando la directiva:

DocumentRoot /var/www/html/bootstrap

La plantilla descargada incluía archivos .js, .html y .css, pero además realicé modificaciones en el HTML para incluir un **buscador** que permite hacer consultas a la base de datos, filtrando los productos, poder editarlos, y dependiendo de si eres admin poder añadir nuevos empleados a la web ademas de productos, entre otras opciones con la ayuda de **PHP**.

Para que esto funcionara correctamente, añadí diferentes archivos PHP adicionales para poder realizar todas las operaciones que quería, al final el uso de Bootstrap estaba más enfocado a la parte de css que con un poco de paciencia el css, ya que tuve que crear yo las paginas para poder editar productos, añadir empleados, etc.

En resumen, usar Bootstrap me resultaba más sencillo a la hora de crear mi intranet, ya que lo que yo quería era un sitio para gestionar inventariado de manera sencilla sin tener que preocuparme por el CSS.

4.2 Servidor de Base de datos

Los servidores de base de datos (*Servidor de Base de Datos*, s. f.) proporcionan un servicio de gestión de muchos datos almacenados de manera segura permitiendo hacerle consulta para gestionarlos.

4.2.1 Sistemas de gestión de base de datos

Hay diferentes tipos de SGBD y cada uno se clasifica dependiendo de su distribución. En este caso me planteé usar MySQL o PostgreSQL, ya que cada uno tiene sus ventajas.

MySQL es un sistema de gestión de datos relacional, que relaciona los datos en forma de tablas mediante claves primarias. Este es de código abierto y es famoso por su alta velocidad de ejecución y facilidad de uso además de ser compatible con una gran variedad de lenguajes de programación.

Por otro lado, PostgreSQL también es un sistema de gestión relacional, pero este es más complejo de comprender y, además, la compatibilidad general es inferior a lo que lo podría ser MySQL.

Así que me decante por usar MySQL por diferentes motivos como su facilidad de uso o su alto rendimiento y su alta capacidad de compatibilidad, pues esto es importante al crear la base de datos sobre Docker.

4.2.2 Instalación y configuración

Para la instalación del servidor de base de datos, hay que recordar que se va a ejecutar en un Docker. Esto es importante, ya que no es una instalación normal, sino que tendremos con la ayuda de dockerfile y docker compose para la creación.

Como podemos ver la estructura entidad relación es sencilla ya que tendremos unos proveedores que nos brindaran nuestros productos, los cuales podrán ser gestionados por los usuarios. Estos los separaremos en 2, USUARIOS empleado, que podrán hacer uso de la aplicación gestionando-los, y el admin que podrá añadir nuevos empleados y añadir mas productos entre otras características únicas de su rol

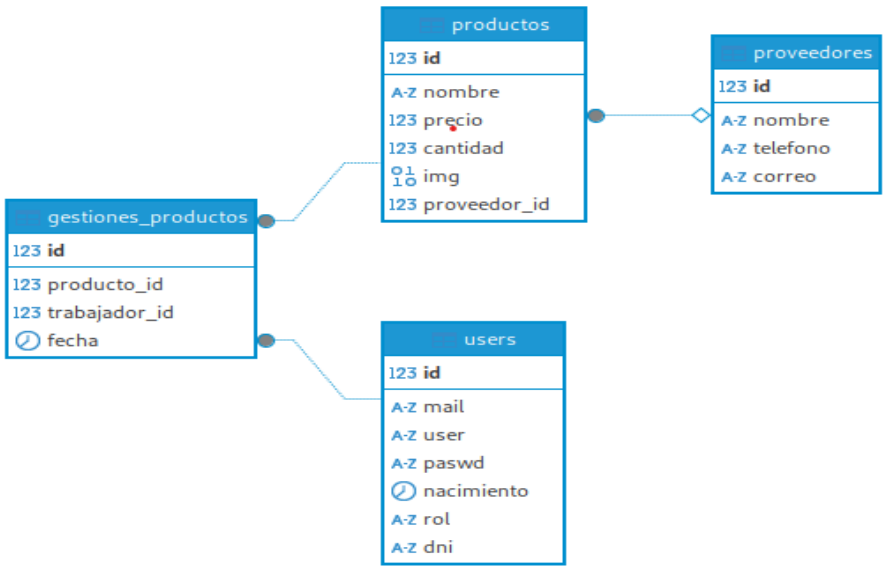


Figura 6: Diagrama entidad relación

Ahora si, para la creación de esta base de datos, lo que necesitaremos es indicar lo necesario en el dockerfile, pero en este a diferencia de como podemos configurar otros, simplemente se debe instalar la imagen de MySQL y aplicar las características necesarias.

```
 mysql:
image: mysql:latest
container_name: database
environment:
 MYSQL_ROOT_PASSWORD: 1234
volumes:
 - mysql_data:/var/lib/mysql
 - ./db:/docker-entrypoint-initdb.d
networks:
 redbridge:
  ipv4_address: 192.168.100.4
ports:
 - "3306:3306"
```

En este bloque vemos el servicio de MySQL montado con la imagen oficial. Le damos un nombre al contenedor (database), le pasamos la contraseña del root por variable de entorno y le asignamos una IP fija dentro de la red redbridge.

En la parte de volúmenes, se le asignan dos rutas:

- La primera (mysql_data:/var/lib/mysql) es un volumen nombrado que sirve para que los datos de la base de datos no se pierdan aunque se reinicie el contenedor. Es por esto que las backups se hacen de este volumen, y se subirán a la nube para que en caso de perder el PC entero poder restaurarlo manualmente.
- La segunda (`./db:/docker-entrypoint-initdb.d`) se utiliza para que al levantar el contenedor, MySQL ejecute automáticamente un script `.sql` que hay en esa carpeta. Ese script contiene la base de datos que vamos a utilizar en el proyecto, por lo tanto es necesario dejar este volumen montado para que se cargue correctamente al iniciar el servicio por primera vez.

Ya por último, exponemos el puerto 3306 para poder conectarse desde fuera del contenedor con una aplicación externa como Dbeaver.

4.3 Servidor DNS

DNS o Domain Name Server, es un sistema que es fundamental en internet ya que sirve para hacer más legible los nombres de dominios, es decir que sea más humano, en vez de ser IP numéricas que las máquinas utilizan para identificarlas.

Por ejemplo, en vez de acceder a 142.250.217.68, lo que buscamos es www.google.com ya que, para nosotros, suele ser más fácil recordar algo así como www.amazon.com, www.netflix.com a una dirección ip diferente para cada sitio web. En conclusión, es un servicio que sirve para dar un nombre a una dirección IP a la cual la máquina se dirige, pero que el ser humano se dirige a esta mediante un nombre para que sea más fácil de recordar.

4.3.1 Instalación y configuración

Para la instalación del DNS sera necesario instalar bind9, junto con las siguientes dependencias para poder continuar con la configuración y comprobar configuraciones:

```
apt install iproute2 dnsutils iputils-ping nano bind9 bind9utils bind9-doc openssh-server -y
```

Para la configuración, solo se necesita modificar 5 archivos, siendo el primero el /etc/default/named indicándole que pueda leer peticiones en ipv4.

Seguido de esto en el archivo named.conf.local especificaré donde se encuentran las zonas que utilizaremos para resolver, estas serán dos, la zona normal, y la zona inversa, una servirá para resolver las direcciones por nombre, y la otra zona para resolver por dirección ip. Una vez todo configurado correctamente hay que recordar actualizar el servicio bind9 o named, y ya estaría listo para su uso.

La configuración del archivo named.conf.local y de la zona principal es la siguiente:

A continuación editaremos las zonas para que resuelvan las direcciones correspondientes.

```
;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     proyecto.com. root.proyecto.com. (
                              3         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.proyecto.com.
@       IN      A       192.168.100.2
ns      IN      A       192.168.100.2
www     IN      A       192.168.100.3
```

Básicamente de esta manera le estamos indicando que la dirección ip del servidor se resuelva por www.proyecto.com esto como ya hemos dicho nos facilitara la conexión al sitio web ya que es más cómodo escribir esto.

Una vez configurado el DNS y que este en funcionamiento correctamente, ya resolverá las direcciones que estén en la red de docker. Es por esto que para que podamos acceder a la intranet desde nuestra maquina host hay que indicar en el **/etc/resolv.conf** la dirección ip de este contenedor DNS, en este caso la **192.168.100.2**. Esto lo podemos hacer con:

Y, así pues, así quedaría la intranet con el DNS configurado, dejando buscar por nombre en nuestro equipo local.

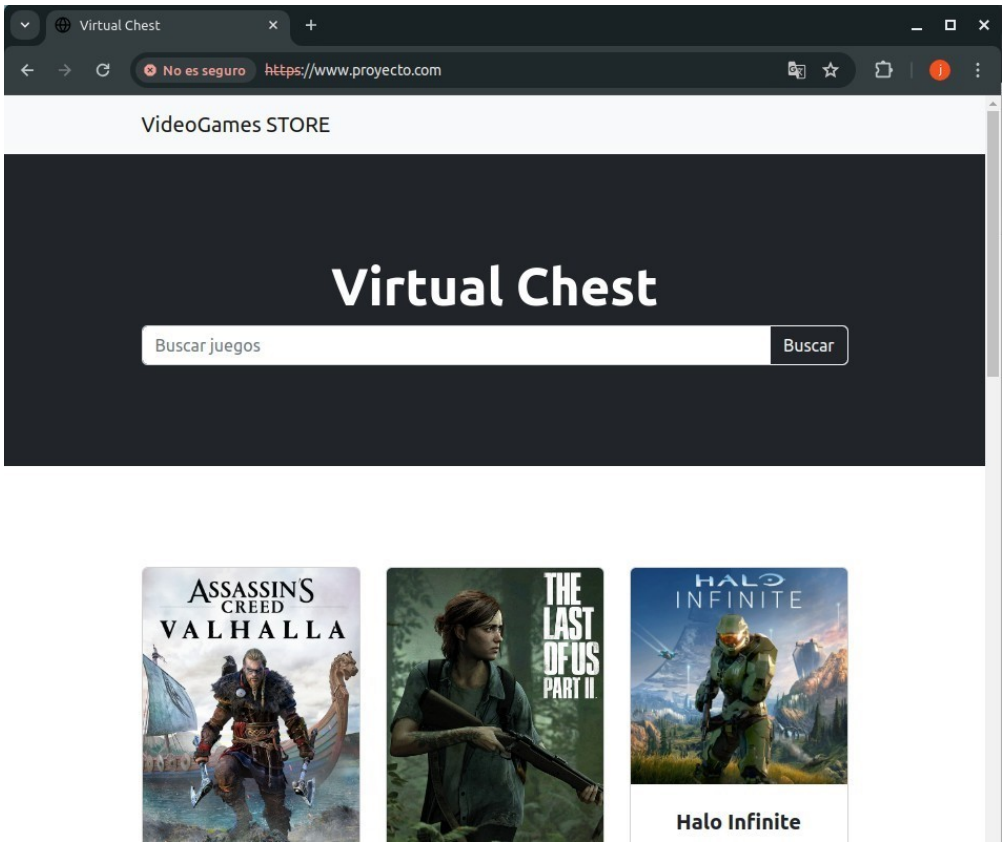


Figura 7: Intranet con resolución por nombre

4.4 Servidor SSH (Manager)

Este servicio tiene como objetivo ser el único punto de entrada seguro a la infraestructura. Permite una vez nos conectemos a el por ssh y pasando la barrera del fail2ban poder conectarnos a los contenedores restantes permitiendo un mantenimiento asegurando la gestión remota de forma controlada y protegida.

¿Como lo configuro?

Este contenedor a diferencia de los otros lo configuro directamente en el dockerfile ya que es más rápido

```
FROM debian:latest

RUN apt-get update && apt-get install -y openssh-server iputils-ping default-mysql-client
dnsutils net-tools fail2ban rsyslog && rm -rf /var/lib/apt/lists/*

RUN mkdir -p /var/run/sshd
RUN echo 'root:1234' | chpasswd

RUN echo "PermitRootLogin yes" > /etc/ssh/sshd_config.d/custom-login.conf && \
  echo "PasswordAuthentication yes" >> /etc/ssh/sshd_config.d/custom-login.conf
COPY jail.local /etc/fail2ban/jail.local
EXPOSE 22
RUN touch /var/log/auth.log && chmod 640 /var/log/auth.log
CMD ["sh", "-c", "/usr/sbin/rsyslogd -n & /usr/sbin/sshd -D & /usr/bin/fail2ban-server -b &
tail -F /var/log/auth.log"]
```

Este dockerfile construye un contenedor de manera muy sencilla que proporcionara un **punto de acceso SSH** seguro a la infraestructura. Como vemos, durante la creación se instalan diferentes dependencias como el openssh-server, para poder conectarnos, así como el fail2ban y el rsyslog para ver los logs y la protección contra ataques de fuerza bruta. Además añade un **cliente MySQL** para facilitar la interacción con la BD.

Además podemos ver un parte de un bloque donde se configura el acceso SSH, permitiendo el login de root con contraseña y nos aseguramos de que los logs se generen y monitoricen. Finalmente, el contenedor **inicia simultáneamente todos los servicios de log, SSH y Fail2ban** nada más arrancar

5 Scripts

Los scripts son una herramienta fundamental en este proyecto, ya que al final son como el motor que permite automatizar todo. Estos son archivos que contienen secuencias de instrucciones que se ejecutan de manera ordenada para realizar diferentes tareas.

Para esta infraestructura podemos diferenciar 2 diseños, uno para automatizar el despliegue completo y otro para gestionar las copias de seguridad.

La finalidad de estos scripts es transformar el proceso de configuración manual, que podría ser bastante largo y con errores, en una ejecución rápida y consistente permitiendo así crearlo todo en cuestión de segundos o pocos minutos.

5.1 Despliegue y configuración para los backups

Este es el script principal que organiza el despliegue de toda la infraestructura asegurando la configuración para el correcto funcionamiento del sistema, incluyendo la preparación para poder ejecutar sin problemas el siguiente script que nos permitirá hacer backups y subirlos a la nube

```
#!/bin/bash

if ! docker network ls | grep -q "redbridge"; then
 echo "La red 'redbridge' no existe. Creándola..."
 sudo docker network create --driver=bridge --subnet=192.168.100.0/24 redbrid-
ge
else
 echo "La red 'redbridge' ya existe."
fi
sudo docker compose up -d
echo "Configurando systemd-resolved para usar el contenedor DNS..."
if ! grep -q "DNS=192.168.100.2" /etc/systemd/resolved.conf; then
```

```
  sudo bash -c 'echo -e "[Resolve]\nDNS=192.168.100.2\nFallbackDNS=8.8.8.8
1.1.1.1" > /etc/systemd/resolved.conf'
 sudo systemctl restart systemd-resolved
fi
echo "Configuración completada. Comprueba el resolv.conf"
if ! command -v rclone &> /dev/null; then
 echo "rclone no está instalado. Instalando..."
 curl https://rclone.org/install.sh | sudo bash
else
 echo "rclone ya está instalado."
fi

if [ ! -d ~/.config/rclone ]; then
 mkdir -p ~/.config/rclone
 echo "He creado la carpeta ~/.config/rclone"
else
 echo "La carpeta ~/.config/rclone ya existe"
fi
if cp -f rclone.conf ~/.config/rclone/rclone.conf 2>/dev/null; then
 echo "He copiado rclone.conf a ~/.config/rclone/"
else
 echo "No he encontrado rclone.conf aquí"
fi
sudo mkdir -p /opt/scripts
sudo cp "backups.sh" /opt/scripts/backups.sh
LINEA="0 2 * * * /opt/scripts/backups.sh"
crontab -l 2>/dev/null | grep -q "backups.sh"
if [ $? != 0 ]; then
 crontab -l 2>/dev/null > mycron
 echo "$LINEA" >> mycron
 crontab mycron
 rm mycron
 echo "Tasca afegida al crontab per a cada dia a les 02:00h."
else
 echo "La tasca ja estava al crontab."
fi
```

El script comienza creando la red de docker para que los contenedores se comuniquen entre ellos. Seguido de esto, lanza el docker compose y modifica el **system-resolved** de la maquina host para indicar que el DNS principar es el que nosotros hemos creado, asi por nombre desde nuestra maquina podremos acceder y a la vez al tener otros DNS de Google como respaldo podremos usar cualquier función que requiera de internet.

A continuación, instala rclone y crea su carpeta de configuración añadiendo un archivo **.conf** ya configurado anteriormente con las credenciales necesarias para sincronizar

con servicios en la nube como **Dropbox**. Ya para finalizar añadimos el script que realiza las backups en una carpeta externa y con la ayuda de crontab del host indicamos que todos los dias a las 2:00 AM se ejecute este script, guardando una copia de seguridad en la nube.

5.2 Creación de Backups y copias en la nube

Este script realiza una copia de seguridad del volumen de datos de MySQL gestionado por Docker. Primero define el nombre de la empresa, que se usará como identificador en la ruta de destino en la nube. Luego **localiza la ruta del volumen mysql_data** dentro del sistema mediante docker volume inspect, y genera un nombre único para el archivo de backup en función de la fecha y la hora actuales.

Crea (si no existe) el directorio **/opt/backups**, donde se almacenará temporalmente el archivo comprimido. A continuación, comprime todo el contenido del volumen en un archivo .zip utilizando zip.

Finalmente, utiliza rclone para subir el archivo comprimido a una carpeta específica dentro del almacenamiento en la nube de Dropbox, permitiendo así mantener una copia externa segura y accesible de los datos de la base de datos.

```
#!/bin/bash

EMAIL_EMPRESA="proyecto_videogames"
VOLUMEN="mysql_data"
RUTA_VOLUM=$(sudo docker volume inspect "$VOLUMEN" -f '{{ .Mountpoint }}')
FECHA=$(date +%F-%H-%M)
NOMBRE_ZIP="backup-$FECHA.zip"
RUTA_LOCAL="/opt/backups"
sudo mkdir -p "$RUTA_LOCAL"
RUTA_ZIP="$RUTA_LOCAL/$NOMBRE_ZIP"
CARPETA_DROPBOX="/backups/$EMAIL_EMPRESA"
sudo zip -r "$RUTA_ZIP" "$RUTA_VOLUM"
rclone copy "$RUTA_ZIP" "dropbox:$CARPETA_DROPBOX"
```

6 Conclusión

En resumen, este proyecto ha consistido en crear un script que ejecuta una serie de órdenes en un orden concreto, y que, con ayuda de Docker Compose y Dockerfile, genera varias imágenes y arranca cada servicio en su propio contenedor con una configuración específica. Gracias a esto, se ha podido montar una pequeña intranet para gestionar el inventario de una tienda de videojuegos.

El proceso ha sido bastante complejo, sobre todo al principio, porque no encontré nada similar que me sirviera de guía. Tenía los conocimientos necesarios para hacer cada parte por separado, pero no sabía cómo unirlas en un solo proyecto. Me vi muchas veces sin saber por dónde tirar, y aunque ahora me parezca más fácil, la verdad es que he tenido que hacer mil pruebas, cometiendo muchos errores hasta dar con la manera que funciona.

Aun así, todo esto me ha ayudado mucho a aprender. He entendido mejor cómo funciona Docker Compose y he mejorado bastante mis conocimientos previos. Hasta ahora, lo que sabía de Docker era más teórico o para prácticas simples, pero este proyecto me ha hecho enfrentarme a problemas reales, buscando soluciones prácticas y aprendiendo a desenvolverme mejor en este tipo de situaciones técnicas.

Para entender bien cómo funciona todo, cuando se ejecuta el script principal, primero se construyen las imágenes siguiendo las instrucciones de los Dockerfile. Después, con Docker Compose, se levantan los contenedores, cada uno configurado para su función. Así se pone en marcha la intranet con todos sus servicios trabajando juntos para gestionar el inventario de manera eficiente.

Además, el sistema hace copias de seguridad automáticas para proteger los datos, y el acceso seguro a los contenedores se controla desde el contenedor “manager” mediante SSH, lo que facilita la gestión y el mantenimiento remoto del servicio.

7 Problemas

Uno de los mayores problemas lo tuve con la **base de datos.** Quería usar un archivo .**sql** con todos los datos ya preparados y que se aplicara directamente al levantar el contenedor con Docker Compose, pero no fue nada fácil. Estuve probando varias formas hasta que conseguí que funcionara como yo quería. La idea era que al levantar todo el sistema, la base de datos ya estuviera montada y lista, sin tener que hacer nada manual, y me costó bastante dejarlo bien configurado.

También tuve que tomarme bastante tiempo para los **scripts**. Empecé haciéndolos de forma sencilla, pero poco a poco fueron cogiendo más complejidad. Al final han acabado siendo bastante más avanzados de lo que esperaba, y hacer que todo encajara y funcionara sin errores ha sido todo un reto. No ha sido nada fácil, pero ha valido la pena porque he aprendido mucho.

Otro punto complicado fue **Docker Compose**, ya que era la primera vez que lo usaba. Al principio no entendía bien cómo funcionaba, pero eso me sirvió para profundizar más en Docker. Antes tenía una idea más básica, como si fuese una máquina virtual, pero ahora ya tengo más claro lo que es un contenedor, cómo se comunican entre ellos y cómo se gestionan.

8 Futuras mejora

Una mejora importante sería hacer que el cuarto contenedor, que está pensado para la VPN, funcione correctamente para poder acceder al sistema desde cualquier lugar con total seguridad. Ahora mismo la VPN está configurada de forma básica, pero mejorarla para que sea más estable, fácil de usar y compatible con distintos dispositivos sería un gran avance. Así podría ofrecer mantenimiento remoto sin importar dónde esté el técnico o el cliente.

También se podría implementar un panel de control web sencillo para monitorizar el estado de los diferentes servicios (Apache, base de datos, DNS y VPN) y poder reiniciarlos o configurar-los desde ahí, sin necesidad de acceder por consola. Esto facilitaría mucho el uso diario y el mantenimiento.

Otra mejora interesante sería automatizar las actualizaciones de los contenedores para que siempre usen las últimas versiones seguras sin necesidad de intervención manual, asegurando que el sistema esté siempre protegido y actualizado.

Por último, se podría trabajar en mejorar el sistema de copias de seguridad, con notificaciones automáticas por correo o mensajes cuando las copias se realicen correctamente o fallen, para tener un control más preciso y poder reaccionar rápido ante cualquier problema.

9 Mi repositorio GitHub

En mi repositorio GitHub podremos encontrar todos los archivos de configuración. Aquí podremos encontrar seis archivos, de los cuales uno sera el yaml del docker compose, y los otros cuatro son carpetas en las que cada una tiene dentro todo lo necesario para ejecutar su servicio, así pues las carpetas tienen el nombre de su respectivo servicio ya sea el de apache o la carpeta con los scripts. Por ultimo el archivo que quedara es el pdf de esta documentación

El enlace a mi repositorio es el siguiente:

https://github.com/ieslavereda-projects/23_24_ASIR_JOSE_GAVILAN_AUTOMATIZACION_ENTORNO

Los contenidos del repositorio se verían algo así:

apache/

databas

e/ dns/

scripts/

docker-compose.yaml

Proyecto-Jose-Gavilan.pdf

10 Índice de imágenes

Índice de figuras

11 Bibliografía

Automatización. (s. f.). [Software]. https://www.rae.es/dpd/automatizaci%C3%B3n

Bootstrap. (s. f.). [Software].

https://www.inboundcycle.com/blog-de-inbound-marketing/que-es-bootstrap

Contenedores Docker. (s. f.). [Software].

https://www.netapp.com/es/devops-solutions/what-are-containers/

Diagrama de gantt. (s. f.). [Software].

https://www.atlassian.com/es/agile/project-management/gantt-chart

DNS. (s. f.). [Software]. https://www.cloudflare.com/es-es/learning/dns/what-is-dns/

Docker compose. (s. f.). [Software]. https://www.dongee.com/tutoriales/que-es-docker-compose/

Docker (v2.9). (s. f.). [Linux]. https://www.redhat.com/es/topics/containers/what-is-docker

Dockerfile. (s. f.). [Software]. https://www.ionos.es/digitalguide/servidores/know-how/dockerfile/

HTTPS. (s. f.). [Software]. https://www.cloudflare.com/es-es/learning/ssl/what-is-https/

Máquina virtual. (s. f.). [Software]. https://www.redhat.com/es/topics/virtualization/what-is-a-virtual-machine

Script. (s. f.). [Software]. https://www.inboundcycle.com/diccionario-marketing-online/script

Servidor de Base de Datos. (s. f.). [Software]. https://www.ovhcloud.com/es-es/learn/what-is-database-server/

Servidor Web. (s. f.). [Software]. https://rockcontent.com/es/blog/que-es-un-servidor/

SGBD. (s. f.). [Software]. https://www.hostinger.es/tutoriales/sgbd

Volumenes Docker. (s. f.). [Software]. https://kinsta.com/es/blog/volumenes-docker-compose/

Crontab. (s. f.). [Software]. https://www.redhat.com/sysadmin/how-use-cron-linux

SSH. (s. f.). [Software]. https://www.ssh.com/ssh/

Rclone. (s. f.). [Software]. https://rclone.org/