

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:

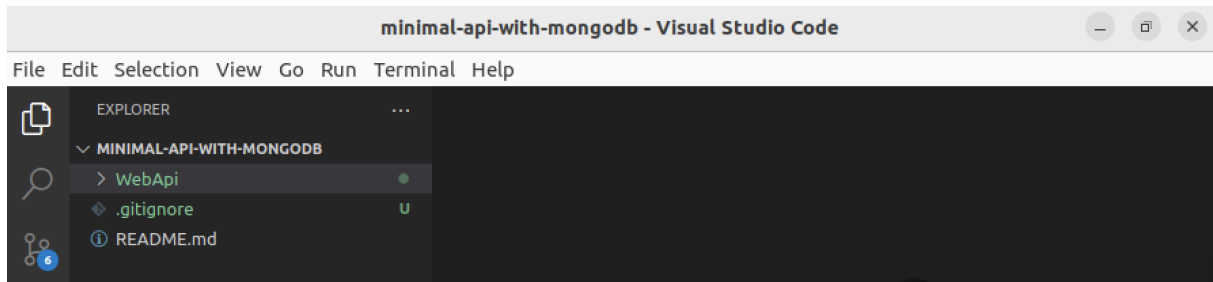


Abbildung 1: Projektstruktur

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 | Einzelerbeit | 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.

The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows the project structure with files like bin, obj, Properties, appsettings.D..., appsettings.json, Dockerfile, Program.cs, WebApi.csproj, .gitignore, and README.md. The main editor shows the launchSettings.json file with the following content:

```
1 {
2   "profiles": {
3     "WebApi": {
4       "commandName": "Project",
5       "dotnetRunMessages": true,
6       "launchBrowser": true,
7       "applicationUrl": "https://localhost:5001",
8       "environmentVariables": {
9         "ASPNETCORE_ENVIRONMENT": "Development"
10      }
11    }
12  }
13 }
14
```

The TERMINAL pane at the bottom shows the output of the Docker build process:

```
8.3s
=> [runtime 3/3] COPY --from=build /app/out ./
0.0s
=> exporting to image
0.0s
=> => exporting layers
0.0s
=> => writing image sha256:163260c5bfeaaa9d11546027f2655819d8b538978
ca0c45c407c7cdf4a66f02a
0.0s
=> => naming to docker.io/library/my-webapi-image
0.0s
vmadmin@lp-22-04:~/Documents/Web
o vmadmin@lp-22-04:~/Documents/WebApi$
```

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.

The screenshot shows the terminal output of the Docker build process and the creation of a new branch 'master':

```
Delta compression using up to 2 threads
Compressing objects: 100% (30/30), done.
Writing objects: 100% (40/40), 56.77 KiB | 6.31 MiB/s, done.
Total 40 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/Coerres/min-api-with-mongo/pull/new/master
remote:
remote: To https://github.com/Coerres/min-api-with-mongo.git
* [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
vmadmin@lp-22-04:~/Documents/WebApi$ git branch
* master
vmadmin@lp-22-04:~/Documents/WebApi$ touch docker-compose.yml
vmadmin@lp-22-04:~/Documents/WebApi$ nano docker-compose.yml
o vmadmin@lp-22-04:~/Documents/WebApi$
```

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.