



DOCKER

PRÁCTICAS INICIALES

Grupo 8

Estudiantes:

- Aída Alejandra Mansilla Orantes	202100239
- Alvaro Norberto García Meza	202109567
- Benjamin Alexander Torcelli Barrios	201901803
- Kevin Ernesto García Hernández	202113553
- Luis Daniel Salán Letona	202000549

Tutores:

- Walter Guerra	201709073
- Jeffry Méndez	201901557



Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de EPS
Prácticas Iniciales



Índice

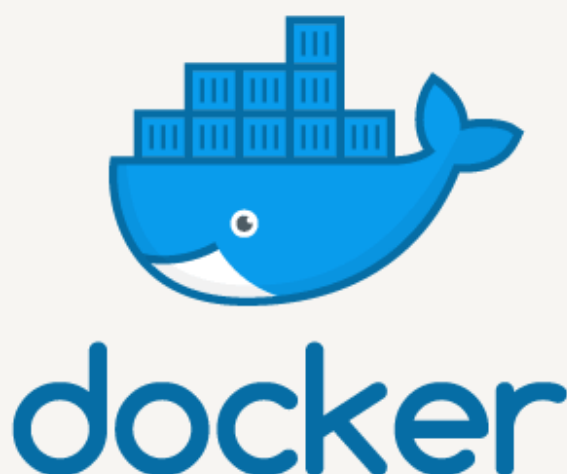
- 01** ¿Qué es Docker y cómo funciona?
- 02 - 04** ¿Cómo crear un DockerFile?
- 05 - 07** ¿Cómo obtener imágenes de DockerHub?
- 08** ¿Cómo subir imágenes de DockerHub?
- 09 - 11** ¿Cómo mostrar imágenes que estén corriendo actualmente y consumirlas?
- 11 - 12** Aplicación Cliente: Ver la aplicación cliente en el puerto alojado en Docker
- 12** Aplicación Servidor: Hacer un GET en el puerto alojado en Docker
- 13** ¿Cómo crear un servicio con Docker-compose?

¿Qué es Docker y cómo funciona?

Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta un software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, como lo que son bibliotecas, herramientas de sistema, código, tiempo de ejecución, entre otras.

Con Docker, puedes implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de que tu código se ejecutará. proporciona una manera estándar de ejecutar su código ya que es un sistema operativo para contenedores, que funciona De manera similar a una máquina virtual, es decir que elimina la necesidad de administrar directamente el hardware del servidor y los contenedores virtualizan el sistema operativo de un servidor.

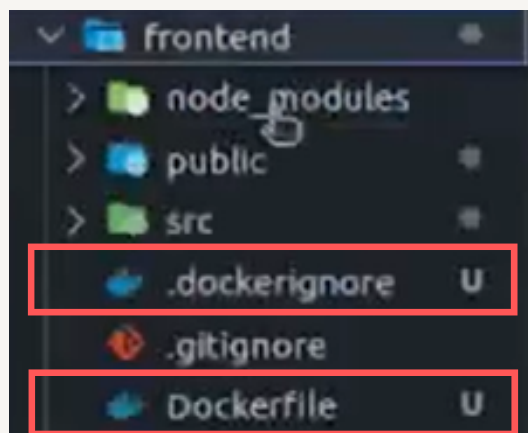
Docker se instala en cada servidor y proporciona comandos sencillos que puede utilizar para crear, iniciar o detener contenedores, también permite entregar código con mayor rapidez, estandarizar las operaciones de las aplicaciones, transferir el código con facilidad y ahorrar dinero al mejorar el uso de recursos. La sintaxis es sencilla y simple y contiene una amplia adopción, lo que significa que existe un gran ecosistema de herramientas y aplicaciones listas para su uso.



¿Cómo crear un DockerFile?

Paso No. 1

Una vez creado el backend y el frontend, debemos tener una carpeta lista que contenga los archivos con la construcción de ambos. Para obtener los docker-Files, abrimos la carpeta en Visual Studio Code y en el archivo del frontend creamos dos archivos nuevos, un .dockerignore y un Dockerfile.



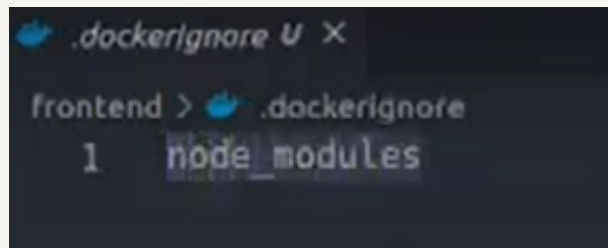
Paso No. 2

Dentro del DockerFile se escribe la version del nodo, se crea el espacio de trabajo, se copiaran los archivos con dependencias y librerías necesarias para realizar los servicios, el ejecutador para crear todas las dependencias, se copia todo el código y se delimita el puerto en el que se va a correr. Por ultimo, se levanta la aplicacion y todo esto respetando la estructura de la siguiente imagen:

```
Dockerfile U X
frontend > Dockerfile
1  # Version node
2  FROM node:18
3
4  # Creando el espacio de trabajo
5  WORKDIR /app
6
7
8  # Copiando archivos con dependencias y librerías necesarias para el servicio
9  COPY package.json ./
10 COPY package-lock.json ./
11
12
13 # Ejecutando para crear dependencias
14 RUN npm install
15
16 # Copiando todo el código
17 COPY . .
18
19 EXPOSE 3000
20
21 # Levantando la aplicación
22 CMD ["npm", "start"]
```

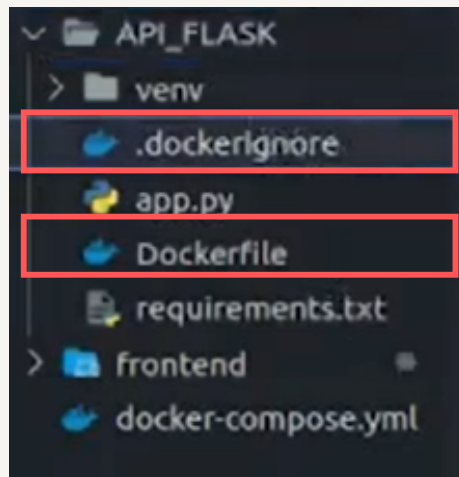
Paso No. 3

En el archivo docker ignore, unicamente se llama al node_modules



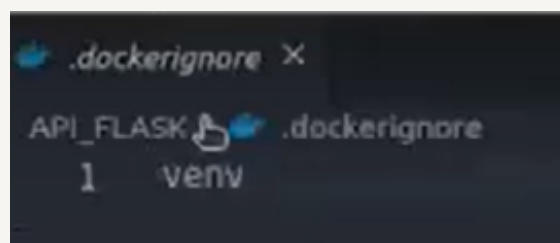
Paso No. 4

Se abre la carpeta que contiene nuestra API y nuevamente se crean los archivos .dockerignore y el DockerFile.



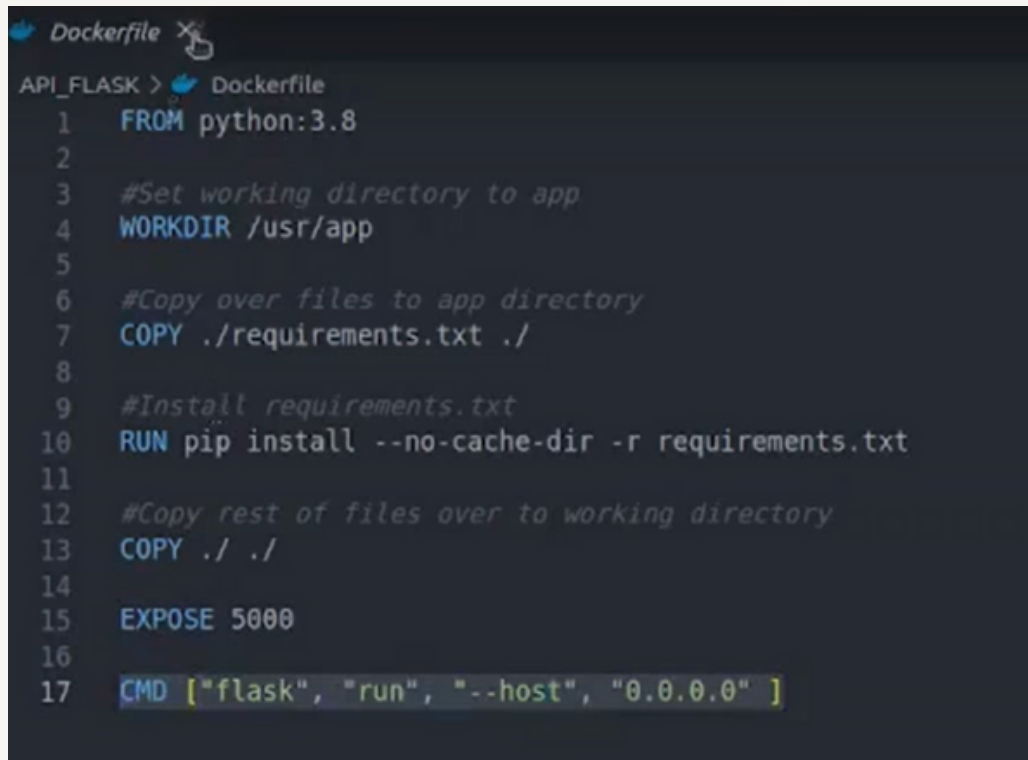
Paso No. 5

En el archivo .dockerignore unicamente llamamos a la carpeta venv de la siguiente forma.



Paso No. 6

En el Dockerfile, de igual manera que en el del frontend, agregamos la version del nodo, se crea el espacio de trabajo, se copiaran los archivos con dependencias, el ejecutador para crear todas las dependencias, se copia todo el código, se delimita el puerto en el que se va a correr y se levanta la aplicación respetando la estructura de la siguiente imagen:



```
Dockerfile
API_FLASK > Dockerfile
1 FROM python:3.8
2
3 #Set working directory to app
4 WORKDIR /usr/app
5
6 #Copy over files to app directory
7 COPY ./requirements.txt ./
8
9 #Install requirements.txt
10 RUN pip install --no-cache-dir -r requirements.txt
11
12 #Copy rest of files over to working directory
13 COPY ./ ./
14
15 EXPOSE 5000
16
17 CMD ["flask", "run", "--host", "0.0.0.0"]
```

¿Cómo obtener imagenes de DockerHub?

Paso No. 1

Para empezar a usar Docker hay que instalarlo antes, por lo cual se usará el comando "sudo apt install docker.io" para hacerlo. Este junto con el comando "sudo docker snap install docker".

```
alvaro@alvaro-System-Product-Name:~$ sudo apt install docker.io
```

```
alvaro@alvaro-System-Product-Name:~$ sudo docker snap install docker
```

Para ver la versión del Docker que estamos manejando se usará el comando "docker --version".

```
alvaro@alvaro-System-Product-Name:~$ docker --version
Docker version 20.10.12, build 20.10.12-0ubuntu4
```

Paso No. 2

En el proyecto que manejamos también tuvimos que usar mysql, por lo cual desde la página de DockerHub pudimos obtener el comando para pullear su imagen, este siendo "sudo docker mysql".

```
alvaro@alvaro-System-Product-Name:~$ sudo docker pull mysql
```

Y para crear un contenedor de la imagen mysql se utilizará el comando "sudo docker run -d -p puerto --name nombre -e MYSQL_ROOT_PASSWORD=contraseñamysql"

```
alvaro@alvaro-System-Product-Name:~$ sudo docker run -d -p 3306:3306 --name database -e MYSQL_ROOT_PASSWORD=secret mysql
```


Una vez creado el contenedor mysql, para iniciarlo deberemos utilizar únicamente el comando "sudo docker start nombre".

```
alvaro@alvaro-System-Product-Name:~$ sudo docker start database;
database
```

Y por último, para ingresar a la base de datos y usarla de una forma normal, utilizaremos el comando "sudo docker exec -it nombre mysql -p"; donde nos pedirá la contraseña que habíamos dado anteriormente.

```
alvaro@alvaro-System-Product-Name:~$ sudo docker exec -it database mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 30
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> █
```

Paso No. 2

Una vez ubicado el dockerfile donde deseemos, para poder construir nuestra imagen utilizaremos el comando "sudo docker build -t nombre/carpeta .". Es muy importante que no se olvide el punto final, para poder construir nuestra imagen correctamente.

```
alvaro@alvaro-System-Product-Name:~/Escritorio/ACTIVIDAD_DOCKER$ sudo docker build -t alvarousac/frontend .
```

Paso No. 3

Por último, para poder corroborar que nuestras imágenes fueron creadas correctamente utilizaremos el comando "sudo docker images".

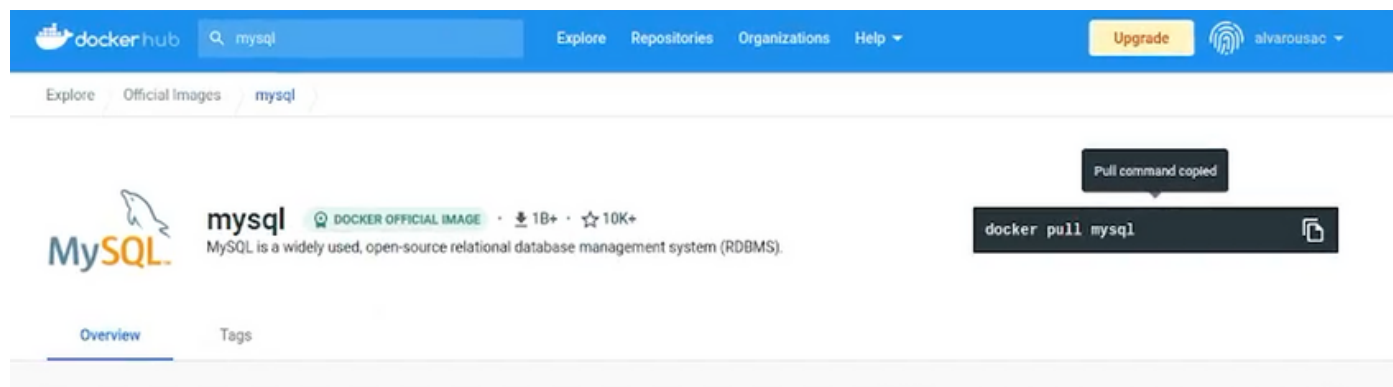
```
alvaro@alvaro-System-Product-Name:~$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alvarousac/frontend	latest	4a78edd7c140	54 minutes ago	1.27GB
alvarousac/backend	latest	52746f6ba4dc	About an hour ago	940MB
mysql	latest	43fcfca0776d	6 days ago	449MB
python	3.8	9fa000b720b7	8 days ago	913MB
node	18	2577ab2cda97	8 days ago	991MB
alpine	latest	9c6f07244728	6 weeks ago	5.54MB

¿Cómo subir imágenes a DockerHub ?

Paso No. 1

Antes que nada, deberemos tener una cuenta creada en DockerHub para poder seguir realizando los pasos de este manual, esta se puede crear fácilmente ingresando a la página oficial.



Paso No. 2

Una vez tengamos una cuenta creada, iniciaremos sesión con el comando "sudo docker login", donde nos pedirá un usuario y contraseña.

```
alvaro@alvaro-System-Product-Name:~$ sudo docker login
```

Paso No. 3

Una vez tengamos iniciada la sesión en nuestra computadora, utilizaremos el comando "sudo docker push alvarousac/backend".

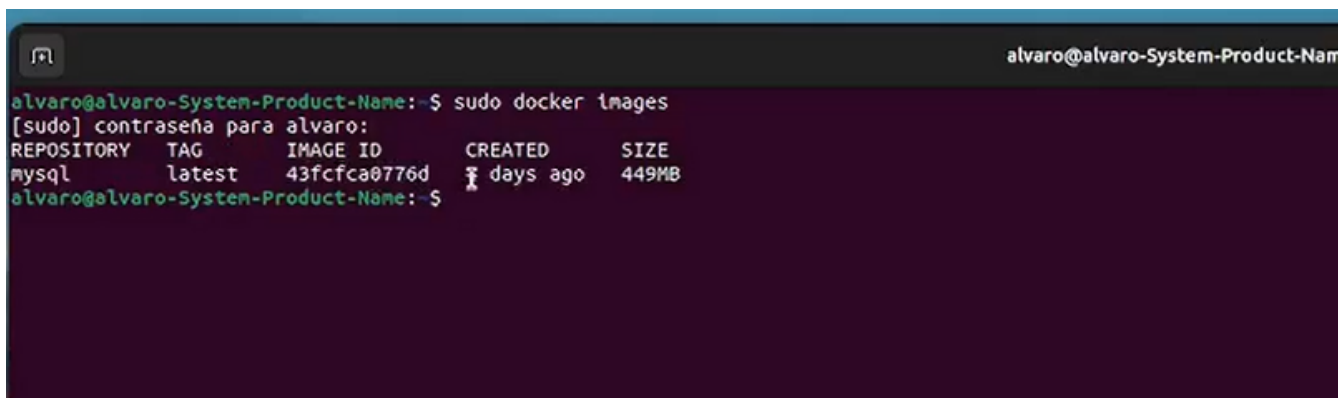
```
alvaro@alvaro-System-Product-Name:~$ sudo docker push alvarousac/backend
```

¿Cómo correr las imágenes de Dockerhub?

¿Cómo mostrar imágenes que estén corriendo actualmente y consumirlas?

Paso No. 1

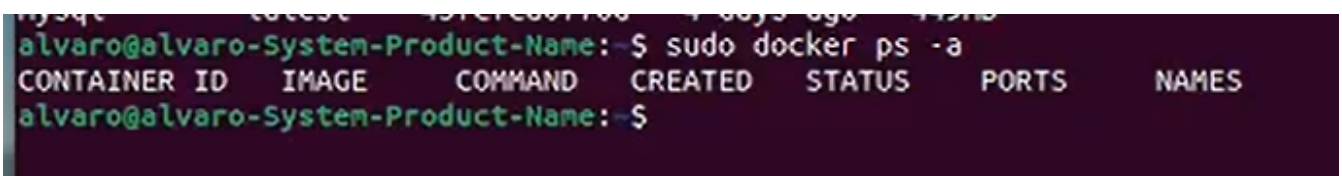
Se escribe el comando "sudo docker images" en una nueva terminal de Ubuntu y generalmente al presionar enter pide la contraseña, así que se ingresa. Posteriormente se despliega el listado de las imágenes que se encuentran en el sistema y todos los atributos como el ID, la fecha de creación, etc.



```
alvaro@alvaro-System-Product-Name:~$ sudo docker images
[sudo] contraseña para alvaro:
REPOSITORY    TAG       IMAGE ID      CREATED       SIZE
mysql         latest    43fcfca0776d  7 days ago   449MB
alvaro@alvaro-System-Product-Name:~$
```

Paso No. 2

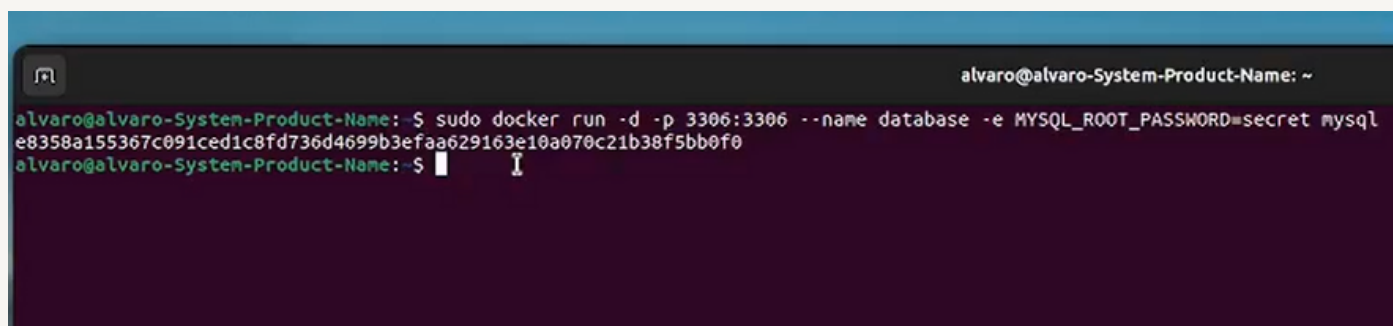
Para ver si existen imágenes que se encuentran corriendo en los contenedores de Docker, se escribe el comando "sudo docker ps -a".



```
alvaro@alvaro-System-Product-Name:~$ sudo docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
alvaro@alvaro-System-Product-Name:~$
```

Paso No. 3

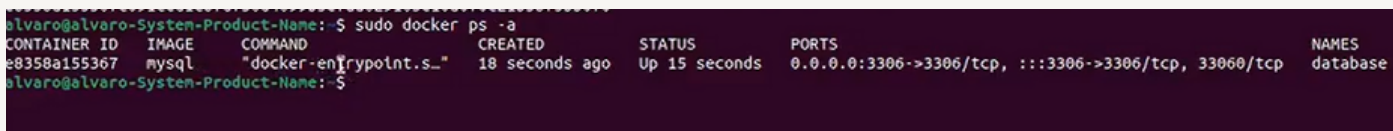
Para ejecutar una imagen y consumir el contenedor de Docker, se abre una nueva terminal en Ubuntu y se escribe el comando "sudo docker run -d -p" acompañado del puerto por defecto de mysql, el nombre y una contraseña. De este modo el comando completo quedaría de la forma "sudo docker run -d -p 3306:3306 --name database -e MYSQL_ROOT_PASSWORD=secret mysql" y se creará un nuevo contenedor de mysql.



```
alvaro@alvaro-System-Product-Name: ~
alvaro@alvaro-System-Product-Name: $ sudo docker run -d -p 3306:3306 --name database -e MYSQL_ROOT_PASSWORD=secret mysql
e8358a155367c091ced1c8fd736d4699b3efaa629163e10a070c21b38f5bb0f0
alvaro@alvaro-System-Product-Name: $
```

Paso No. 4

El resultado será el ID del contenedor y enseguida escribiremos el comando "sudo docker ps -a" para ver si se ha cargado la imagen correctamente.



```
alvaro@alvaro-System-Product-Name: $ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
e8358a155367   mysql    "docker-entrypoint.s..." 18 seconds ago Up 15 seconds 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp database
alvaro@alvaro-System-Product-Name: $
```

Paso No. 5

Para ejecutar el contenedor se escribe el comando "sudo docker exec -it" acompañado del nombre mas "-p", es decir que el comando final queda de la forma: "sudo docker exec -it database mysql -p". posteriormente se ingresa la contraseña del usuario de Ubuntu y la de la imagen de mysql. Una vez realizado esto, se puede observar que la imagen ya se encuentra corriendo.

```
alvaro@alvaro-System-Product-Name:~$ sudo docker exec -it database mysql -p
[sudo] contraseña para alvaro:
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> █
```

Aplicación Cliente: Ver la aplicación cliente en el puerto alojado en Docker

Paso No. 1

escribir el comando "sudo docker ps" para verificar los contenedores que se encuentran corriendo actualmente y se mostrará el ID del contenedor y el puerto en el que está alojado, en este caso, el 3,000.

```
alvaro@alvaro-System-Product-Name:~$ sudo docker ps
[sudo] contraseña para alvaro:
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                                     NAMES
d3e80af6a69c   alvarousac/backend   "flask run --host 0.0.0.0"  18 minutes ago Up 18 minutes 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp  backend
34ef56847019   alvarousac/frontend  "docker-entrypoint.s..."  18 minutes ago Up 18 minutes 0.0.0.0:3000->3000/tcp, :::3000->3000/tcp  frontend
577424f7c5db   mysql                "docker-entrypoint.s..."  2 hours ago   Up 2 hours    0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp  database
alvaro@alvaro-System-Product-Name:~$
```


Paso No. 2

Dirigirse al navegador y escribir "Localhost/3000" y esta ruta redireccionara a la aplicacion cliente.

The screenshot shows a web application with two main sections. On the left, there is a form titled 'Agregar nuevo usuario' (Add new user) with input fields for 'Id:', 'Nombre:' (Name), 'Edad:' (Age), 'Genero:' (Gender), and 'Telefono:' (Phone), followed by a 'Guardar' (Save) button. On the right, there is a table titled 'Lista de Usuarios:' (List of Users) with columns for 'Id', 'Nombre', 'Edad', 'Genero', and 'Telefono'. The table contains three rows of user data. Each row has two buttons: 'Editar' (Edit) and 'Eliminar' (Delete).

Id	Nombre	Edad	Genero	Telefono
1	Ale	9	Binario	12323
2	Kevin	19	Masculino	1231233
3	Alvaro	19	Fluido	12323

Aplicación Servidor: Hacer un GET en el puerto alojado en Docker

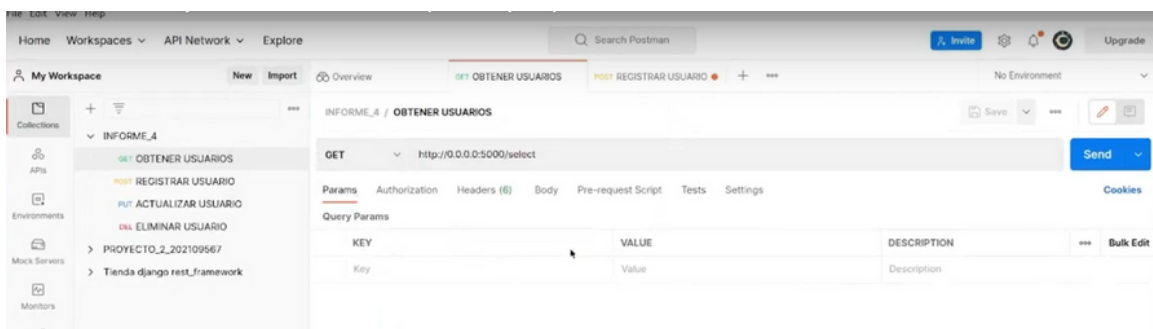
Paso No. 1

Escribir el comando "sudo docker ps" para verificar los contenedores que se encuentran corriendo actualmente y se mostrará el ID del contenedor de la API y el puerto en el que está alojado, en este caso, el 5,000.

```
alvaro@alvaro-System-Product-Name: $ sudo docker ps
[sudo] contraseña para alvaro:
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS                                                                 NAMES
d3e80af6a69c   alvarousac/backend   "flask run --host 0.0.0.0"   18 minutes ago   Up 18 minutes   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   backend
34ef56847019   alvarousac/frontend   "docker-entrypoint.s..."   18 minutes ago   Up 18 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   frontend
577424f7c5db   mysql                "docker-entrypoint.s..."   2 hours ago     Up 2 hours     0.0.0.0:3306->3306/tcp, :::3306->3306/tcp   database
```

Paso No. 2

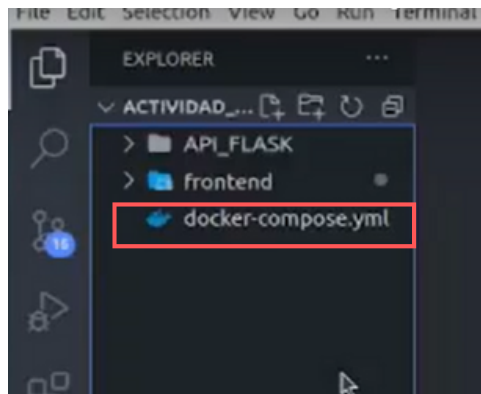
Dirigirse a Postman (programa de peticiones http) para realizar las peticiones correspondientes a la API. Posteriormente, se realiza una petición con el método GET a la URL "http://0.0.0.0:5000/select" para obtener los valores de la base de datos conectada a nuestro servidor.



¿Cómo crear un servicio con Docker-compose?

Paso No. 1

Crear una carpeta y agregar en ella los dos archivos que contienen nuestras aplicaciones, es decir el frontend y el backend y abrir dicha carpeta en Visual Studio Code. Posteriormente, se crea un archivo con el nombre "docker-compose" y con extension .yaml



Paso No. 2

Dentro del archivo se escribe la siguiente estructura que permite crear el servicio del backend y el servicio para el frontend, los cuales se encontraran alojados en distintos puertos y haciendo referencia a sus respectivas imágenes.

```
docker-compose.yml
1  version: "3"
2
3  services:
4    backend:
5      container_name: backend
6      restart: always
7      build: ./API_FLASK
8      image: alvarousac/backend
9      ports:
10     - "5000:5000"
11   frontend:
12     container_name: frontend
13     restart: always
14     build: ./frontend
15     image: alvarousac/frontend
16     ports:
17     - "3000:3000"
```