

Trabajo Práctico 2 - Introducción a Docker

1- Instalar Docker Community Edition

Obtener la imagen BusyBox

```
Last login: Thu Aug 24 09:59:56 on ttys000
agostinamorellato@Agostinas-MacBook-Pro ~ % docker version
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
Client:
  Cloud integration: v1.0.35-desktop+001
  Version:          24.0.5
  API version:      1.43
  Go version:       go1.20.6
  Git commit:       ced0996
  Built:            Fri Jul 21 20:32:30 2023
  OS/Arch:          darwin/arm64
  Context:          default
agostinamorellato@Agostinas-MacBook-Pro ~ % docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
8a0af25e8c2e: Pull complete
Digest: sha256:3fbc632167424a6d997e74f52b878d7cc478225cffac6bc977eedfe51c7f4e79
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
```

What's Next?

```
View summary of image vulnerabilities and recommendations → docker scout quickview busybox
agostinamorellato@Agostinas-MacBook-Pro ~ % docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
busybox         latest   fc9db2894f4e   5 weeks ago   4.04MB
agostinamorellato@Agostinas-MacBook-Pro ~ %
```

Verificar qué versión y tamaño tiene la imagen bajada, obtener una lista de imágenes locales

```
[agostinamorellato@Agostinas-MacBook-Pro ~ % docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
miproyectowebapi    latest   7f2851ffe4ce   2 days ago   220MB
<none>           <none>  a68278053aa0   2 days ago   932MB
agostinamorellato/mywebapi  latest   e56d491b416b   2 weeks ago  220MB
mywebapi          latest   e56d491b416b   2 weeks ago  220MB
redis              alpine   3f778f9772d5   3 weeks ago  38.5MB
busybox            latest   fc9db2894f4e   7 weeks ago  4.04MB
agostinamorellato@Agostinas-MacBook-Pro ~ % ]
```

2- Ejecutando contenedores:

- Ejecutar un contenedor utilizando el comando run de docker
- Explicar por qué no se obtuvo ningún resultado:

Se intentó ejecutar un contenedor utilizando el comando docker run busybox. El contenedor se inició con la imagen de BusyBox y se ejecutó una vez que se creó, pero lo que hizo fue ejecutar la tarea predeterminada de BusyBox, que es simplemente mostrar un mensaje de salida y luego finalizar. **Una vez que el contenedor completó su tarea, no hubo ninguna aplicación o servicio en ejecución dentro del contenedor que mantuviera vivo el contenedor. Por lo tanto, el contenedor se detuvo automáticamente, y como resultado, no se obtuvo ningún resultado visible en la terminal después de ejecutar el comando.**

- Especificamos algún comando a correr dentro del contenedor (docker run busybox echo "Hola Mundo")
- Ver los contenedores ejecutados utilizando el comando ps:
- Vemos que no existe nada en ejecución entonces usamos **docker ps -a**

- Mostrar el resultado y explicar que se obtuvo como salida del comando anterior.
- docker ps:** muestra solo los contenedores en ejecución en ese momento. Dado que el contenedor creado se ejecutó brevemente para ejecutar el comando echo y luego finalizó, no aparecerá en la salida de docker ps.
- docker ps -a:** Este comando muestra todos los contenedores, independientemente de si están en ejecución o no. Vemos una lista que incluye el ID del contenedor, el estado, el tiempo en que se creó y otros detalles.

```
[agostinamorellato@Agostinas-MacBook-Pro ~ % docker run busybox
[agostinamorellato@Agostinas-MacBook-Pro ~ % docker run busybox echo "Hola Mundo"
Hola Mundo
[agostinamorellato@Agostinas-MacBook-Pro ~ % docker ps
CONTAINER ID   IMAGE      COMMAND   CREATED    STATUS     PORTS      NAMES
[agostinamorellato@Agostinas-MacBook-Pro ~ % docker ps -a
CONTAINER ID   IMAGE      COMMAND   CREATED    STATUS     PORTS      NAMES
PORTS      NAMES
ad76c1f4f6fe  busybox    "echo 'Hola Mundo'"  19 seconds ago  Exited (0) 19 seconds ago
  cranky_jepsen
adfc9f99b35d  busybox    "sh"        About a minute ago  Exited (0) About a minute ago
  quirky_poitras
```

5- Ejecutando en modo interactivo

Ejecutar comando: **docker run -it busybox sh**

Mostrar resultados para cada uno de los comandos:

ps

uptime

free

ls -l /

Salimos del contenedor con **exit**

```
agostinamorellato@Agostinas-MacBook-Pro ~ % docker run -it busybox sh
[ / # ps
[ PID   USER      TIME  COMMAND
  1 root      0:00 sh
  7 root      0:00 ps
/ # uptime
[ 17:39:39 up 0 min,  0 users,  load average: 0.28, 0.07, 0.02
/ # free
[ total        used        free      shared  buff/cache   available ]
Mem:       8039868      482076    6558088      308792      999704      7088644
Swap:      1048572          0      1048572
/ # ls -l /
[total 40
drwxr-xr-x  2 root      root      12288 Jul 17 18:29 bin
drwxr-xr-x  5 root      root      360 Sep  7 17:39 dev
drwxr-xr-x  1 root      root      4096 Sep  7 17:39 etc
drwxr-xr-x  2 nobody    nobody    4096 Jul 17 18:29 home
drwxr-xr-x  2 root      root      4096 Jul 17 18:29 lib
lrwxrwxrwx  1 root      root      3 Jul 17 18:29 lib64 -> lib
dr-xr-xr-x  221 root     root      0 Sep  7 17:39 proc
drwx-----  1 root      root      4096 Sep  7 17:39 root
dr-xr-xr-x  13 root     root      0 Sep  7 17:39 sys
drwxrwxrwt  2 root      root      4096 Jul 17 18:29 tmp
drwxr-xr-x  4 root      root      4096 Jul 17 18:29 usr
drwxr-xr-x  4 root      root      4096 Jul 17 18:29 var
/ #
/ #
/ #
[ / # exit
agostinamorellato@Agostinas-MacBook-Pro ~ %
```

6- Borrando contenedores terminados

- Obtener la lista de contenedores

- Para borrar podemos utilizar el id o el nombre (autogenerado si no se especifica) de contenedor que se desee con el comando : **docker rm elated_lalande** o borrar todos los que no se usen con :

```
docker rm $(docker ps -a -q -f status=exited)
docker container prune
```

```
[/ # exit
[agostinamorellato@Agostinas-MacBook-Pro ~ % docker ps -a
CONTAINER ID   IMAGE      COMMAND      CREATED          STATUS          PORTS
 NAMES
0bc81aeaf04d   busybox    "sh"        About a minute ago   Exited (0) 9 seconds ago
charming_hofstadter
ad76c1f4f6fe   busybox    "echo 'Hola Mundo'"  2 minutes ago    Exited (0) 2 minutes ago
cranky_jepsen
adfc9f9b35d   busybox    "sh"        3 minutes ago    Exited (0) 3 minutes ago
quirky_poitras
[agostinamorellato@Agostinas-MacBook-Pro ~ % docker rm $(docker ps -a -q -f status=exited)
0bc81aeaf04d
ad76c1f4f6fe
adfc9f9b35d
[agostinamorellato@Agostinas-MacBook-Pro ~ % docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
agostinamorellato@Agostinas-MacBook-Pro ~ % ]
```

7-Construir una imagen

- A partir del código <https://github.com/ingsoft3ucc/SimpleWebAPI> crearemos una imagen.
- Clonar repo
- Crear imagen etiquetando con un nombre. El punto final le indica a Docker que use el dir actual

```
[agostinamorellato@Agostinas-MacBook-Pro ~ % cd desktop
[agostinamorellato@Agostinas-MacBook-Pro desktop % git clone git@github.com:ingsoft3ucc/SimpleWebAPI.
git
Cloning into 'SimpleWebAPI'...
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 43 (delta 17), reused 35 (delta 11), pack-reused 0
Receiving objects: 100% (43/43), 9.47 KiB | 2.37 MiB/s, done.
Resolving deltas: 100% (17/17), done.
[agostinamorellato@Agostinas-MacBook-Pro desktop % docker build -t mywebapi .
[+] Building 0.0s (2/2) FINISHED                                            docker:desktop-linux
=> [internal] load .dockerrcignore                                         0.0s
=> => transferring context: 2B                                           0.0s
=> [internal] load build definition from Dockerfile                   0.0s
=> => transferring dockerfile: 2B                                         0.0s
ERROR: failed to solve: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount373180630/
Dockerfile: no such file or directory
[agostinamorellato@Agostinas-MacBook-Pro desktop % cd SimpleWebAPI
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker build -t mywebapi .
[+] Building 56.9s (5/17)                                              docker:desktop-linux
=> [internal] load .dockerrcignore                                         0.0s
=> => transferring context: 2B                                           0.0s
=> [internal] load build definition from Dockerfile                   0.0s
=> => transferring dockerfile: 810B                                     0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0       2.4s
=> [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0     2.8s
=> [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:926b9337622ccc594ed051d3559f1c4 54.1s
=> => resolve mcr.microsoft.com/dotnet/sdk:7.0@sha256:926b9337622ccc594ed051d3559f1c4544686c 0.0s
=> => sha256:5df5175ca8be3457791cd99e55bd8aa65365ae8ceeb35ef16000ff3b217f 14.92MB / 14.92MB 38.0s
=> => sha256:ed434d8d0c6232b0d665022308bd7d5eb42f829871df394be44d9ad35dcf6a1 7.23kB / 7.23kB 0.0s
=> => sha256:1e6e5abd5185e937bd32667ce859aebead9841c9271b759140d91c78a5dd450 2.01kB / 2.01kB 0.0s
=> => sha256:41f92d5a73b9bee296c7b4a3817b28098b22fb60112608b42bb03570ca29 18.87MB / 30.06MB 54.1s
=> => sha256:a64d7b05f29270b0d104db8bac08feaf20c10160db51eab0f89ce701bbf5 23.07MB / 30.72MB 54.1s
=> => sha256:926b9337622ccc594ed051d3559f1c4544686c9fd5f4656f09d1fe37d9a5ace 1.82kB / 1.82kB 0.0s
=> => sha256:c9f99a6f21d7f96dfc15a577262d3d3e3ab3400734105f968aa708813b305554 156B / 156B 38.4s
=> => sha256:e904e1ee5fe4f9bd5502a2a7af39a65b7f231fc90e9f754e030b2ddcd73ec 6.29MB / 9.81MB 54.1s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:7.0@sha256:54a3864f1c7dbb232982f61105aa1 54.1s
=> => resolve mcr.microsoft.com/dotnet/aspnet:7.0@sha256:54a3864f1c7dbb232982f61105aa18a59b9 0.0s
=> => sha256:5df5175ca8be3457791cd99e55bd8aa65365ae8ceeb35ef16000ff3b217f 14.92MB / 14.92MB 38.0s
=> => sha256:a64d7b05f29270b0d104db8bac08feaf20c10160db51eab0f89ce701bbf5 23.07MB / 30.72MB 54.1s
```

- Revisar Dockerfile y explicar cada línea

#See <https://aka.ms/containerfastmode> to understand how Visual Studio uses this Dockerfile to build your images for faster debugging.

```
FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443
EXPOSE 5254

FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
WORKDIR /src
COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]
RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"
COPY . .
WORKDIR "/src/SimpleWebAPI"
RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]
#CMD ["/bin/bash"]
```

#See <https://aka.ms/containerfastmode> to understand how Visual Studio uses this Dockerfile to build your images for faster debugging.: Proporciona información sobre la utilidad de este Dockerfile y cómo Visual Studio lo utiliza para generar imágenes de Docker para depuración más rápida. No afecta la construcción de la imagen en sí.

FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base: Utiliza la imagen oficial de ASP.NET de Microsoft con .NET 7.0 como base. Esta etapa se utilizará para la ejecución de la aplicación.

WORKDIR /app: Esto establece el directorio de trabajo dentro del contenedor en "/app". Todos los comandos siguientes se ejecutarán en este directorio.

EXPOSE 80, EXPOSE 443, EXPOSE 5254: Estos comandos exponen los puertos 80, 443 y 5254 en el contenedor. Esto significa que cualquier aplicación que se ejecute en este contenedor puede ser accesible desde fuera del contenedor a través de estos puertos.

FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build: Esta línea establece la imagen base para la etapa "build". Utiliza la imagen oficial del SDK de .NET 7.0 de Microsoft como base. Esta etapa se utilizará para compilar la aplicación.

WORKDIR /src: Cambia el directorio de trabajo dentro de esta etapa a "/src", donde se copiará y compilará el código fuente.

COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]: Copia el archivo de proyecto "SimpleWebAPI.csproj" y su contenido desde la carpeta "SimpleWebAPI/" en el sistema de archivos local al directorio de trabajo "/src/SimpleWebAPI" en el contenedor.

RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj": Ejecuta el comando "dotnet restore" para restaurar las dependencias del proyecto dentro del contenedor.

COPY . .: Copia todo el contenido del directorio de trabajo actual (en el sistema de archivos local) al directorio de trabajo "/src/SimpleWebAPI" en el contenedor.

WORKDIR "/src/SimpleWebAPI": Cambia el directorio de trabajo dentro del contenedor al directorio del proyecto donde se encuentra el archivo "SimpleWebAPI.csproj".

RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build: Compila el proyecto utilizando el comando "dotnet build" en modo de liberación (" -c Release") y especifica la carpeta de salida como "/app/build".

FROM build AS publish: Esta línea crea una nueva etapa llamada "publish" a partir de la etapa "build". Esta etapa se utilizará para publicar la aplicación.

**RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish
/p:UseAppHost=false:** Publica la aplicación utilizando el comando "dotnet publish" en modo de liberación (" -c Release") y especifica la carpeta de publicación como "/app/publish". La opción "/p:UseAppHost=false" indica que no se utilizará un host de aplicación para la publicación.

FROM base AS final: Esta línea crea una nueva etapa llamada "final" a partir de la etapa "base". Esta etapa se utilizará para la ejecución final de la aplicación.

WORKDIR /app: Cambia el directorio de trabajo dentro de esta etapa a "/app".

COPY --from=publish /app/publish . .: Copia el contenido de la etapa "publish" (la aplicación publicada) al directorio de trabajo "/app" en esta etapa.

ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]: Establece el punto de entrada para la aplicación dentro del contenedor. Cuando se inicia el contenedor, ejecutará la aplicación utilizando el comando "dotnet SimpleWebAPI.dll". Esta línea es típica para aplicaciones .NET.

#CMD ["/bin/bash"]: Esta línea está comentada, por lo que no se ejecutará. Si se descomentara, se configuraría un comando predeterminado para ejecutar "/bin/bash" en lugar de la aplicación principal. Actualmente, está desactivado y la aplicación .NET es el punto de entrada.

- Ver imágenes disponibles

mywebapi	e56d491b416b	latest	Unused	0 seconds ago	219.62 MB	►	:	✖
busybox	fc9db2894f4e	latest	Unused	1 month ago	4.04 MB	►	:	✖

Showing 2 items

- Ejecutar un contenedor con nuestra imagen

```
View summary of image vulnerabilities and recommendations → docker scout quickview
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker images
REPOSITORY      TAG      IMAGE ID      CREATED             SIZE
mywebapi        latest   e56d491b416b   About a minute ago   220MB
busybox         latest   fc9db2894f4e   5 weeks ago       4.04MB
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % ]
```



```
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker run mywebapi
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % ]
```

- Subir imagen a nuestra cuenta de docker hub

```
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker tag mywebapi agostinamorellato/mywebapi
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker push agostinamorellato/mywebapi
Using default tag: latest
The push refers to repository [docker.io/agostinamorellato/mywebapi]
ce69e606933e: Pushed
5f70bf18a086: Pushed
d87bc14b7341: Pushed
6a5abc030d56: Pushed
699984159708: Pushed
383b68f57520: Pushed
4b3f908aac7a: Pushed
9136a4eb6504: Pushed
latest: digest: sha256:9d3d8b4098f8fb6d675911843acf3435d6e748e607239132ebe5be1f0242388c size: 1994
agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % ]
```

8- Publicando puertos

- Ejecutar la siguiente imagen, en este caso utilizamos la bandera -d (detach) para que nos devuelva el control de la consola
- Ejecutamos un comando ps

```
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker run --name myapi -d mywebapi
72ba56a43b61dcf6f31ab0988531852cde3d2fc0153207d4ee1b8869d1de9c37
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
72ba56a43b61 mywebapi "dotnet SimpleWebAPI..." 12 seconds ago Up 11 seconds 80/tcp, 443/tcp,
5254/tcp myapi
agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % ]
```

- Vemos que el contendor expone 3 puertos el 80, el 5254 y el 443, pero si intentamos en un navegador acceder a <http://localhost/WeatherForecast> no sucede nada.



This site can't be reached

localhost refused to connect.

Try:

- Checking the connection
- [Checking the proxy and the firewall](#)

ERR_CONNECTION_REFUSED

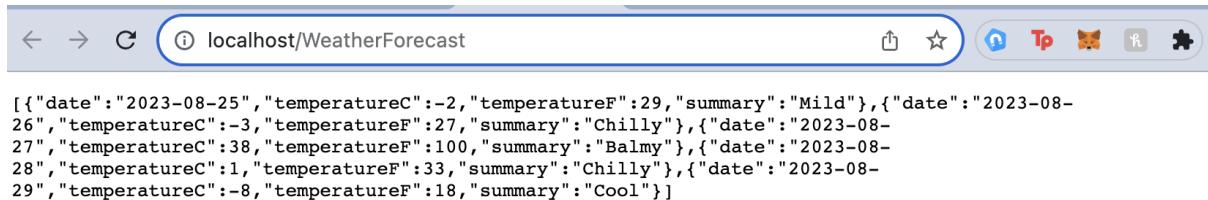
[Details](#)

[Reload](#)

- Procedemos entonces a parar y remover este contenedor
- Vamos a volver a correrlo otra vez, pero publicando los puertos 80 y 5254

```
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker kill myapi ]  
myapi  
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker rm myapi ]  
myapi  
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker run --name myapi -d -p 80:80 -p 5254:5]  
254 mywebapi  
88faaa97a9ff1f45a1863c3fb30f2c8dc5a37392ce2ba3c03b95264b77f249dc  
agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % ]
```

- Accedamos nuevamente a <http://localhost/WeatherForecast> y a <http://localhost/swagger/index.html> y expliquemos que sucede.



En la última parte del ejercicio, después de haber ejecutado el contenedor utilizando el comando docker run con la opción -p para publicar los puertos 80 y 5254, se han expuesto estos puertos desde el contenedor al sistema host. Al exponer y mapear los puertos del contenedor al sistema host, puedes acceder a la aplicación en ejecución en el contenedor desde el sistema host utilizando un navegador web. Esto facilita la interacción con la aplicación y la prueba de sus funcionalidades.

9- Modificar Dockerfile para soportar bash

- Modificamos dockerfile para que entre en bash sin ejecutar automáticamente la app
`#ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]`
`CMD ["/bin/bash"]`

The screenshot shows a code editor with a Dockerfile. The file contains instructions to build the application using .NET Core, copy the project files, restore dependencies, build the project, publish it, and finally run the application in bash. The last two lines of the Dockerfile are highlighted in red, indicating they are being modified.

```
FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443
EXPOSE 5254

FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
WORKDIR /src
COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]
RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"
COPY .
WORKDIR "/src/SimpleWebAPI"
RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
#ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]
CMD ["/bin/bash"]
```

- Rehacemos la imagen

- Corremos contenedor en modo interactivo exponiendo puerto
docker run -it --rm -p 80:80 mywebapi

```
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker run -it --rm -p 80:80 mywebapi
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
  Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
  Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
  Content root path: /app
```

- Navegamos a <http://localhost/weatherforecast>
- Vemos que no se ejecuta automáticamente



This localhost page can't be found

No webpage was found for the web address: <http://localhost/swagger/index.html>
HTTP ERROR 404

[Reload](#)

- Ejecutamos app:
dotnet SimpleWebAPI.dll

```
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker run -it --rm -p 80
:80 mywebapi
info: Microsoft.Hosting.Lifetime[14]
  Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
  Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
  Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
  Content root path: /app
[dotnet SimpleWebAPI.dll
warn: Microsoft.AspNetCore.HttpsPolicy.HttpsRedirectionMiddleware[3]
  Failed to determine the https port for redirect.
^Cinfo: Microsoft.Hosting.Lifetime[0]
  Application is shutting down...
```

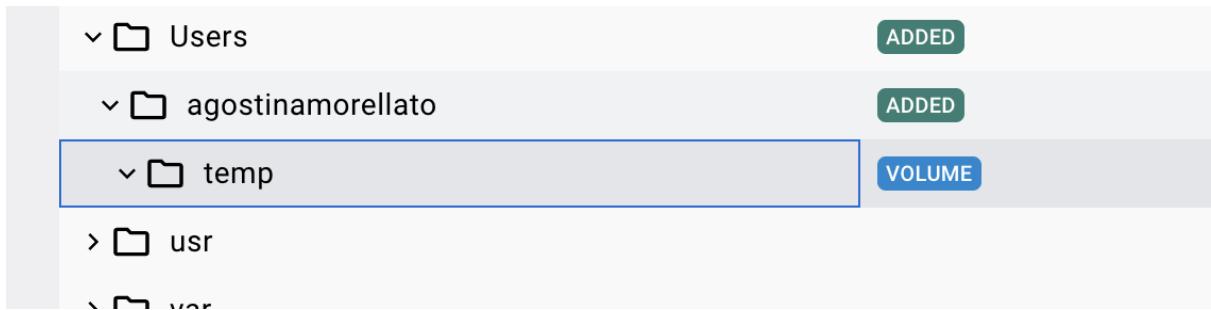


```
[{"date": "2023-08-25", "temperatureC": 45, "temperatureF": 112, "summary": "Sweltering"}, {"date": "2023-08-26", "temperatureC": 24, "temperatureF": 75, "summary": "Freezing"}, {"date": "2023-08-27", "temperatureC": 35, "temperatureF": 94, "summary": "Hot"}, {"date": "2023-08-28", "temperatureC": 44, "temperatureF": 111, "summary": "Cool"}, {"date": "2023-08-29", "temperatureC": 42, "temperatureF": 107, "summary": "Scorching"}]
```

10- Montando volúmenes

- Ejecutar el siguiente comando, cambiar myusuario por el usuario que corresponda.
En Mac puede utilizarse /Users/miusuario/temp)
- Ejecutar el siguiente comando, cambiar myusuario por el usuario que corresponda.
En Mac puede utilizarse /Users/miusuario/temp):
docker run -it --rm -p 80:80 -v /Users/miuser/temp:/var/temp mywebapi
- Dentro del contenedor correr
ls -l /var/temp
touch /var/temp/hola.txt
- Verificar que el archivo se ha creado en el directorio del guest y del host.

```
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker run -it --rm -p 80:80 -v /Users/agostinamorellato/temp mywebapi
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://[::]:80
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /app
ls -l /var/temp
touch /var/temp/hola.txt
```



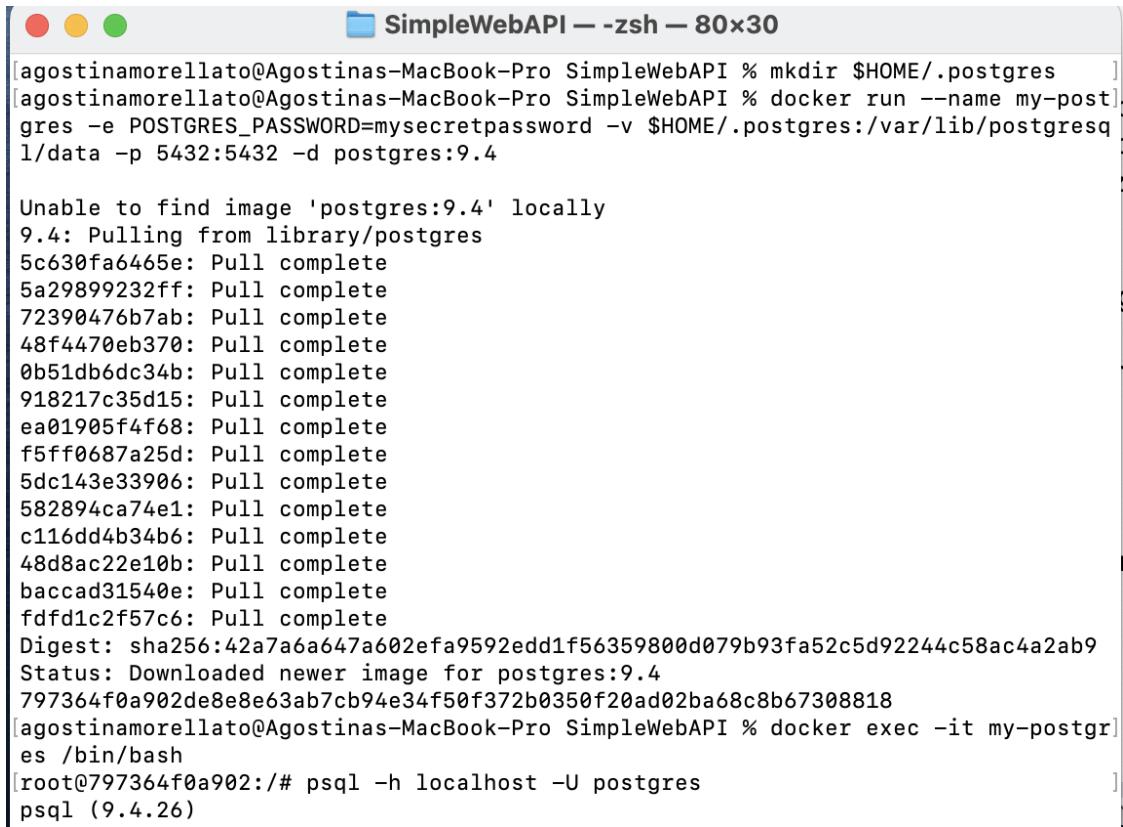
11- Utilizando una base de datos

- Levantar una base de datos PostgreSQL:

```
mkdir $HOME/.postgres
docker run --name my-postgres -e
POSTGRES_PASSWORD=mysecretpassword -v
$HOME/.postgres:/var/lib/postgresql/data -p 5432:5432 -d postgres:9.4
```

- Ejecutar sentencias utilizando esta instancia:

```
docker exec -it my-postgres /bin/bash
psql -h localhost -U postgres
```



```
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % mkdir $HOME/.postgres
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker run --name my-postgres -e POSTGRES_PASSWORD=mysecretpassword -v $HOME/.postgres:/var/lib/postgresql/data -p 5432:5432 -d postgres:9.4

Unable to find image 'postgres:9.4' locally
9.4: Pulling from library/postgres
5c630fa6465e: Pull complete
5a29899232ff: Pull complete
72390476b7ab: Pull complete
48f4470eb370: Pull complete
0b51db6dc34b: Pull complete
918217c35d15: Pull complete
ea01905f4f68: Pull complete
f5ff0687a25d: Pull complete
5dc143e33906: Pull complete
582894ca74e1: Pull complete
c116dd4b34b6: Pull complete
48d8ac22e10b: Pull complete
baccad31540e: Pull complete
fdfd1c2f57c6: Pull complete
Digest: sha256:42a7a6a647a602efa9592edd1f56359800d079b93fa52c5d92244c58ac4a2ab9
Status: Downloaded newer image for postgres:9.4
797364f0a902de8e8e63ab7cb94e34f50f372b0350f20ad02ba68c8b67308818
[agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI % docker exec -it my-postgres /bin/bash
[root@797364f0a902:/# psql -h localhost -U postgres
psql (9.4.26)
```

- #Estos comandos se corren una vez conectados a la base

```

\

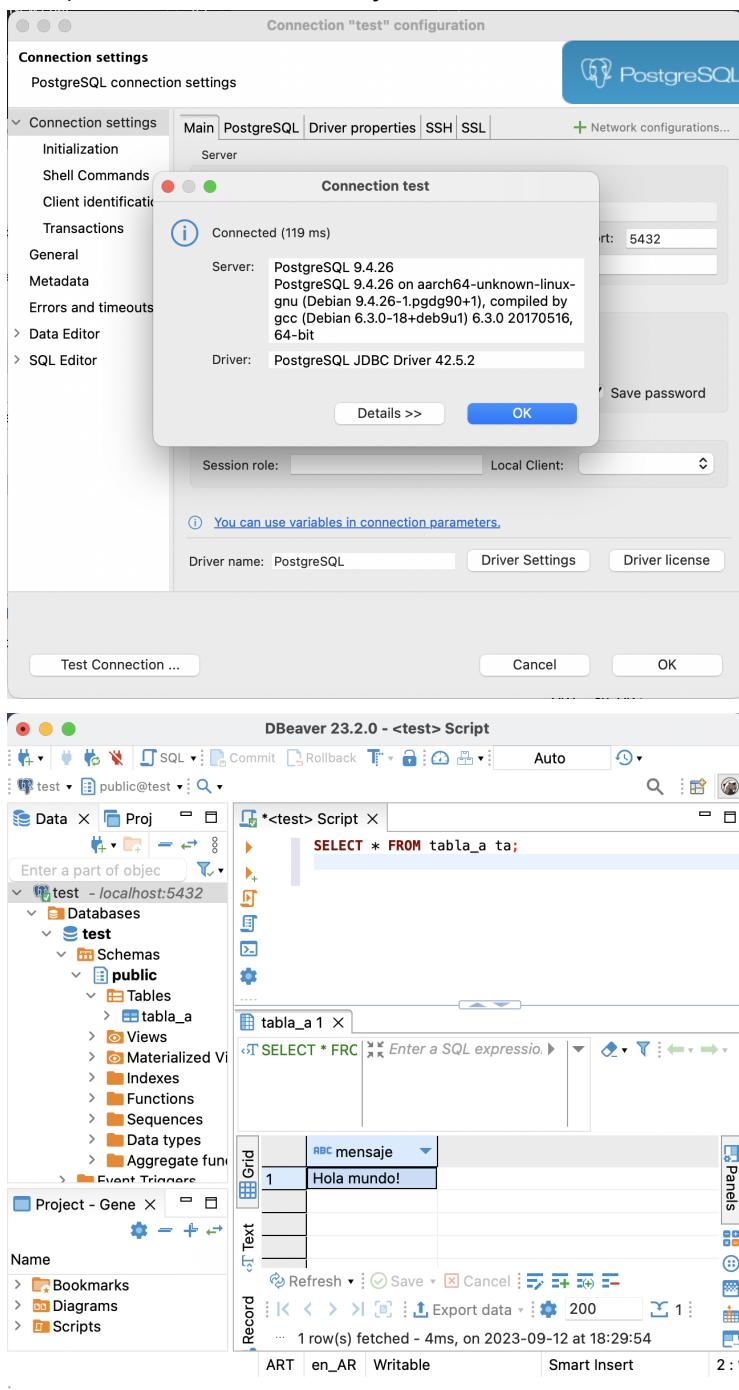
create database test;
\connect test
create table tabla_a (mensaje varchar(50));
insert into tabla_a (mensaje) values('Hola mundo!');
select * from tabla_a;
\q
exit

[postgres=# \l
                                         List of databases
  Name    |  Owner   | Encoding | Collate      | Ctype       | Access privileges
-----+-----+-----+-----+-----+-----+
postgres | postgres | UTF8     | en_US.utf8  | en_US.utf8  | =c/postgres
template0 | postgres | UTF8     | en_US.utf8  | en_US.utf8  | =c/postgres
+
|         |          |          |          |          |
|         |          |          |          |          | postgres=CTc/postgres
es
template1 | postgres | UTF8     | en_US.utf8  | en_US.utf8  | =c/postgres
+
|         |          |          |          |          |
|         |          |          |          |          | postgres=CTc/postgres
es
(3 rows)

[postgres=# create database test;
CREATE DATABASE
[postgres=# \connect test
You are now connected to database "test" as user "postgres".
[test=# create table tabla_a (mensaje varchar(50));
CREATE TABLE
test=# insert into tabla_a (mensaje) values('Hola mundo!');
INSERT 0 1
[test=# select * from tabla_a;
  mensaje
-----
  Hola mundo!
(1 row)

[test=# \q
[root@797364f0a902:/# exit
exit
agostinamorellato@Agostinas-MacBook-Pro SimpleWebAPI %
]
```

Conectarse a la base utilizando alguna IDE (Dbeaver - <https://dbeaver.io/>, eclipse, IntelliJ, etc...). Interactuar con los objetos creados



DBeaver 23.2.0 - <test> Script

test public@test

Data X Proj

Enter a part of object

test - localhost:5432

Databases test

Schemas public

Tables tabla_a

Views

Materialized Vi

Indexes

Functions

Sequences

Data types

Aggregate fun

Event Triggers

Project - Gene X

Name Bookmarks Diagrams Scripts

ART en_AR Writable Smart Insert 2 : 1

*<test> Script X

```
SELECT * FROM tabla_a ta;
INSERT INTO tabla_a (mensaje) VALUES ('Hola');
```

Statistics 1 X

Name	Value
Updated Rows 1	

Query INSERT INTO tabla_a (mensaje) VALUES ('Hola')

Start time Tue Sep 12 18:37:44 ART 2023

Finish time Tue Sep 12 18:37:44 ART 2023

Panes

DBeaver 23.2.0 - <test> Script

test public@test

Data X Proj

Enter a part of object

test - localhost:5432

Databases test

Schemas public

Tables tabla_a

Views

Materialized Vi

Indexes

Functions

Sequences

Data types

Aggregate fun

Event Triggers

Project - Gene X

Name Bookmarks Diagrams Scripts

ART en_AR Writable Smart Insert 5 : 1

*<test> Script X

```
SELECT * FROM tabla_a ta;
INSERT INTO tabla_a (mensaje) VALUES ('Hola');
```

tabla_a 1 X

SELECT * FRC Enter a SQL expression

Grid

	ABC mensaje
1	Hola mundo!
2	Hola

Text

Record

Refresh Save Cancel Export data 200 2

2 row(s) fetched - 2ms, on 2023-09-12 at 18:37:53

Explicar que se logró con el comando docker run y docker exec ejecutados en este ejercicio.

A lo largo del ejercicio, se realizaron tareas específicas utilizando los comandos docker run y docker exec en Docker.

docker run:

- Creación y Ejecución de Contenedores
- Descarga de Imágenes, utilizándolas como base para los contenedores.
- Configuración de Contenedores mediante la especificación de opciones como puertos expuestos.
- Exposición de puertos dentro de los contenedores para permitir la comunicación con aplicaciones que se ejecutan en esos puertos.
- Gestión de Credenciales: Se proporcionaron contraseñas para acceder a servicios dentro de los contenedores, como bases de datos.

docker exec:

- Ejecución de Comandos en Contenedores en Ejecución sin necesidad de iniciar un nuevo contenedor. Esto permitió interactuar con aplicaciones y servicios en tiempo real.
- Acceso a Shells o Consolas dentro de un contenedor en ejecución, lo que facilitó la depuración y la ejecución de comandos específicos dentro del contenedor.