1 Grundgerüst eines Minimal API

1.1 Ziele

- Sie erstellen das Grundgerüst eines .NET8 Minimal API
- Sie erstellen ein docker-compose.yml und Dockerfile, um die Anwendung zu erstellen und auszuführen

1.2 Umgebung

Die Übung wird auf der VM LP-22.04 durchgeführt. Als IDE wird Visual Studio Code verwendet.

1.3 Aufgaben

Aufgabe 1: Installation .NET 8 | Einzelarbeit | 10'

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
```

Aufgabe 2: GIT-Repository erstellen | Einzelarbeit | 10'

Erstellen Sie ein leeres GIT-Repository *min-api-with-mongo* für Ihr Projekt.

Fügen Sie dem Projekt ein *README.md* hinzu.

Aufgabe 3: Grundgerüst erstellen | Einzelarbeit | 10'

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

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

Erstellen Sie ein .NET Projekt *WebApi* mit Template *web*:

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

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
```

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:

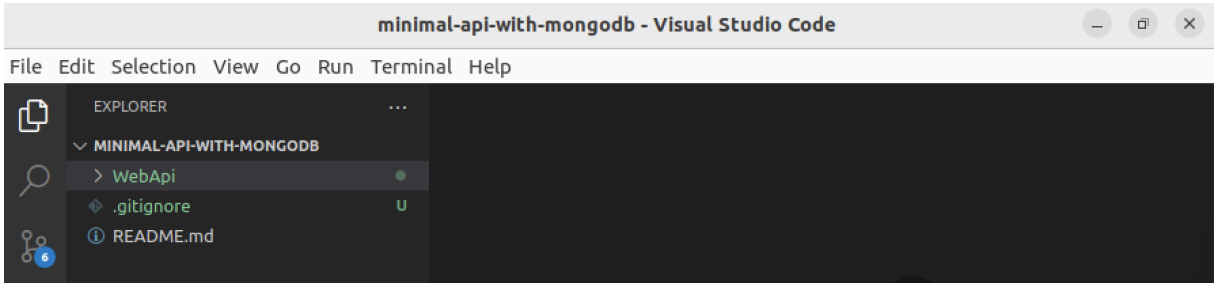


Abbildung 1: Projektstruktur

Öffnen Sie in VS Code ein Terminal (Terminal -> New Terminal)

Navigieren Sie in den Folder *WebApi* und starten Sie die Anwendung:

```
dotnet run
```

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

Mit [ctrl] c beenden Sie die Anwendung.

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.

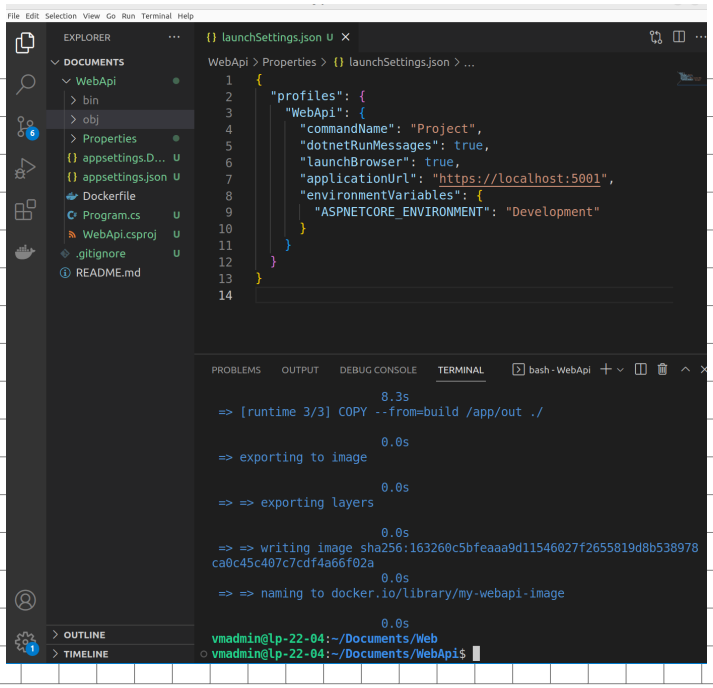
Löschen Sie die nicht benötigte Section *iisSettings* und die beiden Profile *https* und *IIS Express* und starten Sie die Anwendung neu. Sie müsste nun über http://localhost:5001 erreichbar sein.

Aufgabe 4: Dockerfile | Einzelarbeit | 20’

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.

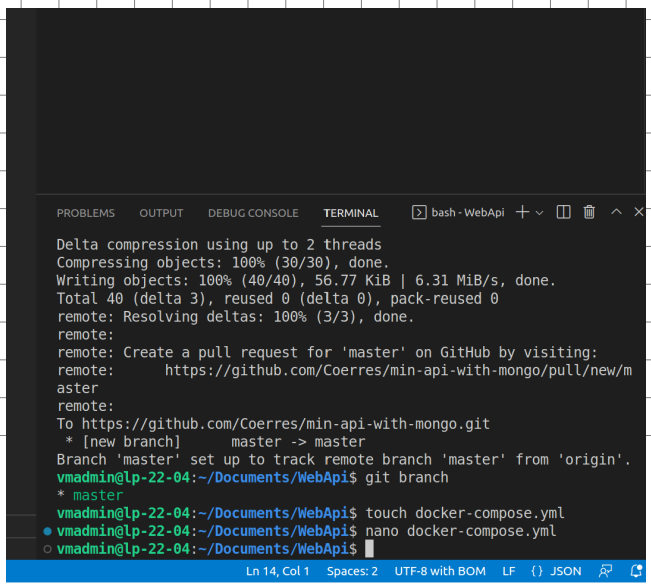
Die Anwendung soll wie bei lokaler Ausführung auch über http://localhost:5001 erreichbar sein.



Aufgabe 5: docker-compose.yml | Einzelarbeit | 10'

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.



Aufgabe 6: Commit und Push | Einzelarbeit | 10’

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.