

7.4-1 | Grundgerüst eines Minimal API

1. Installation .NET

Prüfen Sie mit folgendem Command, ob .NET 8 bereits installiert ist.

```
dotnet --list-sdks
```

Falls .NET 8 nicht aufgeführt ist, installieren Sie es wie folgt:

```
# Install dotnet 8 sdk
sudo apt-get update
sudo apt-get install -y dotnet-sdk-8.0
```

mit folgendem Command überprüfen Sie, ob .NET8 erfolgreich installiert ist.

```
dotnet --list-sdks
```

```
vmadmin@lp-22-04:~$ dotnet --list-sdks
8.0.115 [/usr/lib/dotnet/sdk]
```

2. Git Repository erstellen

Erstellen Sie ein leeres GIT-Repository "min-api-with-mongo" für Ihr Projekt. Fügen Sie dem Projekt ein README.md hinzu.

→ Beim Erstellen des Repositories muss die Visibility auf **Public** gesetzt sein weil sonst der Error auftaucht:

```
fatal: Authentication failed for 'https://github.com/technxlgy/min-api-with-mongo.git/'
```

3. Grundgerüst erstellen

Klonen Sie das frisch angelegte Projekt mit git clone auf Ihre VM in ein beliebiges Verzeichnis (z.B. ~/Documents).

```
mkdir min-api
cd min-api
git clone [HTTPS von GitHub]
```

Navigieren Sie in das Projektverzeichnis min-api-with-mongo.

```
cd min-api-with-mongo
```

Erstellen Sie ein .NET Projekt WebApi mit Template web:

```
dotnet new web --name WebApi --framework net8.0
```

```
vmadmin@lp-22-04:~$ cd Documents/
vmadmin@lp-22-04:~/Documents$ git clone https://github.com/technxlxgy/min-api-with-mongo.git
Cloning into 'min-api-with-mongo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
vmadmin@lp-22-04:~/Documents$ cd min-api-with-mongo/
vmadmin@lp-22-04:~/Documents/min-api-with-mongo$ dotnet new web --name WebApi --framework net8.0
The template "ASP.NET Core Empty" was created successfully.

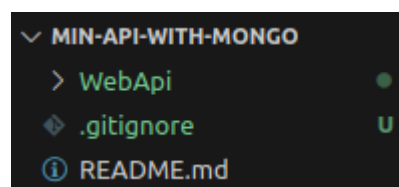
Processing post-creation actions...
Restoring /home/vmadmin/Documents/min-api-with-mongo/WebApi/WebApi.csproj:
  Determining projects to restore...
  Restored /home/vmadmin/Documents/min-api-with-mongo/WebApi/WebApi.csproj (in 371 ms).
Restore succeeded.
```

Erstellen Sie im gleichen Verzeichnis ein .gitignore. Es sorgt dafür, dass nur relevanter Source-Code ins GIT-Repository übertragen wird (ohne Binaries, etc.).

```
dotnet new gitignore
```

```
vmadmin@lp-22-04:~/Documents/min-api-with-mongo$ dotnet new gitignore
The template "dotnet gitignore file" was created successfully.
```

Öffnen Sie Visual Studio Code (VS Code) und öffnen Sie das Projektverzeichnis min-api-with-mongo (File → Open Folder). Ihre Projektstruktur müsste jetzt mit folgendem Bild übereinstimmen:

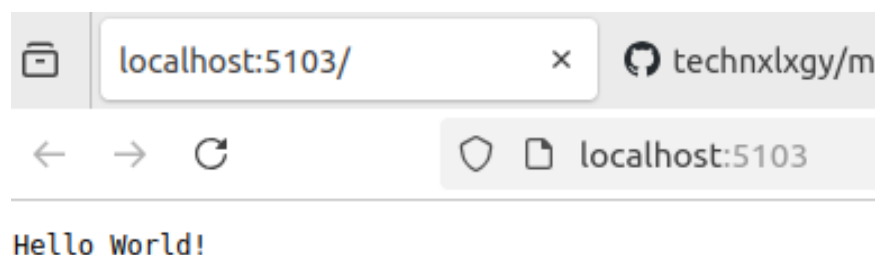


Öffnen Sie in VS Code ein Terminal (Terminal → New Terminal) Navigieren Sie in den Folder WebApi und starten Sie die Anwendung:

```
dotnet run
```

```
vmadmin@lp-22-04:~/Documents/min-api-with-mongo$ cd WebApi
vmadmin@lp-22-04:~/Documents/min-api-with-mongo/WebApi$ dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5103
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /home/vmadmin/Documents/min-api-with-mongo/WebApi
```

Öffnen Sie die in der Konsole ausgegebene URL und vergewissern Sie sich, dass im Browser Hello World angezeigt wird.



In der Datei Properties/launchSettings.json ist definiert, wie die Anwendung gestartet wird. Per Default sind verschiedene Profile hinterlegt. Passen Sie das http-Profil so an, dass die Anwendung über http://localhost:5001 (http-Profil) erreichbar ist. Löschen Sie die nicht benötigte Section iisSettings und die beiden Profile https und IIS Express und starten Sie die Anwendung neu. Sie müsste nun über http://localhost:5001 erreichbar sein.

```
{ launchSettings.json U x
WebApi > Properties > {} launchSettings.json > ...
1 {
2   "$schema": "http://json.schemastore.org/launchsettings.json",
3   "profiles": {
4     "http": {
5       "commandName": "Project",
6       "dotnetRunMessages": true,
7       "launchBrowser": true,
8       "applicationUrl": "http://localhost:5001",
9       "environmentVariables": {
10        "ASPNETCORE_ENVIRONMENT": "Development"
11      }
12    }
13  }
14 }
15
```

alter Port:



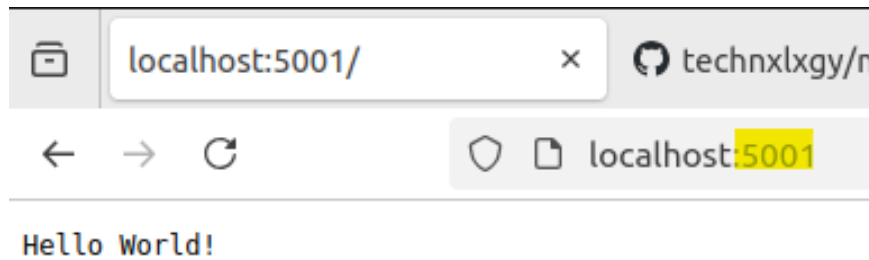
Unable to connect

Firefox can't establish a connection to the server at localhost:5103.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

Try Again

neuer Port:



4. Dockerfile

Erstellen Sie im Verzeichnis WebApi ein Dockerfile für ein Multistage Image. Die Anwendung soll mit

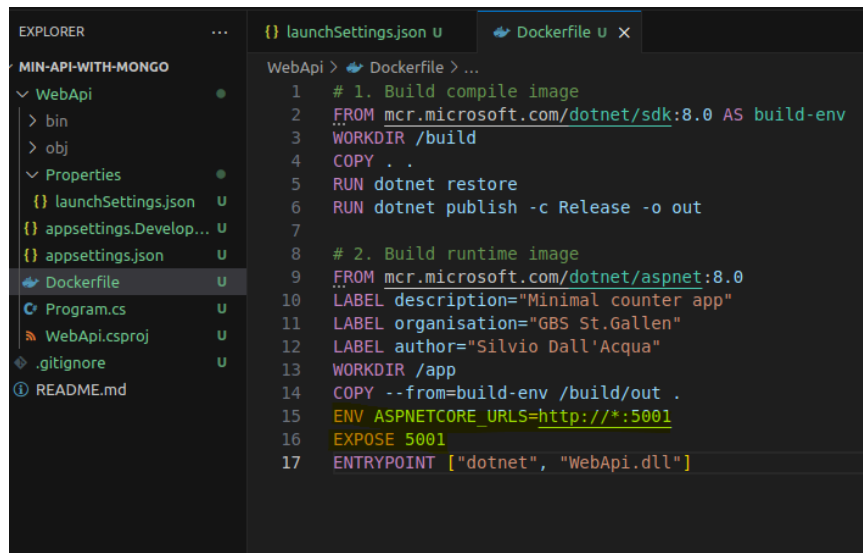
- Image mcr.microsoft.com/dotnet/sdk:8.0 restored und published werden
- Image mcr.microsoft.com/dotnet/aspnet:8.0 ausgeführt werden.

Die Anwendung soll wie bei lokaler Ausführung auch über `http://localhost:5001` erreichbar sein.

Dockerfile:

```
# 1. Build compile image
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build-env
WORKDIR /build
COPY . .
RUN dotnet restore
RUN dotnet publish -c Release -o out

# 2. Build runtime image
FROM mcr.microsoft.com/dotnet/aspnet:8.0
LABEL description="Minimal counter app"
LABEL organisation="GBS St.Gallen"
LABEL author="Silvio Dall'Acqua"
WORKDIR /app
COPY --from=build-env /build/out .
ENV ASPNETCORE_URLS=http://*:5001
EXPOSE 5001
ENTRYPOINT ["dotnet", "WebApi.dll"]
```



Das gelb markierte ist neu hinzugekommen (spezifisch für Webanwendungen)

```
docker build -t min-api-image .
```

```

vmadmin@lp-22-04:~/Documents/min-api-with-mongo/WebApi$ docker build -t min-api-image .
[+] Building 53.6s (14/14) FINISHED
docker:default

```

```
docker run --name min-api-container -p 5001:5001 min-api-image
```

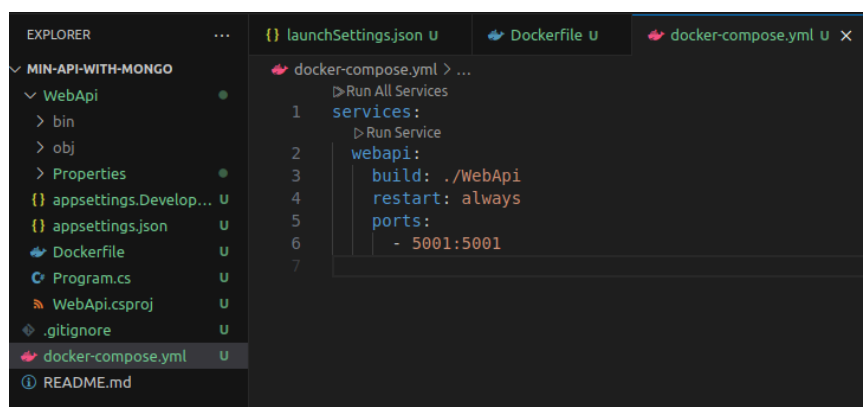
```

vmadmin@lp-22-04:~/Documents/min-api-with-mongo/WebApi$ docker run --name min-api-container -p 5001:5001 min-api-image

```

5. Docker-compose.yml

Erstellen Sie direkt im übergeordneten Projektverzeichnis min-api-with-mongo ein docker-compose.yml. Mit docker compose up soll die Anwendung mit Hilfe des Dockerfiles erzeugt und gestartet werden.



```
docker compose up --build
```

```
^Cvadmin@lp-22-04:~/Documents/min-api-with-mongo/WebApi$ cd ..  
vadmin@lp-22-04:~/Documents/min-api-with-mongo$ touch docker-compose.yml  
vadmin@lp-22-04:~/Documents/min-api-with-mongo$ docker compose up --build  
Compose can now delegate builds to bake for better performance.  
To do so, set COMPOSE_BAKE=true.  
[+] Building 0.8s (15/15) FINISHED  
=> [webapi internal] load build definition from Dockerfile
```

6. Commit und Push

Committen Sie Ihre Änderungen und Pushen Sie Ihren ersten Projektstand in Ihr Git-Repo. VS Code unterstützt Sie dabei mit der integrierten Source Control. Falls User und Email in Ihrem GIT-Client noch nicht gesetzt sind, machen Sie das wie folgt:

```
git config --global user.email "you@example.com"  
git config --global user.name "Your Name"
```

Hinweis Pushen Sie Ihre Änderungen nach jedem Schritt in Ihr GIT-Repo. So können Sie bei Bedarf jederzeit auf einem definierten Stand aufsetzen.

```
git add .  
git commit -m "commit message"  
git push
```

