1. Ziele

- Sie erstellen einen MovieService mit CRUD-Operationen für eine MongoDB
- Sie verwenden den MovieService in den Endpunkten

2. MovieService erstellen

Alle Zugriffe auf die MongoDB sollen nun in eine eigene Komponente ausgelagert werden. In ASP.NET Core werden dazu Services verwendet, die dank dependency injection in der ganzen App verfügbar sind.
Erstellen Sie dazu unter min-api-with-mongo/WebApi ein Interface IMovieService in Datei IMovieService.cs mit dem Codegerüst für die 5 Endpunkte:

```
public interface IMovieService
{
    IEnumerable<Movie> Get();                       //get all movies
    Movie Get(string id);                           //movie by Id
    void Create(Movie movie);                       //create movie
    public void Update(string id, Movie movie);     //update movie
    public void Delete(string id);                  //delete movie
}
```

Erstellen Sie im gleichen Verzeichnis ein Gerüst einer konkreten Implementierung als Klasse MongoMovieService (MongoMovieService.cs), die das Interface IMovieService implementiert.

```
using Microsoft.Extensions.Options;
using MongoDB.Driver;

public class MongoMovieService : IMovieService
{
  private readonly IMongoCollection<Movie> _movieCollection;
  private const string mongoDbDatabaseName = "gbs";
  private const string mongoDbCollectionName = "movies";

  // Constructor.  Settings werden per dependency injection übergeben.
  public MongoMovieService(IOptions<DatabaseSettings> options)
  {
    var mongoDbConnectionString = options.Value.ConnectionString;
    var mongoClient = new MongoClient(mongoDbConnectionString);
    var database = mongoClient.GetDatabase(mongoDbDatabaseName);
    _movieCollection = database.GetCollection<Movie>(mongoDbCollectionName);
  }

  public void Create(Movie movie)
  {
    throw new NotImplementedException();
  }

  public IEnumerable<Movie> Get()
  {
    throw new NotImplementedException();
  }

  public Movie Get(string id)
  {
    throw new NotImplementedException();
  }

  public void Update(string id, Movie movie)
  {
    throw new NotImplementedException();
  }

  public void Delete(string id)
  {
    throw new NotImplementedException();
  }

}
```

Im Konstruktor werden die DatabaseSettings übernommen und damit eine Referenz auf die angegebene Collection *movies* in der Datenbank *gbs* erstellt.

Damit die DatabaseSettings via Dependeny Injection übergeben werden, muss die Klasse in Programm.cs als Singleton registriert werden:

```
...
builder.Services.AddSingleton<IMovieService, MongoMovieService>();
...
```

3. MovieService verwenden

Der Movieservice kann nun in den 5 Endpunkten aufgerufen werden, hier der Endpunkt für get movie by id:

```
// Get Movie by id
app.MapGet("api/movies/{id}", (IMovieService movieService, string id) =>
{
    var movie = movieService.Get(id);
    return movie != null
        ? Results.Ok(movie)
        : Results.NotFound();
});
```

Wie Sie sehen wird einfach die Get-Methode des MovieService aufgerufen un in Abhänigkeit des Resultates der Movie zurückgegeben oder NotFound.

Her der Endpunkt des POST-Endpunktes:

```
// Insert Movie
app.MapPost("/api/movies", (IMovieService movieService, Movie movie) =>
{
    movieService.Create(movie);
    return Results.Ok(movie);
});
```

Nun fehlt noch die Implementierung der Get-Methode:

```
  public Movie Get(string id)
  {
    return _movieCollection.Find(m => m.Id == id).FirstOrDefault();
  }
```

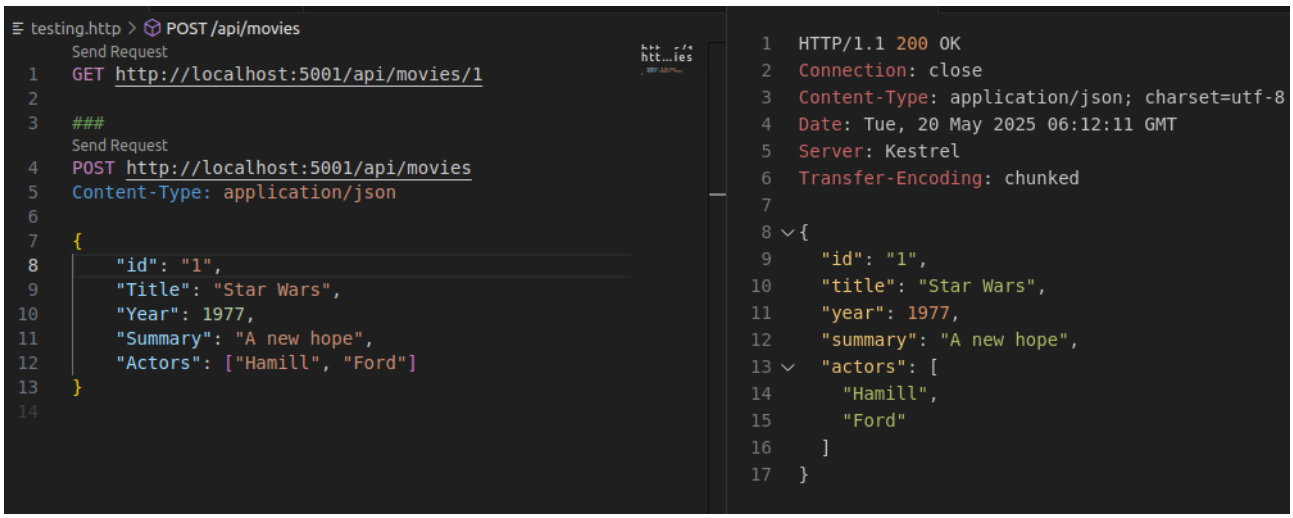
Die Create-Methode lässt sich so implementieren:

```
  public void Create(Movie movie)
  {
    _movieCollection.InsertOne(movie);
  }
```

Vervollständigen Sie nun die restlichen Methoden und Endpunkte. Als Hilfsmittel können Sie z.B. diese Referenz verwenden: https://www.mongodb.com/docs/drivers/csharp/current/quick-reference/

4. Testen mit REST-Client

Die Datenbank und die Collection enthalten zu Beginn keine Filme. Filme können Sie mit einem POST-Request mit Hilfe des REST-Clients hinzufügen. Sie können die bereits zuvor erstellte Datei testing.http verwenden. Fügen Sie den POST-Request nach drei Hashtags ### hinzu.
Beachten Sie, dass Sie im Header des Request den Content-Type: application/json angeben müssen. Der Body mit den JSON-Daten muss durch eine zusätzliche Zeile abgetrennt sein. Die Struktur der JSON-Daten muss der Struktur der Movie-Klasse entsprechen. Weitere Requests können Sie nach weiteren Hashtags hinzufügen.



Testen Sie alle 5 Endpunkte

5. Hilfsmittel

https://gbssg.gitlab.io/m347/
Internet

6. Erwartete Resultate

Mit Screenshots dokumentiertes und kommentiertes Vorgehen (Word) Zeit: 30 Minuten