

Nama : Muhammad Fariz Nur Hidayat

Kelas : SE063

NIM : 2211104069

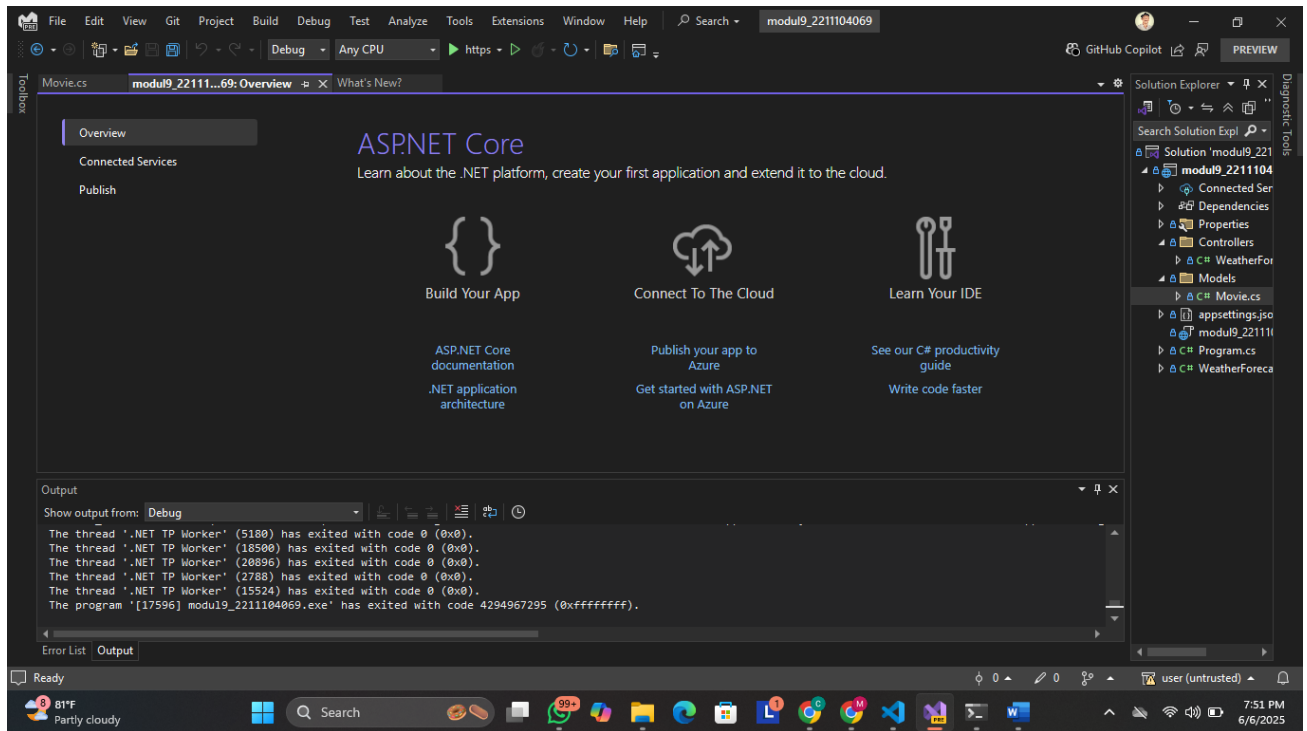
JURNAL MODUL 9

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.

BUKTI Pengerjaan



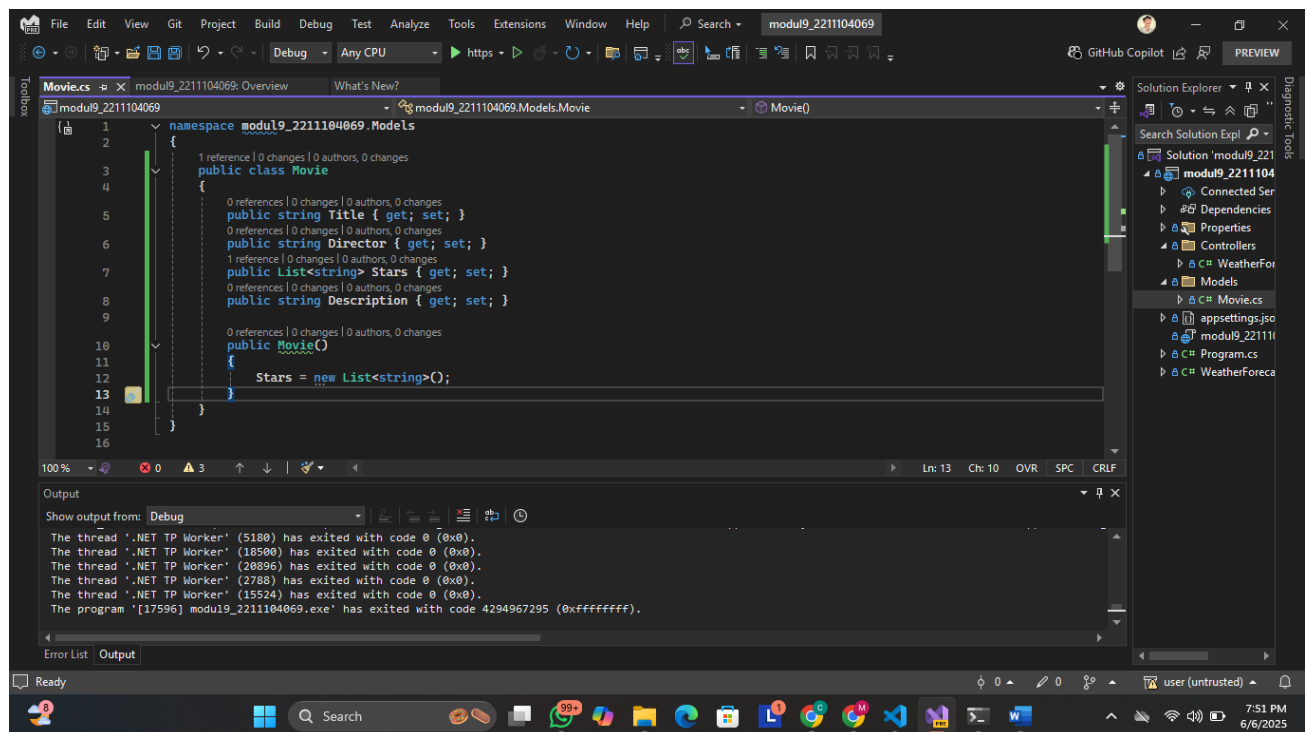
2. IMPLEMENTASI WEB API

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()

BUKTI Pengerjaan



Movie.cs

```
namespace modul9_2211104069.Models
{
    public class Movie
    {
        public string Title { get; set; }
```

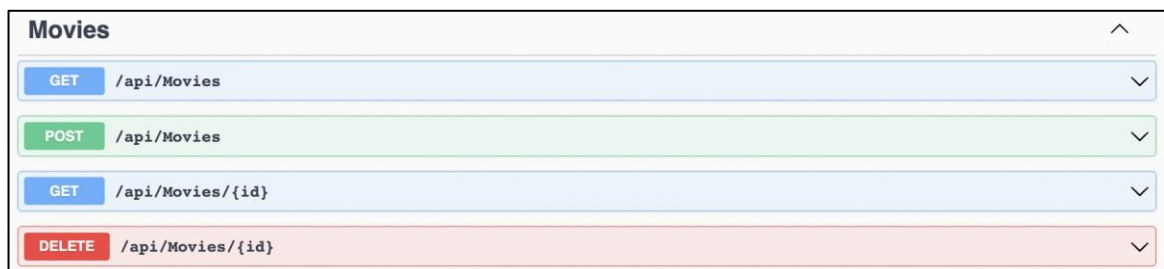
```
public string Director { get; set; }
public List<string> Stars { get; set; }
public string Description { get; set; }

public Movie()
{
    Stars = new List<string>();
}
}
```

Penjelasan kode:

Kode di atas merupakan definisi kelas `Movie` dalam namespace `modul9_2211104069.Models`, yang digunakan untuk merepresentasikan data sebuah film. Kelas ini memiliki empat properti: `Title` (judul film), `Director` (sutradara), `Stars` (daftar aktor/aktris yang membintangi film), dan `Description` (deskripsi film). Properti `Stars` bertipe `List<string>`, yang berarti dapat menyimpan lebih dari satu nama bintang film. Konstruktor `Movie()` digunakan untuk menginisialisasi properti `Stars` dengan objek `List<string>` kosong agar tidak null ketika objek `Movie` dibuat. Kelas ini cocok digunakan dalam aplikasi berbasis MVC untuk menyimpan dan mengelola informasi film.

- 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:<PORT>/swagger/index.html>):

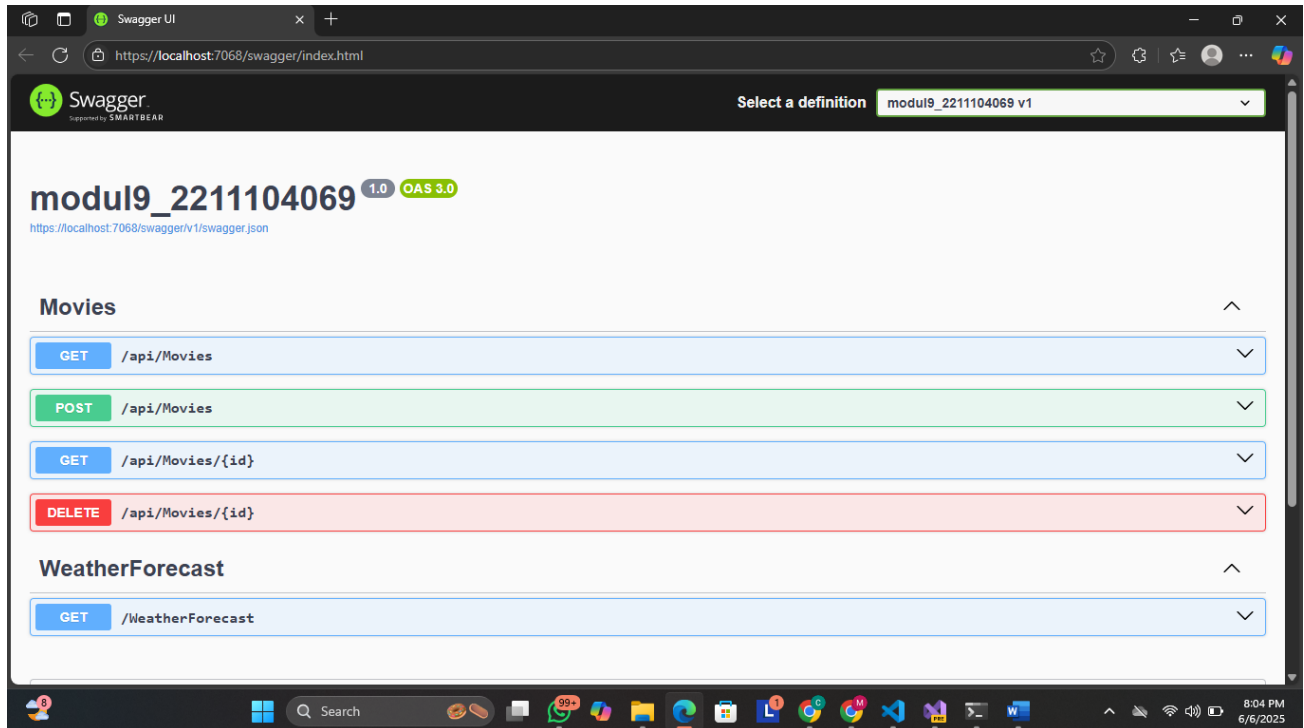


- GET `/api/Movies`: mengembalikan output berupa list/array dari semua objek `Movies`
 - GET `/api/Movies/{id}`: mengembalikan output berupa objek `Movie` untuk index "id"
 - POST `/api/Movies`: menambahkan objek `Movie` baru
 - DELETE `/api/Movies/{id}`: menghapus objek `Movie` pada index "id"
- C.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

BUKTI Pengerjaan



MoviesController.cs

```
namespace modul9_2211104069.Controllers
{
    using Microsoft.AspNetCore.Mvc; // Untuk ControllerBase dan ActionResult
    using modul9_2211104069.Models; // Mengimpor Movie dari Models
    using System.Collections.Generic;

    [Route("api/[controller]")]
    [ApiController]
    public class MoviesController : ControllerBase
    {
        // Static list untuk menyimpan data film
        public static List<Movie> Movies = new List<Movie>
        {
            new Movie
            {
```

```

        Title = "The Shawshank Redemption",
        Director = "Frank Darabont",
        Stars = new List<string> { "Tim Robbins", "Morgan Freeman" },
        Description = "Two imprisoned men bond over a number of years, finding solace and eventual
redemption through acts of common decency."
    },
    new Movie
    {
        Title = "The Godfather",
        Director = "Francis Ford Coppola",
        Stars = new List<string> { "Marlon Brando", "Al Pacino" },
        Description = "The aging patriarch of an organized crime dynasty transfers control of his
clandestine empire to his reluctant son."
    },
    new Movie
    {
        Title = "The Dark Knight",
        Director = "Christopher Nolan",
        Stars = new List<string> { "Christian Bale", "Heath Ledger" },
        Description = "When the menace known as the Joker emerges from his mysterious past, he
wreaks havoc and chaos on the people of Gotham."
    }
};

// GET /api/Movies: Mengembalikan semua movie
[HttpGet]
public IActionResult GetMovies()
{
    return Ok(Movies); // Mengembalikan seluruh list film
}

// GET /api/Movies/{id}: Mengembalikan movie berdasarkan ID
[HttpGet("{id}")]
public IActionResult GetMovie(int id)
{
    // Periksa apakah ID valid
    if (id < 0 || id >= Movies.Count)
    {
        return NotFound(); // Jika ID tidak ditemukan, kembalikan 404 NotFound
    }

    return Ok(Movies[id]); // Mengembalikan movie berdasarkan ID
}

```

```

// POST /api/Movies: Menambahkan movie baru
[HttpPost]
public IActionResult AddMovie([FromBody] Movie newMovie)
{
    if (newMovie == null)
    {
        return BadRequest("Movie data is required."); // Memastikan data valid
    }

    Movies.Add(newMovie); // Menambahkan movie ke dalam list
    return CreatedAtAction(nameof(GetMovie), new { id = Movies.Count - 1 }, newMovie); //
Mengembalikan HTTP 201 Created
}

// DELETE /api/Movies/{id}: Menghapus movie berdasarkan ID
[HttpDelete("{id}")]
public IActionResult DeleteMovie(int id)
{
    // Memeriksa apakah ID valid
    if (id < 0 || id >= Movies.Count)
    {
        return NotFound(); // Jika ID tidak ditemukan, kembalikan 404 NotFound
    }

    Movies.RemoveAt(id); // Menghapus movie berdasarkan ID
    return NoContent(); // Mengembalikan HTTP 204 No Content (berhasil menghapus)
}
}
}

```

Penjelasan kode:

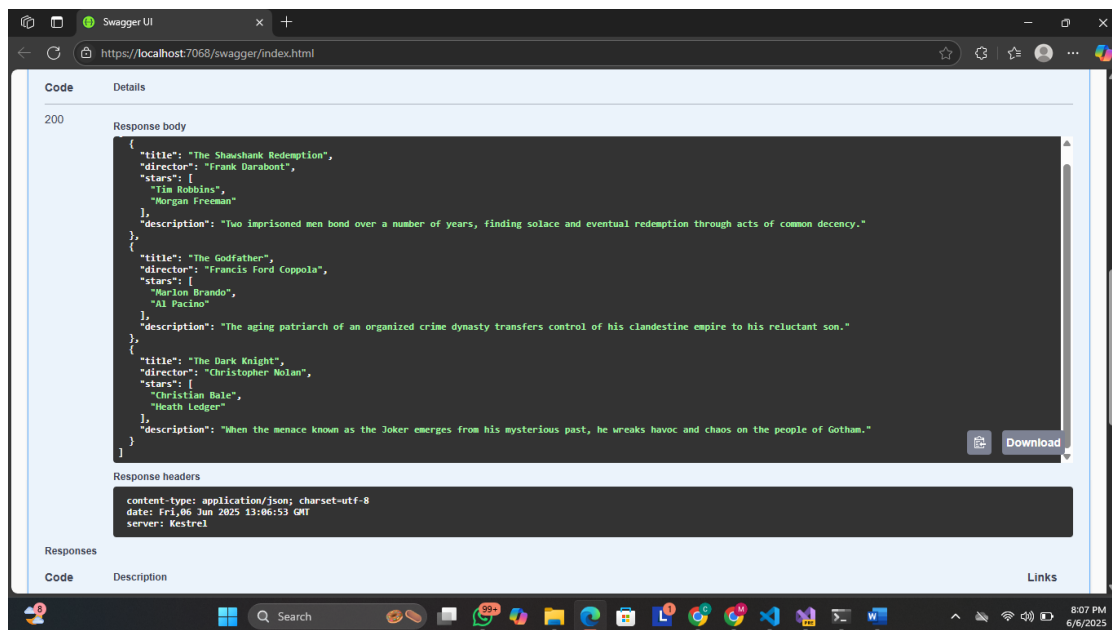
Kode di atas merupakan implementasi dari MoviesController, yaitu sebuah controller dalam ASP.NET Core Web API yang berada di namespace modul9_2211104069.Controllers. Controller ini menangani operasi CRUD sederhana untuk data film menggunakan list statis bertipe List<Movie>. Tiga film telah diinisialisasi secara default. Terdapat empat endpoint utama: GET /api/Movies untuk mengambil semua data film, GET /api/Movies/{id} untuk mengambil data film berdasarkan indeks, POST /api/Movies untuk menambahkan film baru, dan DELETE /api/Movies/{id} untuk menghapus film berdasarkan indeks. Validasi dilakukan untuk memastikan ID yang diminta ada, dan data yang dikirim saat menambah film tidak null. Controller ini memanfaatkan anotasi [ApiController] dan routing [Route("api/[controller]")] untuk mendukung pengelolaan data melalui HTTP request.

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 scenario 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”

BUKTI Pengerjaan



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

BUKTI Pengerjaan

The screenshot displays the Swagger UI interface in a web browser. The top section shows the endpoint **POST /api/Movies**. Under the **Parameters** tab, it states "No parameters". The **Request body** is set to **application/json** and contains a JSON object for a movie titled "Inception".

Below the request body, there are **Execute** and **Clear** buttons. The **Execute** button has been clicked, leading to the **Curl** section which shows the corresponding curl command:

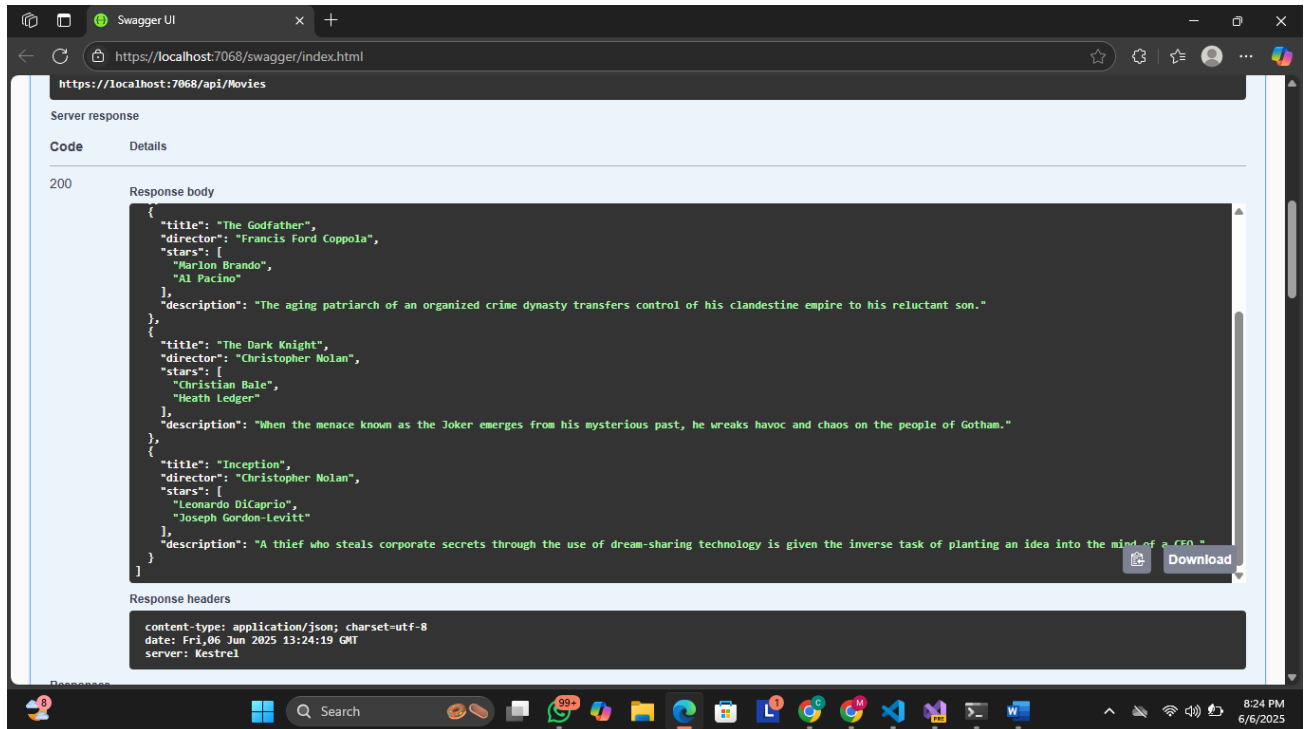
```
curl -X 'POST' \
  'https://localhost:7068/api/Movies' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "title": "Inception",
    "director": "Christopher Nolan",
    "stars": [
      "Leonardo DiCaprio",
      "Joseph Gordon-Levitt"
    ],
    "description": "A thief who steals corporate secrets through the use of dream-sharing technology is given the inverse task of planting an idea into the mind of a CEO."
  }'
```

The **Request URL** is `https://localhost:7068/api/Movies`. The **Server response** section shows a **201** status code (labeled as *Undocumented*). The **Response body** is a JSON object identical to the one in the request, with a **Download** button next to it. The **Response headers** are:

```
content-type: application/json; charset=utf-8
date: Fri, 06 Jun 2025 13:11:16 GMT
location: https://localhost:7068/api/Movies/3
server: Kestrel
```

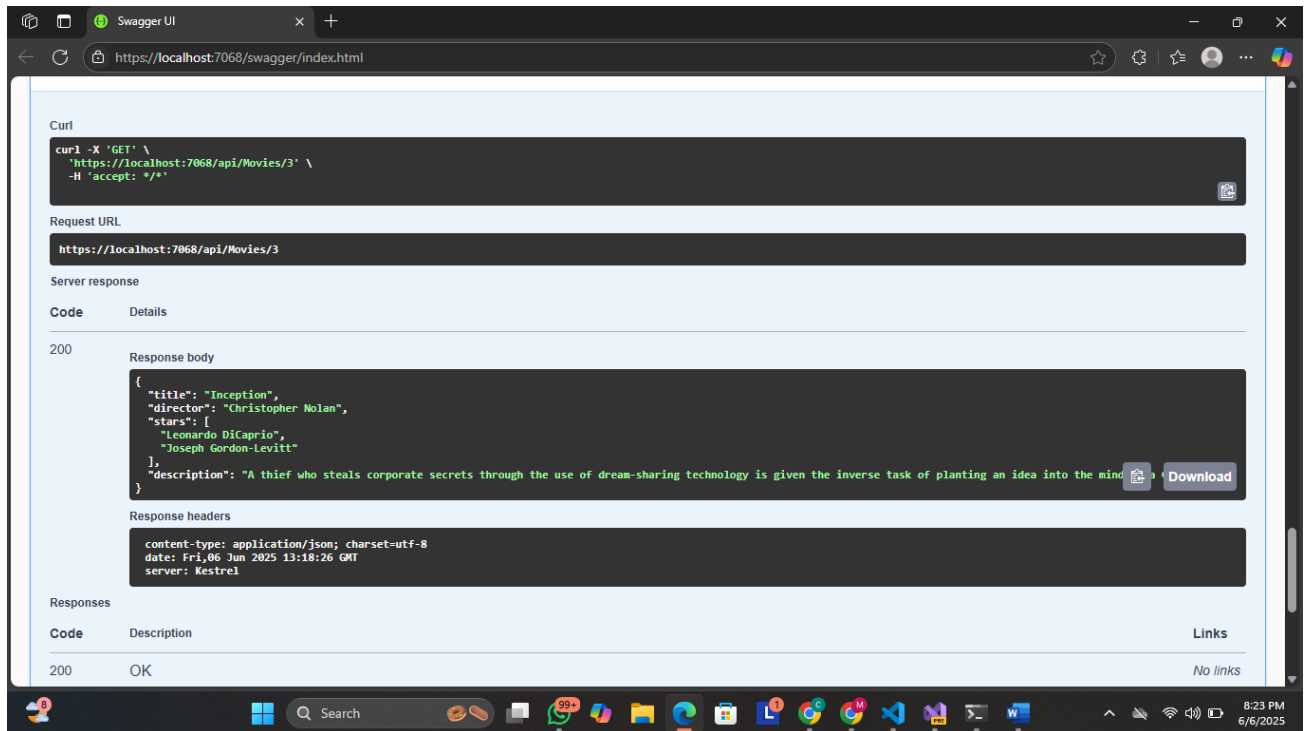
- C. Cek list/array dari semua Movie lagi dengan “GET /api/Movies”, pastikan Movie yang baru ditambahkan sebelumnya sudah ada:

BUKTI Pengerjaan



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

BUKTI Pengerjaan



The screenshot shows the Swagger UI interface in a web browser. The browser's address bar displays `https://localhost:7068/swagger/index.html`. The main content area is titled "Curl" and shows the following command:

```
curl -X 'GET' \
  'https://localhost:7068/api/Movies/3' \
  -H 'accept: */*'
```

Below the curl command, the "Request URL" is set to `https://localhost:7068/api/Movies/3`. The "Server response" section shows a `200` status code. The "Response body" is a JSON object:

```
{
  "title": "Inception",
  "director": "Christopher Nolan",
  "stars": [
    "Leonardo DiCaprio",
    "Joseph Gordon-Levitt"
  ],
  "description": "A thief who steals corporate secrets through the use of dream-sharing technology is given the inverse task of planting an idea into the mind of a CEO."
}
```

The "Response headers" section shows:

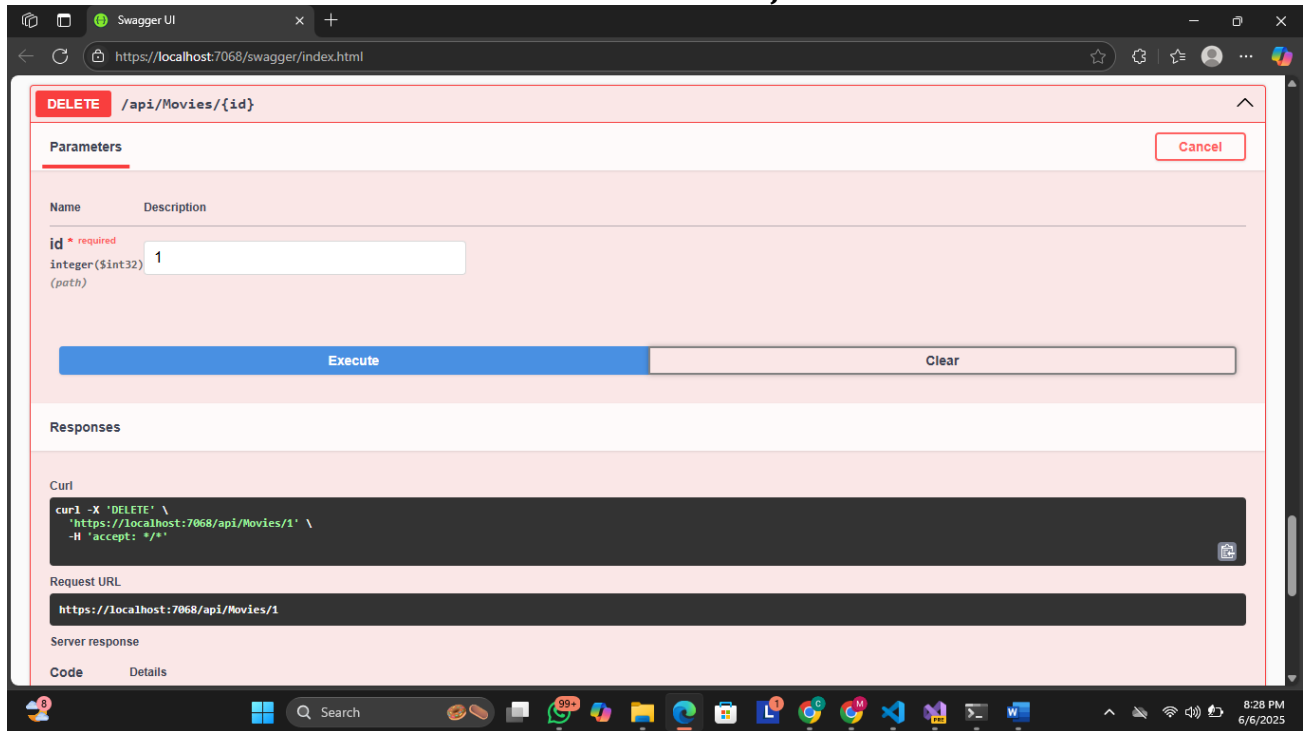
```
content-type: application/json; charset=utf-8
date: Fri, 06 Jun 2025 13:18:26 GMT
server: Kestrel
```

At the bottom, the "Responses" table shows a `200` status code with the description "OK".

Code	Description	Links
200	OK	No links

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

BUKTI Pengerjaan



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

BUKTI Pengerjaan

