Einbindung MongoDB

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

MongoDB-Container

Starten Sie einen MongoDB-Containter, 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.

```
docker run -d --name mongodb -p 27017:27017 -v mongodb:/data/db mongo
```

Installation VS Code-Extension für C#

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

MongoDB-Container

Für den Zugriff auf eine MongoDB exisitert ein offizieller **.NET MongoDB.Driver**.

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

```
cd WebApi
dotnet add package MongoDB.Driver
```

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

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

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.

```
using MongoDB.Driver;
using System.Linq;

var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

app.MapGet("/check", () => {
   try
   {
      var mongoClient = new MongoClient("mongodb://localhost:27017");
      var dbs = mongoClient.ListDatabaseNames().ToList();

      string message = $"Zugriff auf MongoDB ok. Vorhandene DBs: " + String.Join(", ", dbs);
      return Results.Ok(message);
   }
   catch (Exception ex)
   {
      return Results.Problem("Leider hat Zugriff auf MongoDB nicht geklappt.");
   }
});

app.Run();
```

oder mit string als output

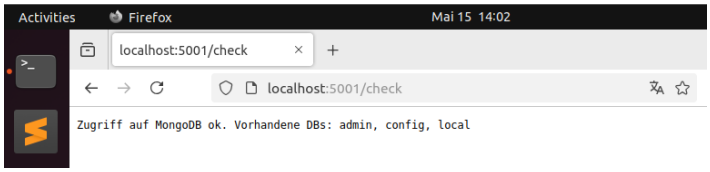
```
using MongoDB.Driver;
using System.Linq;

var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

app.MapGet("/check", () => {
   try
   {
      var mongoClient = new MongoClient("mongodb://localhost:270179");
      var dbs = mongoClient.ListDatabaseNames().ToList();

      return $"Zugriff auf MongoDB ok. Vorhandene DBs: " + String.Join(", ", dbs);
   }
   catch (Exception ex)
   {
      return "Leider hat Zugriff auf MongoDB nicht geklappt.";
   }
});

app.Run();
```



Konfiguration des Connection-String

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:

```
{
 "Logging": {
   "LogLevel": {
     "Default": "Information",
     "Microsoft.AspNetCore": "Warning"
   }
 },
 "AllowedHosts": "*",
 "DatabaseSettings": {
   "ConnectionString": "mongodb://localhost:27017"
 }
}
```

Erweitern Sie min-api-with-mongo/WebApi/Program.cs, um die DatabaseSettings als Service für DependencyInjection zu registrieren:

```
using MongoDB.Driver;
using System.Linq;

var builder = WebApplication.CreateBuilder(args);

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

```
var app = builder.Build();

app.MapGet("/", () => "Minimal API Version 1.0");

app.MapGet("/check", (Microsoft.Extensions.Options.IOptions<DatabaseSettings> options) => {
   try
   {
      var mongoClient = new MongoClient(options.Value.ConnectionString);
      var dbs = mongoClient.ListDatabaseNames().ToList();

      return $"Zugriff auf MongoDB ok. Vorhandene DBs: " + String.Join(", ", dbs);
   }
   catch (Exception ex)
   {
      return "Leider hat Zugriff auf MongoDB nicht geklappt.";
   }
});

app.Run();
```

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

```
services:
  mongodb:
    image: mongo
    volumes:
      - mongodb:/data/db
    ports:
      - 27017:27017
    networks:
      - mongoapisite

  webapi:
    build: ./WebApi
    depends_on:
      - mongodb
    restart: always
    environment:
      DatabaseSettings__ConnectionString: "mongodb://mongodb:27017"
    ports:
      - 5001:5001
    networks:
      - mongoapisite

networks:
  mongoapisite:
    driver: bridge
```

```
volumes:
  mongodb:
```

```
environment:
  DatabaseSettings__ConnectionString: "mongodb://mongodb:27017"
```

wird den ConnectionString aus appsettings.json-Datei überschreiben !