

**TUGAS JURNAL
KONSTRUKSI PERANGKAT LUNAK**

**MODUL IX
API DESIGN & CONSTRUCTION USING SWAGGER**



**Disusun Oleh :
Atika Aji Hadiyani
2211104003
SE06-01**

**Dosen Pengampu :
Yudha Islami Sulistya, S.Kom., M.Cs.**

**PROGRAM STUDI S1 SOFTWARE ENGINEERING FAKULTAS
INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

TUGAS JURNAL

1. MEMBUAT PROJECT WEB API

Berhubung cara membuat project web api berbeda-beda untuk setiap bahasa pemrograman, langkah langkah berikut hanya berlaku apabila dilakukan dengan menggunakan .NET dan Visual Studio. Untuk IDE dan bahasa pemrograman lain, yang terpenting adalah nama project yang dibuat yaitu “modul8_NIM”.

- A. Buka visual studio yang sudah terinstall dengan ASP.NET dan .NET 5.0 SDK atau setelahnya
- B. Pilih New Project dan kemudian pilih ASP.NET Core Web API atau API (pastikan opsi ‘Enable OpenAPI support’ tercentang).
- C. Pastikan untuk memilih .NET versi 5.0 atau yang lebih baru.
- D. Masukkan nama projek “modul9_NIM”.
- E. Langkah-langkah yang disertai gambar dapat dilihat pada link berikut ini (cukup dilihat pada bagian “Create a Web API project”):
<https://docs.microsoft.com/en-us/aspnet/core/tutorials/min-web-api?view=aspnetcore-6.0&tabs=visual-studio>
- F. Setelah project tersebut selesai dibuat, coba run programnya, dan tunggu sampai program selesai di-compile.

Hasil:

Additional information

ASP.NET Core Web API

C#LinuxmacOSWindowsAPICloudService

Framework ⓘ

.NET 8.0 (Long Term Support)

Authentication type ⓘ

None

☒ Configure for HTTPS ⓘ☐ Enable container support ⓘ

Container OS ⓘ

Linux

Container build type ⓘ

Dockerfile

☒ Enable OpenAPI support ⓘ☐ Do not use top-level statements ⓘ☒ Use controllers ⓘ☐ Enlist in .NET Aspire orchestration ⓘ

Aspire version ⓘ

9.0

Tampilan run program:

Swagger

powered by SWAGGER

Select a definitionmodul9_2211104003 v1

modul9_2211104003

1.0OAS 3.0

<https://localhost:7105/swagger/v1/swagger.json>

WeatherForecast

GET /WeatherForecast

Schemas

WeatherForecast >

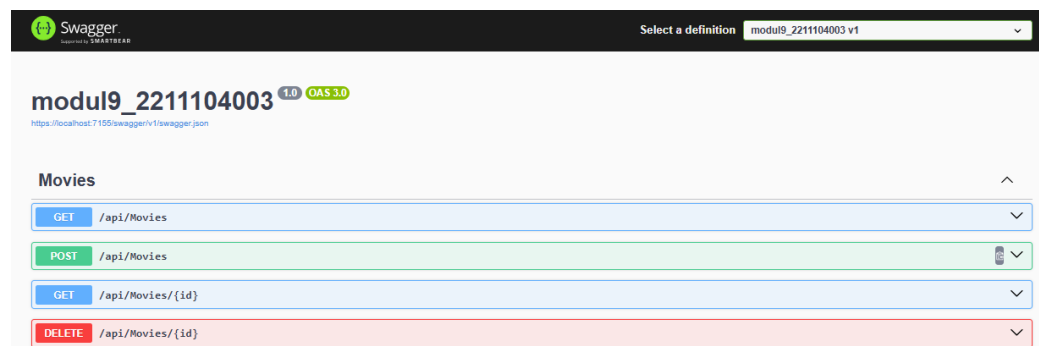
2. IMPLEMENTASI WEB API

Dari master/main branch dan class utama, buatlah program/aplikasi web API dari spesifikasi sebagai berikut ini:

A. API yang dibuat menggunakan data dari kelas Movie.

Movie
+ Title : string
+ Director : string
+ Stars : List<string>
+ Description: string
+ Movie()

B. API yang dibuat mempunyai lokasi sebagai berikut **‘/api/Movies**, URL domain boleh dari port mana saja (port bebas). Dengan menggunakan swagger API tersebut dapat menerima RESTful API dengan metoda sebagai berikut (halaman swagger dapat diakses pada <https://localhost:7150/swagger/index.html>):



- GET /api/Movies: mengembalikan output berupa list/array dari semua objek Movies
- GET /api/Movies/{id}: mengembalikan output berupa objek Movie untuk index “id”
- POST /api/Movies: menambahkan objek Movie baru
- DELETE /api/Movies/{id}: menghapus objek Movie pada index “id”

C. Secara default, program yang dibuat memiliki list film yang berasal dari TOP 3 film IMDB dari link: https://www.imdb.com/search/title/?groups=top_100&sort=user_rating,desc

D. Implementasi yang dibuat tidak menggunakan database, cukup disimpan sebagai suatu variable, dan gunakan “static” di variable tersebut yang menyimpan list/array dari objek objek Movie.

E. Dalam pembuatan program/aplikasi ini, anda dapat mengasumsikan bahwa input dari user selalu benar dan sesuai dengan tipe data yang diharapkan.

Jawab:

- **Source code**

Movie.cs

```
using System.Collections.Generic;

namespace modul9_2211104003
{
    11 references
    public class Movie
    {
        3 references
        public string Title { get; set; }
        3 references
        public string Director { get; set; }
        3 references
        public List<string> Stars { get; set; }
        3 references
        public string Description { get; set; }

        3 references
        public Movie() { }
    }
}
```

MoviesController.cs

```
using Microsoft.AspNetCore.Mvc;
using modul9_2211104003;
using System.Collections.Generic;

namespace modul9_2211104017.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    0 references
    public class MoviesController : ControllerBase
    {
        private static List<Movie> movies = new List<Movie>
        {
            new Movie {
                Title = "The Shawshank Redemption",
                Director = "Frank Darabont",
                Stars = new List<string> { "Tim Robbins", "Morgan Freeman", "Bob Gunton" },
                Description = "Two imprisoned men bond over a number of years..."
            },
            new Movie {
                Title = "The Godfather",
                Director = "Francis Ford Coppola",
                Stars = new List<string> { "Marlon Brando", "Al Pacino", "James Caan" },
                Description = "The aging patriarch of an organized crime dynasty..."
            },
            new Movie {
                Title = "The Dark Knight",
                Director = "Christopher Nolan",
                Stars = new List<string> { "Christian Bale", "Heath Ledger", "Aaron Eckhart" },
                Description = "When the menace known as the Joker wreaks havoc..."
            }
        };
    }
}
```

```

// GET: api/Movies
[HttpGet]
0 references
public ActionResult<List<Movie>> Get() => movies;

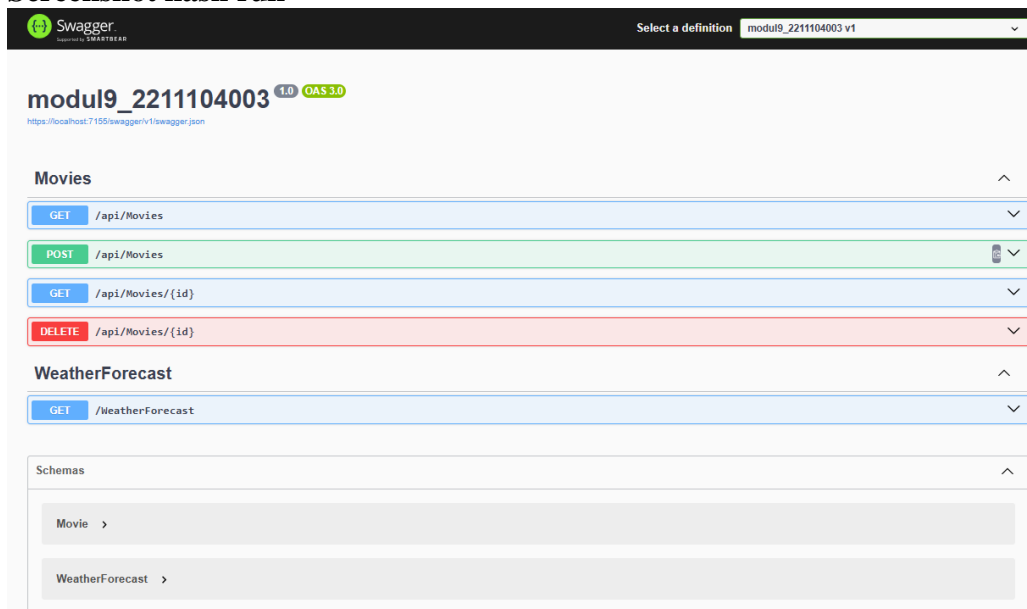
// GET: api/Movies/{id}
[HttpGet("{id}")]
0 references
public ActionResult<Movie> Get(int id)
{
    if (id < 0 || id >= movies.Count)
        return NotFound();
    return movies[id];
}

// POST: api/Movies
[HttpPost]
0 references
public ActionResult<List<Movie>> Post([FromBody] Movie newMovie)
{
    movies.Add(newMovie);
    return movies;
}

// DELETE: api/Movies/{id}
[HttpDelete("{id}")]
0 references
public ActionResult<List<Movie>> Delete(int id)
{
    if (id < 0 || id >= movies.Count)
        return NotFound();
    movies.RemoveAt(id);
    return movies;
}
}

```

- Screenshot hasil run



- Penjelasan

Aplikasi ini merupakan Web API sederhana yang dibangun menggunakan ASP.NET Core dan berfungsi untuk mengelola data film. Data film disimpan dalam sebuah list statis bernama movieList, yang berisi tiga film terpopuler dari daftar IMDB. API ini menggunakan MoviesController sebagai pengendali untuk memproses berbagai permintaan pengguna melalui sejumlah endpoint seperti GET, POST, dan DELETE.

Endpoint GET /api/Movies digunakan untuk menampilkan seluruh koleksi film, sementara GET /api/Movies/{id} akan menampilkan detail film berdasarkan indeks yang diberikan. Untuk menambahkan film baru ke daftar, digunakan endpoint POST /api/Movies, sedangkan DELETE /api/Movies/{id} berfungsi untuk menghapus film sesuai dengan indeksnya.

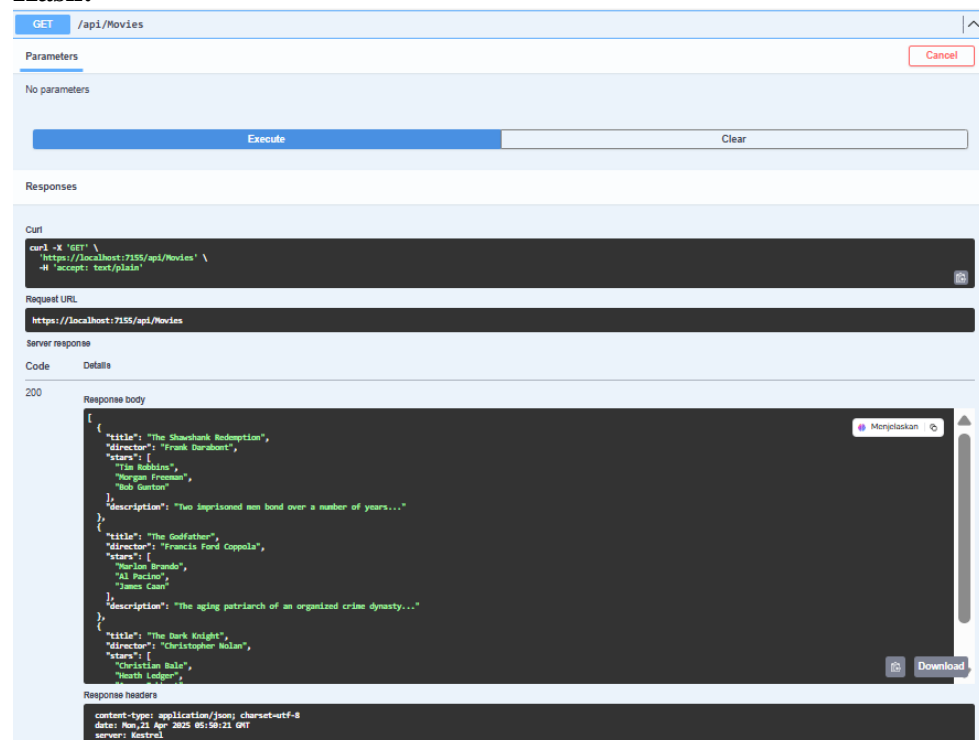
Seluruh data film hanya disimpan dalam memori melalui list statis, sehingga akan terhapus dan kembali ke kondisi awal setiap kali aplikasi dimatikan atau dijalankan ulang. Setiap method diatur untuk menangani jenis permintaan HTTP tertentu menggunakan atribut seperti [HttpGet], [HttpPost], dan [HttpDelete].

3. MENDEMONSTRASI WEB API

Beberapa skenario yang harus dicoba untuk memastikan jika program telah berjalan dengan baik. Buatlah dokumen yang berisi semua screenshot dari hasil uji coba skenario yang disebutkan pada list berikut ini:

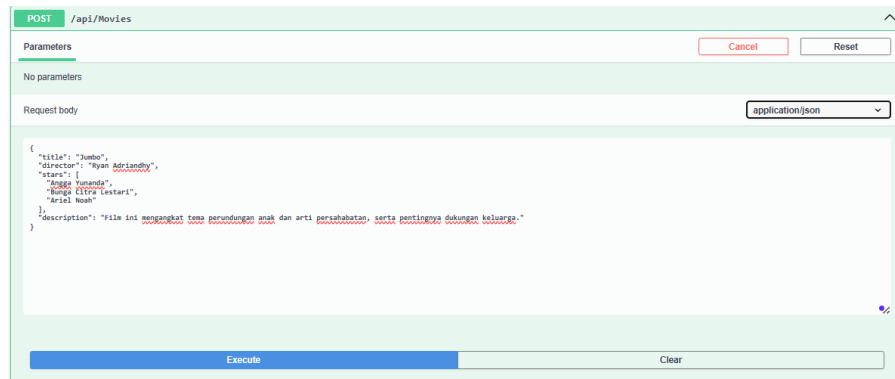
- A. Mencoba “GET /api/Movies” saat baru dijalankan yang mengeluarkan list film dari TOP 3 IMDB seperti pada tampilan berikut pada saat dicoba dengan menekan tombol “Try it out” dan tombol “Execute”

Hasil:



- B. Menambahkan Movie baru yaitu urutan ke-4 pada TOP IMDB list dengan memanggil API pada bagian “POST /api/Movies”

Hasil:



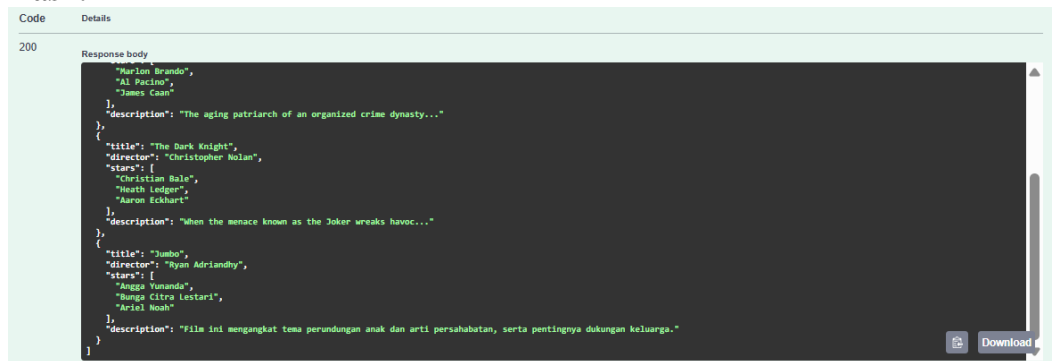
The screenshot shows a REST client interface for a POST request to the endpoint `/api/Movies`. The request body is a JSON object representing a new movie entry:

```
{  "title": "Jumbo",  "director": "Ryan Adriandhy",  "stars": [    "Angga Yunanda",    "Bunga Citra Lestari",    "Ariel Noah"  ],  "description": "Film ini mengangkat tema perundungan anak dan arti persahabatan, serta pentingnya dukungan keluarga."}
```

The interface includes buttons for "Execute" and "Clear", and a "Request body" dropdown menu set to "application/json".

- C. Cek list/array dari semua Movie lagi dengan “GET /api/Movies”, pastikan Movie yang baru ditambahkan sebelumnya sudah ada:

Hasil:



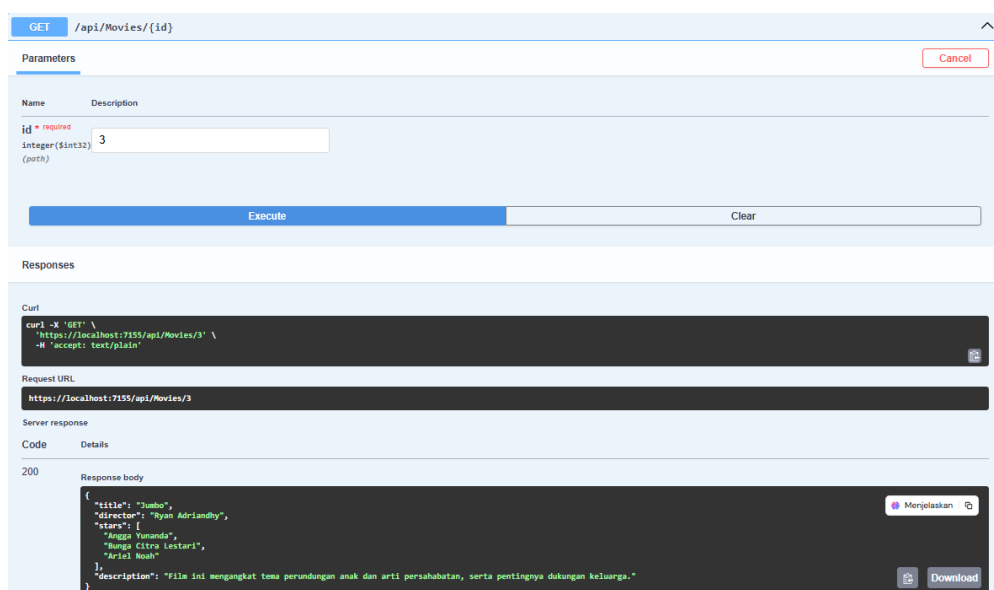
The screenshot shows the response of a GET request to the endpoint `/api/Movies`. The response is a JSON array containing three movie entries, with the newly added movie at the end:

```
[  {    "title": "The Godfather",    "director": "Francis Ford Coppola",    "stars": [      "Marlon Brando",      "Al Pacino",      "James Caan"    ],    "description": "The aging patriarch of an organized crime dynasty..."  },  {    "title": "The Dark Knight",    "director": "Christopher Nolan",    "stars": [      "Christian Bale",      "Heath Ledger",      "Aaron Eckhart"    ],    "description": "When the menace known as the Joker wreaks havoc..."  },  {    "title": "Jumbo",    "director": "Ryan Adriandhy",    "stars": [      "Angga Yunanda",      "Bunga Citra Lestari",      "Ariel Noah"    ],    "description": "Film ini mengangkat tema perundungan anak dan arti persahabatan, serta pentingnya dukungan keluarga."  }]
```

The interface includes a "Download" button for the response.

- D. Mencoba meminta Movie dengan index 3, “GET /api/Movies/3” yang seharusnya mengembalikan Movie yang baru saja ditambah:

Hasil:



The screenshot shows the response of a GET request to the endpoint `/api/Movies/{id}` with the parameter `id` set to `3`. The response is a JSON object representing the movie with index 3:

```
{  "title": "Jumbo",  "director": "Ryan Adriandhy",  "stars": [    "Angga Yunanda",    "Bunga Citra Lestari",    "Ariel Noah"  ],  "description": "Film ini mengangkat tema perundungan anak dan arti persahabatan, serta pentingnya dukungan keluarga."}
```

The interface includes a "Download" button for the response.

E. Menghapus objek Movie dengan index ke-1 dengan “DELETE /api/Movies/1”

Hasil:

DELETE /api/Movies/{id}

Parameters

Name	Description
id * required	Integer(\$int32) (path)

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  https://localhost:7155/api/Movies/1 \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7155/api/Movies/1

Server response

Code Details

200

Response body

```
{
  "title": "The Shawshank Redemption",
  "director": "Frank Darabont",
  "stars": [
    "Tim Robbins",
    "Morgan Freeman",
    "Bob Gunton"
  ],
  "description": "Two imprisoned men bond over a number of years..."
},
{
  "title": "The Dark Knight",
  "director": "Christopher Nolan",
  "stars": [
    "Christian Bale",
    "Heath Ledger",
    "Aaron Eckhart"
  ],
  "description": "When the menace known as the Joker wreaks havoc..."
},
{
  "title": "Jumbo",
  "director": "Ryan Adriandhy",
  "stars": [
    "Angga Yemanda",
    "Bunga Citra Lestari",
    "Rizky Dwi Maulana"
  ],
  "description": "A story about a man who is a giant and a woman who is a giantess..."
}
```

F. Cek list/array dari semua Movie sekali lagi dengan “GET /api/Movies”, film dengan ranking kedua “Godfather” sudah tidak ada di list:

Hasil:

GET /api/Movies

Parameters

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  https://localhost:7155/api/Movies \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7155/api/Movies

Server response

Code Details

200

Response body

```
{
  "title": "The Shawshank Redemption",
  "director": "Frank Darabont",
  "stars": [
    "Tim Robbins",
    "Morgan Freeman",
    "Bob Gunton"
  ],
  "description": "Two imprisoned men bond over a number of years..."
},
{
  "title": "The Dark Knight",
  "director": "Christopher Nolan",
  "stars": [
    "Christian Bale",
    "Heath Ledger",
    "Aaron Eckhart"
  ],
  "description": "When the menace known as the Joker wreaks havoc..."
},
{
  "title": "Jumbo",
  "director": "Ryan Adriandhy",
  "stars": [
    "Angga Yemanda",
    "Bunga Citra Lestari",
    "Rizky Dwi Maulana"
  ],
  "description": "A story about a man who is a giant and a woman who is a giantess..."
}
```

- **Penjelasan**

Aplikasi ini merupakan implementasi Web API sederhana berbasis ASP.NET Core yang berfungsi untuk melakukan pengelolaan data film. Informasi film disimpan di dalam sebuah list statis bernama `movieList`, yang berisi tiga film terbaik berdasarkan peringkat di IMDB. API ini memanfaatkan controller bernama `MoviesController` untuk menangani berbagai permintaan pengguna melalui beberapa endpoint, seperti GET, POST, dan DELETE.

Endpoint GET `/api/Movies` digunakan untuk menampilkan seluruh koleksi film yang ada, sedangkan GET `/api/Movies/{id}` berguna untuk menampilkan detail film berdasarkan posisi indeksinya di dalam list. Untuk menambahkan film baru ke dalam daftar, digunakan endpoint POST `/api/Movies`, sementara endpoint DELETE `/api/Movies/{id}` berfungsi menghapus film dari daftar sesuai indeks yang ditentukan.

Semua data film hanya tersimpan di memori dalam bentuk list statis, sehingga data tersebut akan hilang dan kembali ke kondisi awal setiap kali aplikasi dihentikan atau dijalankan ulang. Setiap method dalam controller ini ditandai dengan atribut seperti `[HttpGet]`, `[HttpPost]`, dan `[HttpDelete]` yang menunjukkan jenis request HTTP yang dilayani.