```
C:\Users\Manu>docker version
Client:
 Version:           27.1.1
 API version:       1.46
 Go version:        go1.21.12
 Git commit:        6312585
 Built:             Tue Jul 23 19:57:57 2024
 OS/Arch:           windows/amd64
 Context:           desktop-linux

Server: Docker Desktop 4.33.1 (161083)
 Engine:
  Version:          27.1.1
  API version:      1.46 (minimum version 1.24)
  Go version:       go1.21.12
  Git commit:       cc13f95
  Built:            Tue Jul 23 19:57:19 2024
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.19
  GitCommit:        2bf793ef6dc9a18e00cb12efb64355c2c9d5eb41
 runc:
  Version:          1.7.19
  GitCommit:        v1.1.13-0-g58aa920
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

OBTENER IMAGEN BUSYBOX:

- Utilizar el comando *"docker pull busybox".*

```
C:\Users\Manu>docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
ec562eabd705: Pull complete
Digest: sha256:9ae97d36d26566ff84e8893c64a6dc4fe8ca6d1144bf5b87b2b85a32def253c7
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview busybox
```

- Utilizar el comando *“docker images”* para verificar cuál versión y tamaño tienen las imágenes bajadas y realizar una lista de imágenes.

```
C:\Users\Manu>docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
busybox      latest    65ad0d468eb1   15 months ago    4.26MB
```

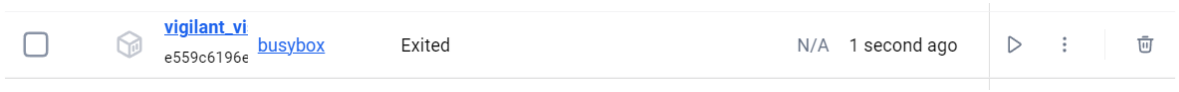
EJECUTAR CONTENEDORES:

- Utilizar el comando *“docker run busybox”* para ejecutar el contenedor.

```
C:\Users\Manu>docker run busybox
```

- Utilizar el comando *docker run busybox echo "Hola Mundo".*

```
C:\Users\Manu>docker run busybox echo "Hola Mundo"
Hola Mundo
```



- Utilizar el comando *“docker ps”* para ver los contenedores utilizados.

```
C:\Users\Manu>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Como no se encuentra ningún contenedor en ejecución, utilizamos el comando *“docker ps -a”*

```
C:\Users\Manu>docker ps -a
CONTAINER ID   IMAGE     COMMAND              CREATED         STATUS                     PORTS     NAMES
e559c6196e78   busybox   "echo 'Hola Mundo'"  4 minutes ago   Exited (0) 4 minutes ago             vigilant_visvesvaraya
1b405fecd2e8   busybox   "sh"                 6 minutes ago   Exited (0) 6 minutes ago             recursing_faraday
```

Explicando un poco, en la salida se puede observar lo siguiente:
CONTAINER ID: 0bf82d4df2da es el identificador del contenedor.
IMAGE: busybox, que es la imagen de Docker utilizada para crear el contenedor.
COMMAND: "echo 'Hola Mundo'", es el comando que se ejecutó dentro del contenedor.
CREATED: 4 minutes ago, indica que el contenedor fue creado hace 4 minutos
STATUS: Exited (0) 4 minutes ago, indica que el contenedor se ejecutó y luego terminó su ejecución (con un código de salida 0, lo que significa que se ejecutó correctamente) hace 4 minutos.
NAMES: vigilant_visvessvaraya, es el nombre asignado automáticamente al contenedor.

EJECUTAR EN MODO INTERACTIVO:

- Utilizar el comando *“docker run -it busybox sh”.*

```
C:\Users\Manu>docker run -it busybox sh
/ # ps
PID   USER     TIME  COMMAND
    1 root      0:00 sh
    7 root      0:00 ps
/ # ls
bin    dev    etc    home   lib    lib64  proc   root   sys    tmp    usr    var
/ #  free
              total        used        free      shared  buff/cache   available
Mem:        3003172      853268     1718252        4208      431652     1986600
Swap:       1048576        1036     1047540
/ # ls -l
total 40
drwxr-xr-x    2 root     root         12288 May 18  2023 bin
drwxr-xr-x    5 root     root           360 Aug 27 20:00 dev
drwxr-xr-x    1 root     root          4096 Aug 27 20:00 etc
drwxr-xr-x    2 nobody   nobody        4096 May 18  2023 home
drwxr-xr-x    2 root     root          4096 May 18  2023 lib
lrwxrwxrwx    1 root     root             3 May 18  2023 lib64 -> lib
dr-xr-xr-x  255 root     root             0 Aug 27 20:00 proc
drwx------    1 root     root          4096 Aug 27 20:01 root
dr-xr-xr-x   11 root     root             0 Aug 27 20:00 sys
drwxrwxrwt    2 root     root          4096 May 18  2023 tmp
drwxr-xr-x    4 root     root          4096 May 18  2023 usr
drwxr-xr-x    4 root     root          4096 May 18  2023 var
/ #
/ # uptime
 20:03:07 up 29 min,  0 users,  load average: 0.01, 0.02, 0.05
/ #
```

Con el comando *“exit”* salimos del contenedor.

BORRAR CONTENEDORES:

- Obtener la lista de contenedores con el comando “*docker ps -a”.*

```
C:\Users\Manu>docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS                     PORTS     NAMES
35b6ffd4b934   busybox   "sh"                     7 minutes ago   Exited (127) 4 minutes ago           crazy_benz
e559c6196e78   busybox   "echo 'Hola Mundo'"      16 minutes ago  Exited (0) 16 minutes ago            vigilant_visvesvaraya
1b405fecd2e8   busybox   "sh"                     19 minutes ago  Exited (0) 19 minutes ago            recursing_faraday
```

- Para borrar un contenedor, podemos utilizar el ID o el nombre del mismo. Utilizamos el comando *“docker rm (nombre del contenedor)”.*

```
C:\Users\Manu>docker rm vigilant_visvesvaraya
vigilant_visvesvaraya
```

- Para borrar todos los contenedores podemos utilizar los siguientes comandos:
  - *“docker rm $(docker ps -a -q -f status=exited)”*
  - *“docker container prune”*

CONSTRUIR UNA IMAGEN:

- Clonar el siguiente repositorio: https://github.com/ingsoft3ucc/SimpleWebAPI

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3>git clone https://github.com/ingsoft3ucc/SimpleWebAPI.git
Cloning into 'SimpleWebAPI'...
remote: Enumerating objects: 150, done.
remote: Counting objects: 100% (150/150), done.
remote: Compressing objects: 100% (115/115), done.
Receiving objects:  60% (90/150), 20.00 KiB | 7.00 KiB/sreused 0 (from 0)
Receiving objects: 100% (150/150), 28.41 KiB | 16.00 KiB/s, done.
Resolving deltas: 100% (60/60), done.
```

- Crear una imágen etiquetándola con un nombre. El punto final le indica a Docker que use el dir actual *“docker build -t mywebapi .”*

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\SimpleWebAPI>docker build -t mywebapi .
[+] Building 452.7s (18/18) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 810B
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:7.0@sha256:c7d9ee6cd01afe9aa80642e577c7cec9f5d87f88e5d70bd36fd61072079bc55b
 => => resolve mcr.microsoft.com/dotnet/aspnet:7.0@sha256:c7d9ee6cd01afe9aa80642e577c7cec9f5d87f88e5d70bd36fd61072079bc55b
 => => sha256:c7d9ee6cd01afe9aa80642e577c7cec9f5d87f88e5d70bd36fd61072079bc55b 1.79kB / 1.79kB
 => => sha256:8a7717ff21c245feacd25dae5ff23306aae0e058578725ad987fd4b8509c36ec 1.37kB / 1.37kB
 => => sha256:ec861be017681c5da7a762bd29eb07b4acd2391a9fa5ca9150ad6e5554c0506c 2.34kB / 2.34kB
 => => sha256:728328ac3bde9b85225b1f0d60f5c149f5635a191f5d8eaeeb00e095d36ef9fd 31.43MB / 31.43MB
 => => sha256:534ba947de6ac79fd6168f4a93847954b23bab2782700bbfff7f31e61a03e8d4 32.46MB / 32.46MB
 => => sha256:82bb7a80de578404d92b5ae5e67f3de90eab30027694d2609be35ad25b09e3bc 14.97MB / 14.97MB
 => => sha256:f1b39e168c1c776458e172f157167607b9fd3cc550af8e6ff0a7dd363c1e64ea 153B / 153B
 => => sha256:f194078e85f8008c084163778aaf266434f7182e0d4f783646f41b388c88a13a 10.12MB / 10.12MB
 => => extracting sha256:728328ac3bde9b85225b1f0d60f5c149f5635a191f5d8eaeeb00e095d36ef9fd
 => => extracting sha256:82bb7a80de578404d92b5ae5e67f3de90eab30027694d2609be35ad25b09e3bc
```

Welcome | Dockerfile

Dockerfile > ...

```
1   #See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile to build your image
2
3   FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
4   WORKDIR /app
5   EXPOSE 80
6   EXPOSE 443
7   EXPOSE 5254
8
9   FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
10  WORKDIR /src
11  COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"]
12  RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"
13  COPY . .
14  WORKDIR "/src/SimpleWebAPI"
15  RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build
16
17  FROM build AS publish
18  RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false
19
20  FROM base AS final
21  WORKDIR /app
22  COPY --from=publish /app/publish .
23  ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"]
24  #CMD ["/bin/bash"]
25
```

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.: proporciona un enlace para entender cómo Visual Studio utiliza el Dockerfile para construir imágenes de manera más eficiente durante la depuración.

FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base : Establece la imagen base para la aplicación, que en este caso es mcr.microsoft.com/dotnet/aspnet:7.0, una imagen oficial de .NET ASP.NET Core Runtime. Esta imagen está optimizada para ejecutar aplicaciones ASP.NET Core.
WORKDIR /app : Establece el directorio de trabajo dentro del contenedor en /app. Cualquier comando subsiguiente se ejecutará desde este directorio.
EXPOSE 80 EXPOSE 443 EXPOSE 5254 :Documenta que la aplicación escucha en los puertos 80 (HTTP), 443 (HTTPS) y 5254. Esto no expone los puertos fuera del contenedor; solo documenta que estos puertos están en uso dentro del contenedor. Puedes mapear estos puertos al host cuando ejecutes el contenedor.
FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build : Cambia a una imagen diferente, mcr.microsoft.com/dotnet/sdk:7.0, que incluye el SDK de .NET 7.0. Esta imagen contiene todas las herramientas necesarias para construir y compilar aplicaciones .NET.
WORKDIR /src : Cambia el directorio de trabajo a /src, donde se llevará a cabo el proceso de construcción de la aplicación
COPY ["SimpleWebAPI/SimpleWebAPI.csproj", "SimpleWebAPI/"] : Copia el archivo de proyecto SimpleWebAPI.csproj desde tu máquina local al contenedor, ubicándolo en el directorio /src/SimpleWebAPI. Esto se hace para que las dependencias se restauren basándose en este archivo.
RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj" : Ejecuta el comando dotnet restore, que restaura todas las dependencias y paquetes NuGet definidos en SimpleWebAPI.csproj
COPY . . : Copia todos los archivos y carpetas desde el contexto de construcción local al directorio de trabajo actual del contenedor (/src). Esto incluye todo el código fuente de la aplicación
WORKDIR "/src/SimpleWebAPI" : Cambia el directorio de trabajo al de la aplicación dentro del contenedor (/src/SimpleWebAPI), donde se encuentra el código de la aplicación.
RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build : Compila la aplicación en modo de lanzamiento (Release) y coloca los archivos compilados en el directorio /app/build.
FROM build AS publish: Utiliza la imagen de la etapa build como base para la etapa publish, donde se publica la aplicación.
RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false: Publica la aplicación compilada, preparando los archivos para su despliegue. Coloca los archivos resultantes en el directorio /app/publish. El parámetro /p:UseAppHost=false indica que no se debe generar un ejecutable nativo específico de la plataforma, lo cual es útil para reducir el tamaño de la imagen final.
FROM base AS final : Cambia a la imagen base definida al principio (base), que es la imagen de ASP.NET Core Runtime. Esto es para minimizar el tamaño de la imagen final, ya que solo contiene lo necesario para ejecutar la aplicación, no las herramientas de desarrollo
WORKDIR /app: Establece nuevamente el directorio de trabajo en /app.

COPY --from=publish /app/publish . : Copia los archivos publicados desde la etapa publish a la imagen final, colocándolos en el directorio /app
ENTRYPOINT ["dotnet", "SimpleWebAPI.dll"] : Define el comando que se ejecutará cuando se inicie el contenedor. En este caso, se ejecuta dotnet SimpleWebAPI.dll, lo que lanza la aplicación ASP.NET Core.
#CMD ["/bin/bash"] : Esta línea está comentada. Si estuviera activa, sobrescribiría el ENTRYPOINT anterior con el comando /bin/bash, que abriría una sesión de terminal bash en lugar de ejecutar la aplicación. Esto es útil para depuración, pero generalmente no se deja activo en una imagen de producción.

- Ver imágenes disponibles.

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\SimpleWebAPI>docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
mywebapi     latest    c0af636b9e14   3 minutes ago   216MB
busybox      latest    65ad0d468eb1   15 months ago   4.26MB
```

- *docker tag <nombre_imagen_local> <tu_usuario_dockerhub>/<nombre_imagen>:*

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\SimpleWebAPI>docker tag mywebapi manuromeromedina/mywebapi:latest
```

- Publicar el docker.

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\SimpleWebAPI>docker push manuromeromedina/mywebapi:latest
The push refers to repository [docker.io/manuromeromedina/mywebapi]
35f1c593cc44: Pushed
5f70bf18a086: Pushed
6f3e85a298f9: Pushed
270f7fde987a: Pushed
4d29f6e29d10: Pushed
b4ec6db9c251: Pushed
ba941484fbe1: Pushed
123eef91533f: Pushed
latest: digest: sha256:c3f5ff1226cd2e945d0a6466e6a69811616657ae80cd5fa927f4401aafe2c2f9 size: 1995
```

- Verificar la publicación con un docker pull.

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\SimpleWebAPI>docker pull manuromeromedina/mywebapi:latest
latest: Pulling from manuromeromedina/mywebapi
Digest: sha256:c3f5ff1226cd2e945d0a6466e6a69811616657ae80cd5fa927f4401aafe2c2f9
Status: Image is up to date for manuromeromedina/mywebapi:latest
docker.io/manuromeromedina/mywebapi:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview manuromeromedina/mywebapi:latest
```

PUBLICANDO PUERTOS:

- Ejecutar la siguiente imagen, en este caso utilizamos la bandera -d (detach) para que nos devuelva el control de la consola: *docker run --name myapi -d mywebapi*

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\SimpleWebAPI>docker run --name myapi -d mywebapi
d58a103a8bc1d1183f1a17d172c62d82a27387cb2e7c084fe6db9070e9eecb2a
```

- Ejecutamos un comando ps.

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\SimpleWebAPI>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED             STATUS             PORTS                      NAMES
d58a103a8bc1   mywebapi   "dotnet SimpleWebAPI…"   About a minute ago   Up About a minute   80/tcp, 443/tcp, 5254/tcp   myapi
```

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.

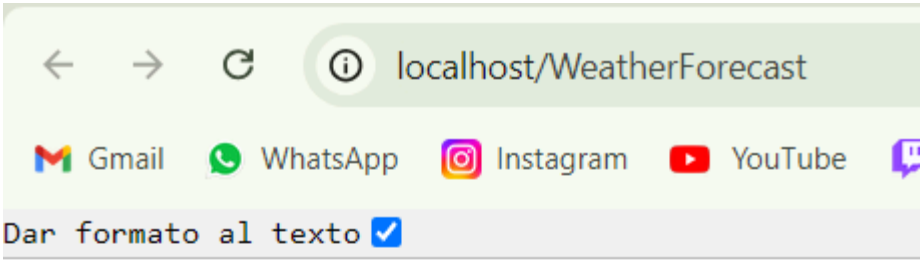
- Procedemos entonces a parar y remover este contenedor: *docker kill myapi*

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\SimpleWebAPI>docker kill myapi
myapi
```

- Vamos a volver a correrlo otra vez, pero publicando el puerto 80 *docker run --name myapi -d -p 80:80 mywebapi*

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\SimpleWebAPI>docker run --name myapi -d -p 80:80 mywebapi
d06ffe517ce72ffdc9ec8b360ef3b496e3e11228acced799d450836adfabff38
```

- Ahora nos dan los datos.




```
[
  {
    "date": "2024-08-28",
    "temperatureC": 41,
    "temperatureF": 105,
    "summary": "Sweltering"
  },
  {
    "date": "2024-08-29",
    "temperatureC": 2,
    "temperatureF": 35,
    "summary": "Balmy"
  },
  {
    "date": "2024-08-30",
    "temperatureC": -10,
    "temperatureF": 15,
    "summary": "Scorching"
  },
  {
    "date": "2024-08-31",
    "temperatureC": -12,
    "temperatureF": 11,
    "summary": "Scorching"
  },
  {
    "date": "2024-09-01",
    "temperatureC": -10,
    "temperatureF": 15,
    "summary": "Helado"
  }
]
```

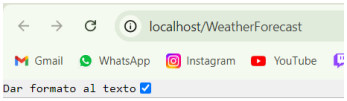
MODIFICAR DOCKERFILE PARA SOPORTAR BASH:

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\SimpleWebAPI>docker build -t mywebapi .
[+] Building 17.2s (18/18) FINISHED                                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                 0.0s
 => => transferring dockerfile: 810B                                                                                                 0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0                                                                    3.3s
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:7.0                                                                 4.4s
 => [internal] load .dockerignore                                                                                                    0.0s
 => => transferring context: 2B                                                                                                      0.0s
 => [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:d32bd65cf5843f413e81f5d917057c82da99737cb1637e905a1a4bc2e7ec6c8d       0.0s
 => [internal] load build context                                                                                                    0.0s
 => => transferring context: 2.89kB                                                                                                  0.0s
 => [base 1/2] FROM mcr.microsoft.com/dotnet/aspnet:7.0@sha256:c7d9ee6cd01afe9aa80642e577c7cec9f5d87f88e5d70bd36fd61072079bc55b      0.0s
 => CACHED [build 2/7] WORKDIR /src                                                                                                  0.0s
 => CACHED [build 3/7] COPY [SimpleWebAPI/SimpleWebAPI.csproj, SimpleWebAPI/]                                                        0.0s
 => CACHED [build 4/7] RUN dotnet restore "SimpleWebAPI/SimpleWebAPI.csproj"                                                         0.0s
 => [build 5/7] COPY . .                                                                                                             0.1s
 => [build 6/7] WORKDIR /src/SimpleWebAPI                                                                                            0.1s
 => [build 7/7] RUN dotnet build "SimpleWebAPI.csproj" -c Release -o /app/build                                                      9.1s
 => [publish 1/1] RUN dotnet publish "SimpleWebAPI.csproj" -c Release -o /app/publish /p:UseAppHost=false                            3.2s
 => CACHED [base 2/2] WORKDIR /app                                                                                                   0.0s
 => CACHED [final 1/2] WORKDIR /app                                                                                                  0.0s
 => CACHED [final 2/2] COPY --from=publish /app/publish .                                                                            0.0s
 => exporting to image                                                                                                               0.0s
 => => exporting layers                                                                                                              0.0s
 => => writing image sha256:c0af636b9e14a2fe579b2f19cdc8a75290762cc1c729929f735688e1e52263b6                                         0.0s
 => => naming to docker.io/library/mywebapi                                                                                          0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

- Corremos contenedor en modo interactivo exponiendo puerto *docker run -it --rm -p 80:80 mywebapi*. Navegamos a http://localhost/weatherforecast Vemos que no se ejecuta automáticamente Ejecutamos app: dotnet SimpleWebAPI.dll

```
C:\Users\Manu\Desktop\INGENIERÍA EN SISTEMAS UCC\CUARTO AÑO\ING SOFT 3\ingsoft3\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
```




```
[
  {
    "date": "2024-08-28",
    "temperatureC": 41,
    "temperatureF": 105,
    "summary": "Sweltering"
  },
  {
    "date": "2024-08-29",
    "temperatureC": 2,
    "temperatureF": 35,
    "summary": "Balmy"
  },
  {
    "date": "2024-08-30",
    "temperatureC": -10,
    "temperatureF": 15,
    "summary": "Scorching"
  },
  {
    "date": "2024-08-31",
    "temperatureC": -12,
    "temperatureF": 11,
    "summary": "Scorching"
  },
  {
    "date": "2024-09-01",
    "temperatureC": -10,
    "temperatureF": 15,
    "summary": "Helado"
  }
]
```