

**TUGAS JURNAL
KONSTRUKSI PERANGKAT LUNAK**

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



Disusun Oleh:

Nadia Putri Rahmانيar / 2211104012

S1 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

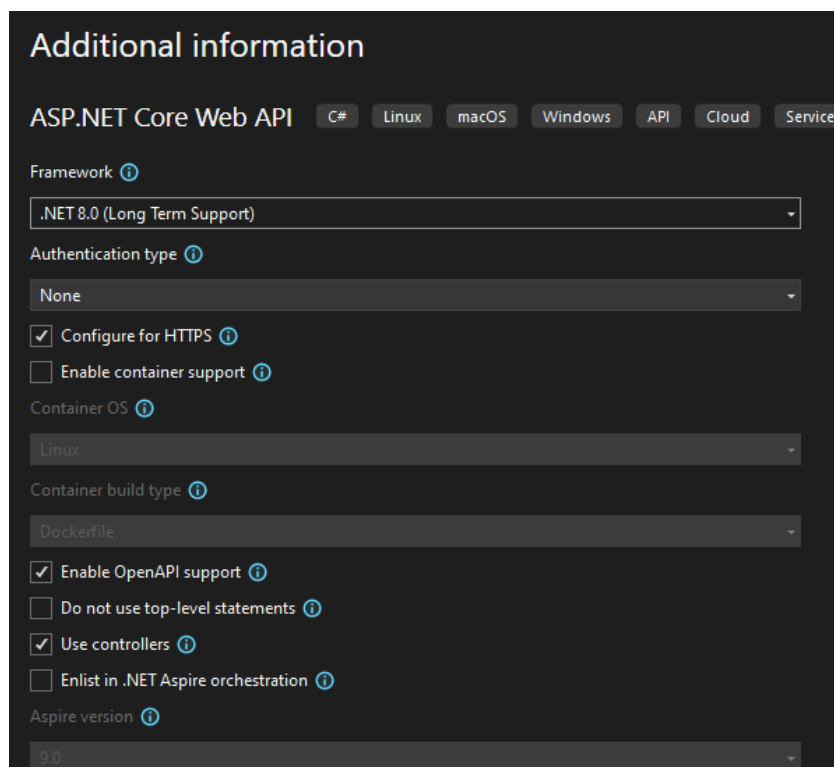
1. MEMBUAT PROJECT WEB API

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”.

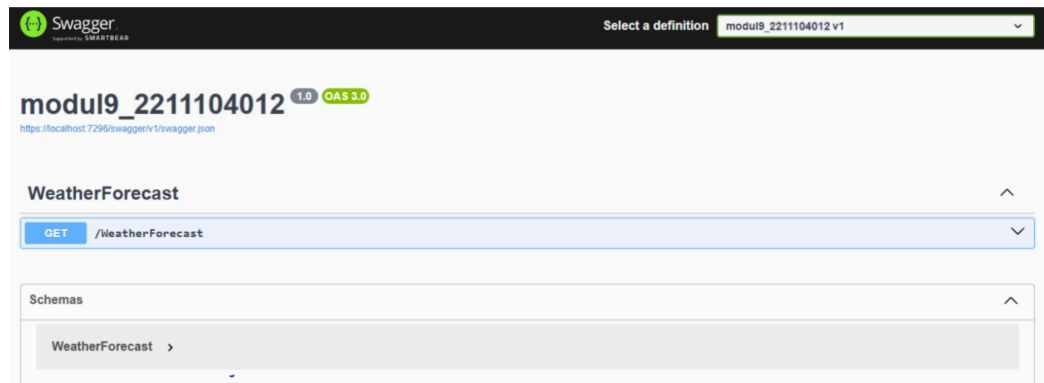
- Buka visual studio yang sudah terinstall dengan ASP.NET dan .NET 5.0 SDK atau setelahnya
- Pilih New Project dan kemudian pilih ASP.NET Core Web API atau API (pastikan opsi ‘Enable OpenAPI support’ tercentang).
- Pastikan untuk memilih .NET versi 5.0 atau yang lebih baru.
- Masukkan nama projek “modul9_NIM”.
- 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>
- Setelah project tersebut selesai dibuat, coba run programnya, dan tunggu sampai program selesai di-compile.

Jawab:



The screenshot shows the 'Additional information' dialog box in Visual Studio. The title is 'Additional information'. Below the title, there are tabs for 'ASP.NET Core Web API', 'C#', 'Linux', 'macOS', 'Windows', 'API', 'Cloud', and 'Service'. The 'ASP.NET Core Web API' tab is selected. The dialog contains several sections with configuration options:

- Framework**: A dropdown menu showing '.NET 8.0 (Long Term Support)'.
- Authentication type**: A dropdown menu showing 'None'.
- Configure for HTTPS**: A checkbox that is checked.
- Enable container support**: A checkbox that is unchecked.
- Container OS**: A dropdown menu showing 'Linux'.
- Container build type**: A dropdown menu showing 'Dockerfile'.
- Enable OpenAPI support**: A checkbox that is checked.
- Do not use top-level statements**: A checkbox that is unchecked.
- Use controllers**: A checkbox that is checked.
- Enlist in .NET Aspire orchestration**: A checkbox that is unchecked.
- Aspire version**: A dropdown menu showing '9.0'.



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_2211104012
{
    public class Movie
    {
        public string Title { get; set; }
        public string Director { get; set; }
        public List<string> Stars { get; set; }
        public string Description { get; set; }

        public Movie() { }
    }
}
```

Controllers/MoviesController.cs

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

namespace modul9_2211104012.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    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]
        public ActionResult<List<Movie>> Get() => movies;

        // GET: api/Movies/{id}
        [HttpGet("{id}")]
        public ActionResult<Movie> Get(int id)
        {
            // ...
        }
    }
}
```

```

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

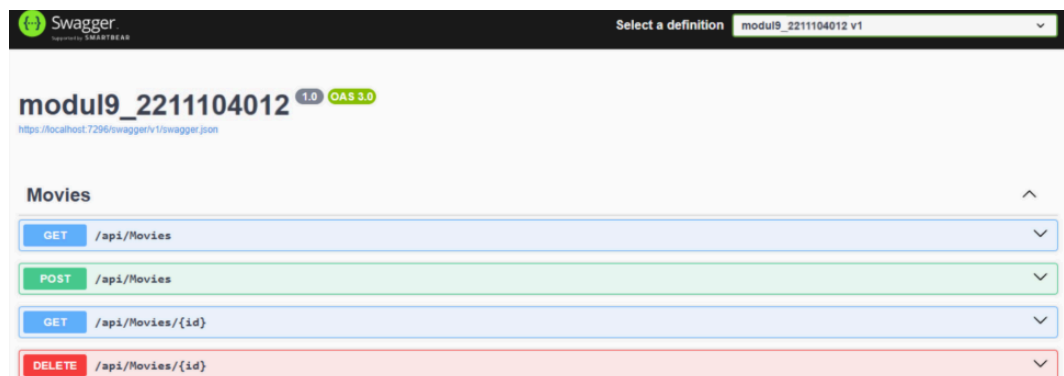
// GET: api/Movies/{id}
[HttpGet("{id}")]
public ActionResult<Movie> Get(int id)
{
    if (id < 0 || id >= movies.Count)
        return NotFound();
    return movies[id];
}

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

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

```

- Hasil run



Penjelasan :

Program ini merupakan implementasi sederhana dari Web API menggunakan ASP.NET Core yang dirancang untuk mengelola informasi film. Data film disimpan secara sementara dalam sebuah list statis bernama movieList, yang memuat tiga film terpopuler berdasarkan peringkat IMDB. Pengelolaan permintaan dari pengguna dilakukan melalui controller bernama MoviesController, yang menyediakan beberapa endpoint seperti GET, POST, dan DELETE.

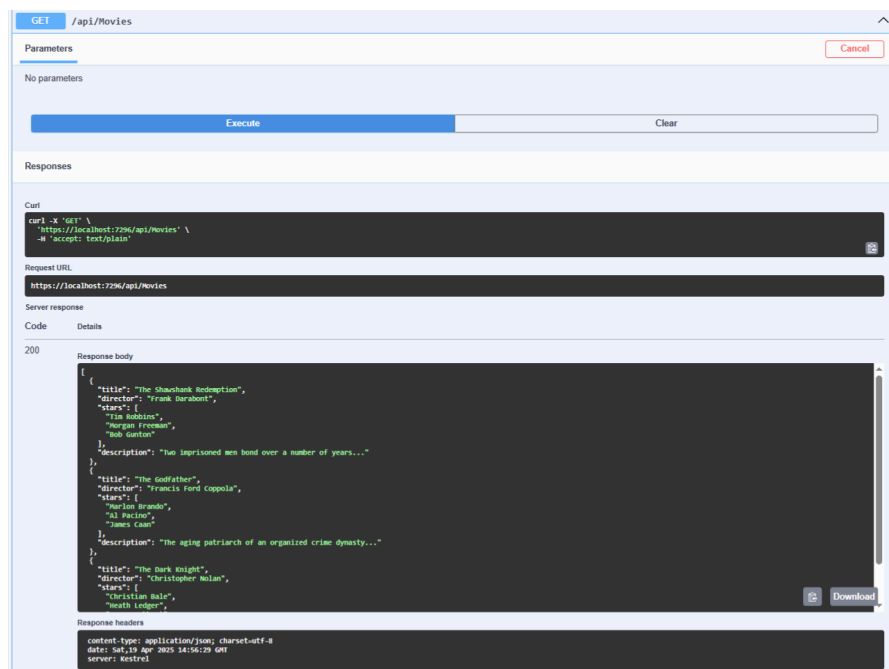
Endpoint GET /api/Movies digunakan untuk menampilkan seluruh koleksi film, sedangkan GET /api/Movies/{id} menampilkan detail film berdasarkan indeks tertentu. Untuk menambahkan film baru, digunakan endpoint POST /api/Movies, dan

penghapusan film berdasarkan indeks dilakukan melalui endpoint DELETE /api/Movies/{id}. Seluruh data hanya disimpan sementara di memori melalui list statis, sehingga akan ter-reset saat aplikasi dimatikan atau dijalankan ulang. Atribut seperti [HttpGet], [HttpPost], dan [HttpDelete] digunakan untuk mengatur jenis permintaan HTTP yang sesuai untuk setiap metode.

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”



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



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

```
Server response
Code    Details
200
Response body
{
  "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": "Dilan 1990",
  "director": "Fajar Bustomi",
  "stars": [
    "Iqbaal Ramadhan",
    "Vanessa 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:

GET /api/Movies/{id}

Parameters

Name Description

id * required Integer(\$int32) (path) 3

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

Response body

```
{
  "title": "Dilan 1990",
  "director": "Fajar Bustomi",
  "stars": [
    "Iqbaal Ramadhan",
    "Vanessa 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."
}
```

Download

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

DELETE /api/Movies/{id}

Parameters

Name Description

id * required Integer(\$int32) (path) 1

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

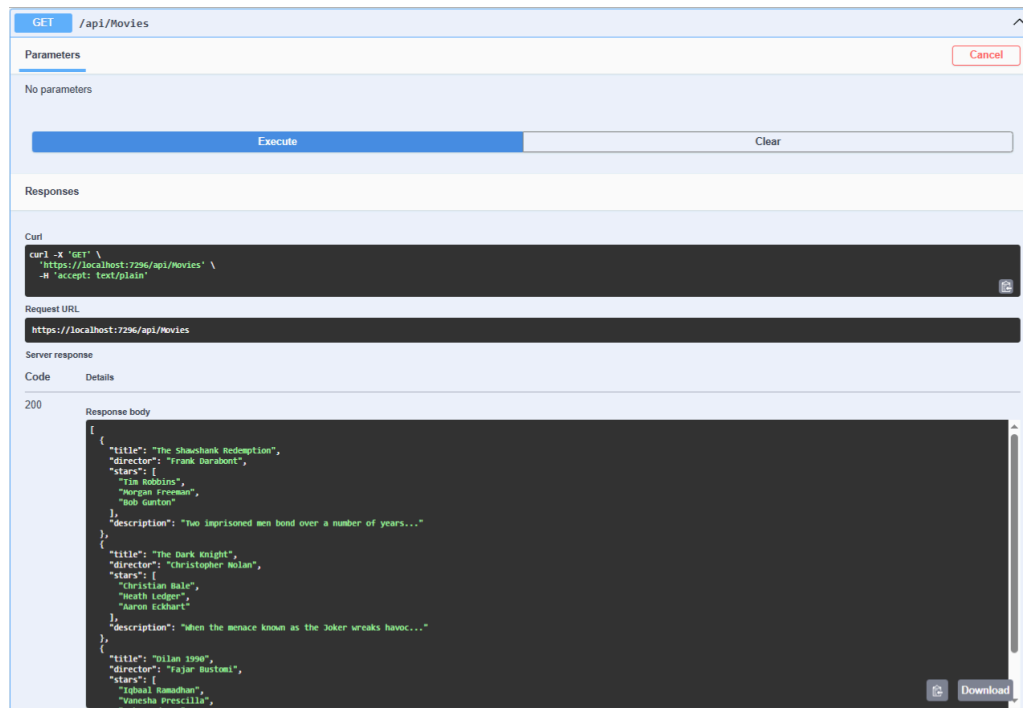
200

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": "Dilan 1990",
    "director": "Fajar Bustomi",
    "stars": [
      "Iqbaal Ramadhan",
      "Vanessa 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."
  }
]
```

Download

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



Penjelasan :

Program ini merupakan Web API sederhana berbasis ASP.NET Core yang dirancang untuk mengelola data mengenai film. Informasi film disimpan dalam sebuah list statis bernama `movieList`, yang berisi tiga film terbaik berdasarkan peringkat IMDB. API ini memanfaatkan controller bernama `MoviesController` untuk memproses berbagai permintaan dari pengguna melalui beberapa endpoint seperti GET, POST, dan DELETE.

Endpoint GET `/api/Movies` digunakan untuk menampilkan seluruh data film yang tersedia, sementara GET `/api/Movies/{id}` menampilkan detail film tertentu berdasarkan indeksinya. Untuk menambahkan data film baru, digunakan endpoint POST `/api/Movies`, sedangkan endpoint DELETE `/api/Movies/{id}` digunakan untuk menghapus film berdasarkan indeks yang diberikan. Seluruh data disimpan hanya dalam memori menggunakan list statis, sehingga akan otomatis terhapus dan kembali ke kondisi awal jika aplikasi dimatikan atau dijalankan ulang. Atribut seperti `[HttpGet]`, `[HttpPost]`, dan `[HttpDelete]` digunakan untuk menentukan jenis permintaan HTTP yang akan dilayani oleh masing-masing metode dalam controller.