

**TUGAS JURNAL
KONSTRUKSI PERANGKAT LUNAK**

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



Disusun Oleh :

Rizky Hanifa Afania

2211104017

SE-06-01

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# Linux macOS Windows API Cloud Service

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 SMARTBEAR

Select a definition modul9_2211104017 v1

modul9_2211104017

1.0 OAS 3.0

<https://localhost:7296/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:/swagger/index.html>):

| Movies | | ^ |
|--------|------------------|---|
| GET | /api/Movies | ▼ |
| POST | /api/Movies | ▼ |
| GET | /api/Movies/{id} | ▼ |
| DELETE | /api/Movies/{id} | ▼ |

- 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_2211104017
{
    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() { }
    }
}
```

Controllers/MoviesController.cs

```
using Microsoft.AspNetCore.Mvc;
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;

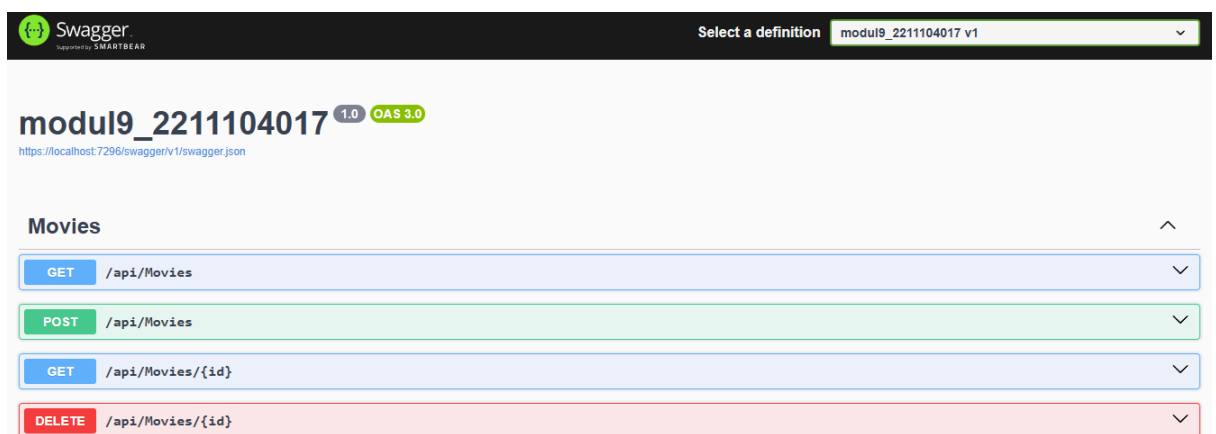
// GET: api/Movies/{id}
[HttpGet("{id}")]
0 references
public ActionResult<Movie> Get(int id)
{
    if (id < 0 || id >= movies.Count)
        return NotFound();
    return movies[id];
}

// 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

Program ini adalah Web API sederhana menggunakan ASP.NET Core yang digunakan untuk mengelola data film. Data film disimpan dalam list statis bernama

movieList, yang berisi 3 film teratas dari daftar IMDB. API ini menggunakan controller MoviesController untuk menangani permintaan dari pengguna melalui berbagai endpoint seperti GET, POST, dan DELETE. Endpoint GET /api/Movies digunakan untuk menampilkan seluruh daftar film, sedangkan GET /api/Movies/{id} menampilkan film berdasarkan indeks.

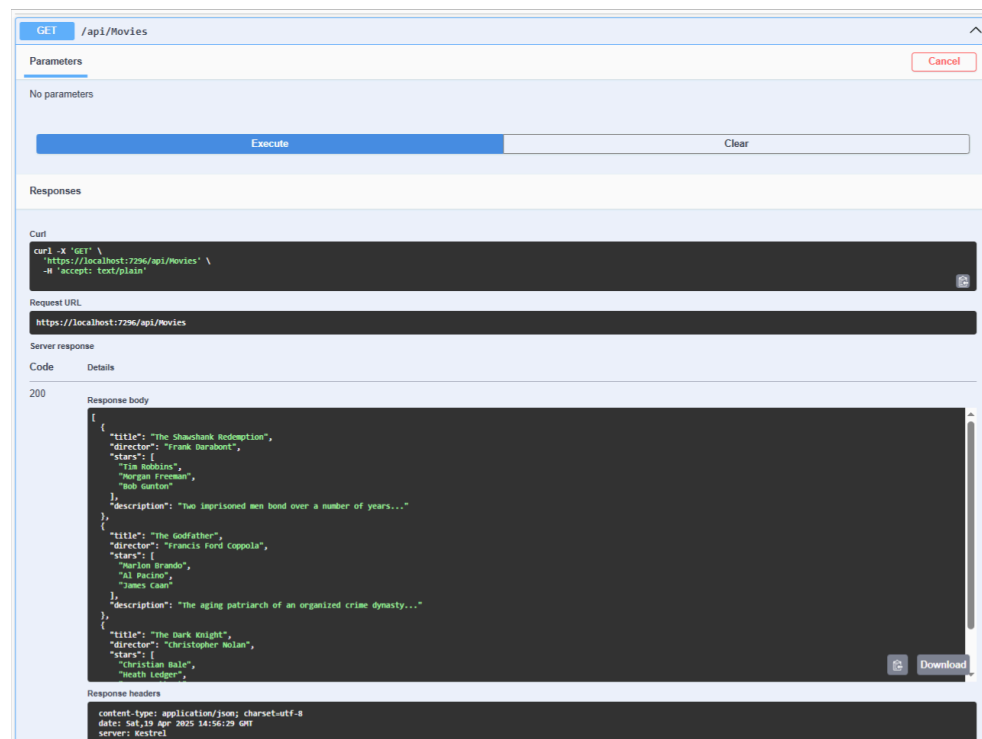
Endpoint POST /api/Movies berfungsi untuk menambahkan film baru ke dalam daftar, dan DELETE /api/Movies/{id} digunakan untuk menghapus film berdasarkan indeks. Semua data hanya disimpan di memori menggunakan list statis, sehingga akan hilang dan kembali ke kondisi awal jika aplikasi dimatikan atau di-restart. Atribut seperti [HttpGet], [HttpPost], dan [HttpDelete] digunakan untuk menentukan jenis permintaan HTTP yang ditangani oleh masing-masing method.

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:

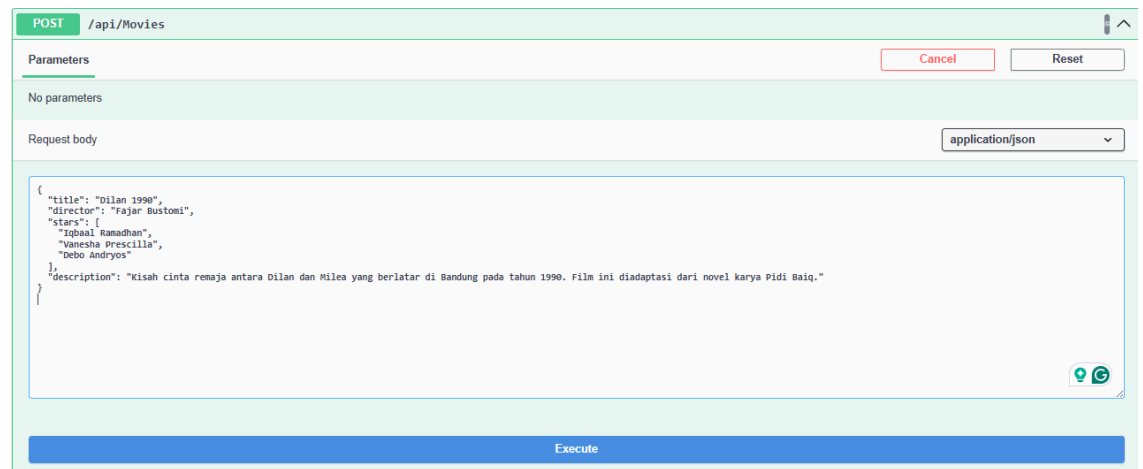
- A. Mencoba “GET /api/Movies” saat baru dijalankan yang mengeluarkan list film dari TOP 3 IMDB seperti pada tampilan berikut pada saat dicoba dengan menekan tombol “Try it out” dan tombol “Execute”

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 with a POST request to the endpoint `/api/Movies`. The request body is a JSON object representing a new movie entry. The interface includes tabs for Parameters, Request body, and Response, with buttons for Cancel, Reset, and Execute.

```
{
  "title": "Dilan 1990",
  "director": "Fajar Bustomi",
  "stars": [
    "Iqbaal Ramadhan",
    "Vanesha Prescilla",
    "Debo Andryos"
  ],
  "description": "Kisah cinta remaja antara Dilan dan Milea yang berlatar di Bandung pada tahun 1990. Film ini diadaptasi dari novel karya Pidi Baiq."
}
```

- 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 server response for a GET request to the endpoint `/api/Movies`. The response status is 200, and the response body is a JSON array containing three movie entries. The interface includes tabs for Code, Details, and Response body, with a button for Download headers.

```
[
  {
    "title": "Marlon Brando",
    "director": "Al Pacino",
    "stars": [
      "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": "Dilan 1990",
    "director": "Fajar Bustomi",
    "stars": [
      "Iqbaal Ramadhan",
      "Vanesha Prescilla",
      "Debo Andryos"
    ],
    "description": "Kisah cinta remaja antara Dilan dan Milea yang berlatar di Bandung pada tahun 1990. Film ini diadaptasi dari novel karya Pidi Baiq."
  }
]
```

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

Hasil:

GET /api/Movies/{id}

Parameters

| Name | Description |
|---------------|-------------------------|
| id * required | integer(\$int32) (path) |

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7296/api/Movies/3' \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7296/api/Movies/3

Server response

| Code | Details |
|------|---|
| 200 | <p>Response body</p> <pre>{ "title": "Dilan 1990", "director": "Fajar Bustomi", "stars": ["Iqbaal Ramadhan", "Yanesha Prescilla", "Debo Andryos"], "description": "Kisah cinta remaja antara Dilan dan Milea yang berlatar di Bandung pada tahun 1990. Film ini diadaptasi dari novel karya Pidi Baiq." }</pre> |

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:7296/api/Movies/1' \
  -H 'accept: text/plain'
```

Request URL

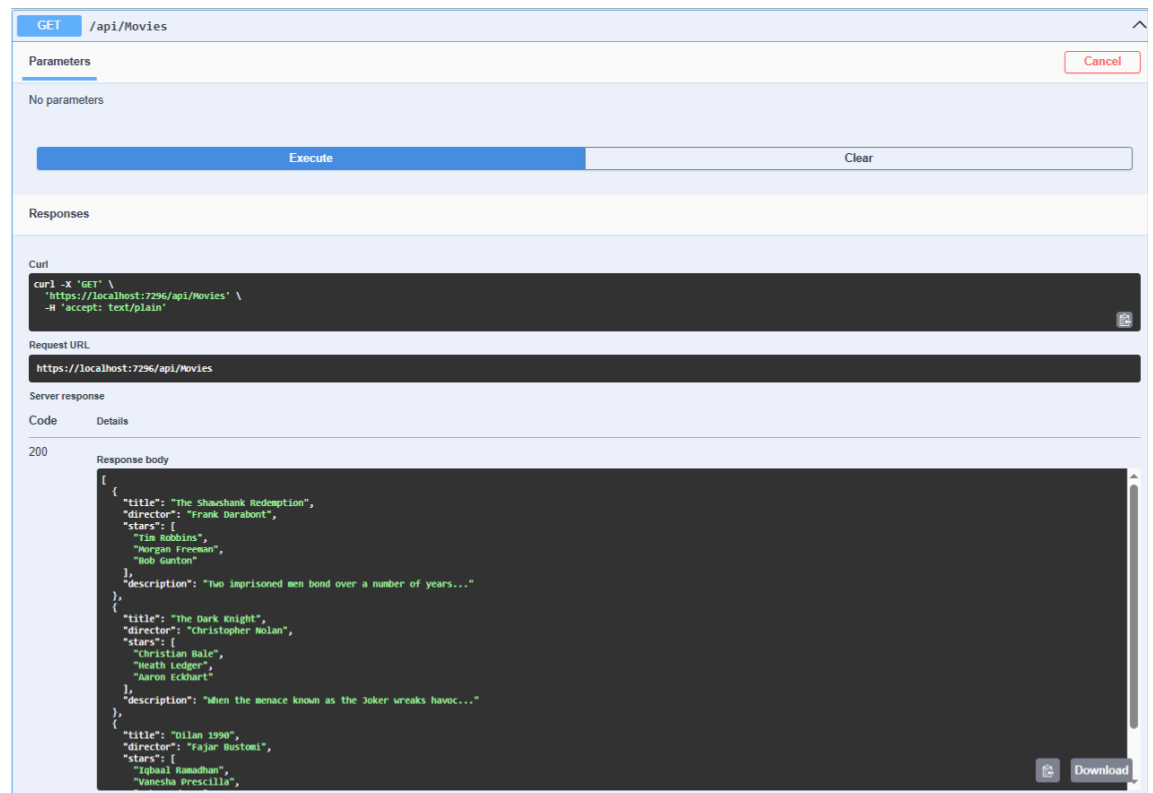
https://localhost:7296/api/Movies/1

Server response

| Code | Details |
|------|--|
| 200 | <p>Response body</p> <pre>[{ "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": "Dilan 1990", "director": "Fajar Bustomi", "stars": ["Iqbaal Ramadhan", "Yanesha Prescilla", "Debo Andryos"], "description": "Kisah cinta remaja antara Dilan dan Milea yang berlatar di Bandung pada tahun 1990. Film ini diadaptasi dari novel karya Pidi Baiq." }]</pre> |

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

Hasil:



- Penjelasan

Program ini adalah Web API sederhana menggunakan ASP.NET Core yang digunakan untuk mengelola data film. Data film disimpan dalam list statis bernama `movieList`, yang berisi 3 film teratas dari daftar IMDB. API ini menggunakan controller `MoviesController` untuk menangani permintaan dari pengguna melalui berbagai endpoint seperti GET, POST, dan DELETE. Endpoint GET `/api/Movies` digunakan untuk menampilkan seluruh daftar film, sedangkan GET `/api/Movies/{id}` menampilkan film berdasarkan indeks.

Endpoint POST `/api/Movies` berfungsi untuk menambahkan film baru ke dalam daftar, dan DELETE `/api/Movies/{id}` digunakan untuk menghapus film berdasarkan indeks. Semua data hanya disimpan di memori menggunakan list statis, sehingga akan hilang dan kembali ke kondisi awal jika aplikasi dimatikan atau di-restart. Atribut seperti `[HttpGet]`, `[HttpPost]`, dan `[HttpDelete]` digunakan untuk menentukan jenis permintaan HTTP yang ditangani oleh masing-masing method.