

1 Einbindung MongoDB

Hinweis: Dieser Auftrag setzt auf erledigtem Auftrag 01-AA-Minimal-API-MongoDB auf.

1.1 Ziele

- Sie starten eine MongoDB als Docker-Container
- Sie binden eine MongoDB in eine .NET-Anwendung ein
- Sie orchestrieren die Anwendung (API und MongoDB) mit docker-compose

1.2 Umgebung

Die Übung wird auf der VM LP-22.04 durchgeführt.

1.3 Aufgaben

Aufgabe 1: MongoDB-Container |  Einzelarbeit |  10'

Starten Sie einen MongoDB-Container, der folgende Anforderungen erfüllt:

- Ausführung im Hintergrund
- Das Datenverzeichnis ist in ein named Volume zu mounten.
- Der Port der Datenbank ist 1:1 zu mappen.

```
vmadmin@lp-22-04:~/M347/M347_Containers/min-api-with-mongo$ docker run -d -v mongovolume:/data/db --name mymongocontainer -p 27017:27017 -e MONGO_INITDB_ROOT_USERNAME=root -e MONGO_INITDB_ROOT_PASSWORD=secret mongo
126e5c0eb59c4fc237351c50aa3e5bdfec0f63869e427621bc7b5eb9d14a8c55
vmadmin@lp-22-04:~/M347/M347_Containers/min-api-with-mongo$
```

Vergewissern Sie sich, dass der Container läuft.

Aufgabe 2: Installation VS Code-Extension für C sharp |  Einzelarbeit |  5'

Um VS-Code mit *Syntax Highlighting*, *IntelliSense*, etc. zu erweitern, installieren Sie die Extension C#.

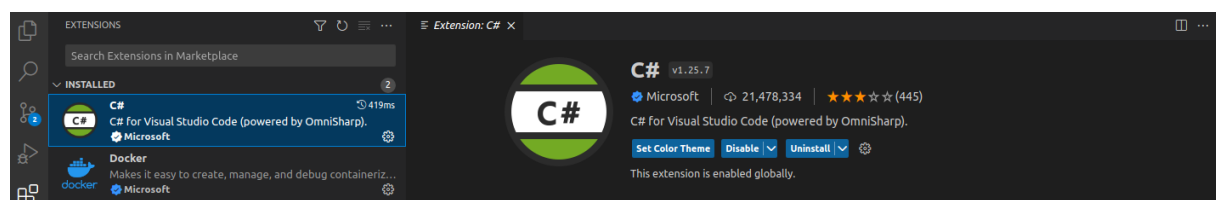


Abbildung 1: C#-Extension für VS code

Aufgabe 3: MongoDB-Container | Einzelerbeit | 15'

Für den Zugriff auf eine MongoDB existiert ein offizieller .NET MongoDB.Driver.

Installieren Sie das Nuget-Package mit folgendem .NET CLI Command:

```
dotnet add package MongoDB.Driver
```

Hinweis: Achten Sie darauf, dass Sie sich im Projektverzeichnis der .NET-Anwendung (min-api-with-mongo/WebApi) befinden.

Erzeugen Sie in *WebApi/Program.cs* unter *app.MapGet(...)* einen weiteren Endpunkt *check*:

```
app.MapGet("/check", () => {  
    /* Code zur Prüfung der DB ... */  
    return "Zugriff auf MongoDB ok.";  
});
```

Den vom Template eingefügten Root-Endpunkt / können Sie mit einer beliebigen Meldung anpassen.
z.B. *Minimal API Version 1.0*

Der Aufruf von <http://localhost:5001/check> soll zeigen, ob der Zugriff auf die MongoDB funktioniert.

Erweitern Sie die eben eingefügte Methode so, dass folgende Anforderungen erfüllt werden

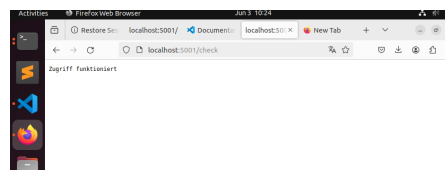
- Die Verbindung zur MongoDB wird über *MongoDB.Driver.MongoClient* aufgebaut.
- die vorhandenen Datenbanken werden abgefragt und in der Antwort ausgegeben.
- Exceptions sind mit try/catch abgefangen und werden als Fehlermeldung zurückgegeben.

Hinweis: Der Connections-String darf fix programmiert werden. Wir werden ihn später konfigurierbar machen. Der Aufbau des Connection-Strings ist in der MongoDB-Dokumentation beschrieben.

Starten sie die Anwendung und rufen Sie im Browser <http://localhost:5001/check> auf. Folgendes Ergebnis wird erwartet:



Abbildung 2: Ergebnis nach Aufruf von /check



Aufgabe 4: Konfiguration des Connection-String | Einzelarbeit | 15'

Durch Umsetzung des Options Patterns soll der Connection-String in appsettings.json konfiguriert werden können.

Erstellen Sie unter *min-api-with-mongo/WebApi* ein neues File *DatabaseSettings.cs* mit folgendem Inhalt:

```
public class DatabaseSettings
{
    public string ConnectionString { get; set; } = "";
}
```

Erweitern Sie *min-api-with-mongo/WebApi/appsettings.json* um den Abschnitt *DatabaseSettings* und weisen Sie *ConnectionString* den bis jetzt fix codierten Wert zu:

```
"AllowedHosts": "*",

"DatabaseSettings": {
  "ConnectionString": "mongodb://gbs:geheim@localhost:27017"
```

Erweitern Sie *min-api-with-mongo/WebApi/Program.cs*, um die *DatabaseSettings* als Service für DependencyInjection zu registrieren. Fügen Sie dazu nach *var builder = WebApplication.CreateBuilder(args);* folgende zwei Codezeilen ein:

```
var movieDatabaseConfigSection =
    ↪ builder.Configuration.GetSection("DatabaseSettings");
builder.Services.Configure<DatabaseSettings>(movieDatabaseConfigSection);
```

Um die *DatabaseSettings* zu injecten, ersetzen Sie *pp.MapGet("/check", () => {* mit

```
pp.MapGet("/check",
    ↪ (Microsoft.Extensions.Options.IOptions<DatabaseSettings> options) => {
```

Über *options.Value* haben Sie Zugriff auf alle Werte von *DatabaseSettings*. Ersetzen Sie die fixe Zuweisung wie folgt:

```
var mongoDbConnectionString = options.Value.ConnectionString;
```

Starten Sie die Anwendung und vergewissern Sie sich, dass sie den *mongoDbConnectionString* aus *appsettings.json* verwendet.

Aufgabe 5: docker-compose erweitern | Einzelarbeit | 15'

Erweitern Sie *min-api-with-mongo/docker-compose.yml*, dass nebst dem *WebApi* auch ein *MongoDB*-Container gestartet wird. Beachten Sie, dass das API erst gestartet werden soll, wenn die *MongoDB* verfügbar ist.

Hinweise: In ASP.NET Core können Settings aus appsetting.json mit Hilfe von Umgebungsvariablen übersteuert werden. Setting `MoviesDatabaseSettings.ConnectionString` wird mit Umgebungsvariable `MoviesDatabaseSettings_ConnectionString` übersteuert. (Ein Punkt im Pfad wird durch zwei ersetzt) Das ermöglicht Ihnen, den Connection-String der MongoDB per Umgebungsvariable zu setzen.

Dem Service *webapi* weisen Sie demzufolge den `ConnectionString` wie folgt zu:

environment:

`MoviesDatabaseSettings__ConnectionString:`

↪ `"mongodb://gbs:geheim@mongodb:27017"`