

LAPORAN PRAKTIKUM JURNAL MODUL 9

NIM / Nama: 2311104072 – Jauhar Fajar Zuhair

Bagian I: Pendahuluan

Laporan ini mendokumentasikan pembuatan Web API menggunakan [ASP.NET](#) Core untuk mengelola data film (Movies). API ini menyimpan data film dalam sebuah list statis di memori dan menyediakan endpoint untuk operasi CRUD (Create, Read, Update - meskipun update tidak diimplementasikan di contoh ini, Delete).

Bagian II: Model Data `Movie.cs`

Kelas `Movie` mendefinisikan struktur data untuk sebuah film.

Isi `Movie.cs`:

```
namespace jmmodul9.Models; // Sesuaikan namespace jika berbeda

public class Movie
{
    // Properti ditambahkan berdasarkan contoh data di controller
    public int Id { get; set; } // Tambahkan ID unik
    public string? Title { get; set; }
    public string? Director { get; set; }
    public List<string>? Stars { get; set; } // Daftar bintang film
    public string? Description { get; set; }

    // Konstruktor default mungkin diperlukan oleh beberapa framework/library
    public Movie() { }

    // Konstruktor untuk inisialisasi mudah (tanpa ID, karena ID akan di-
    generate)
    public Movie(string title, string director, List<string> stars, string
description)
    {
        Title = title;
        Director = director;
        Stars = stars;
        Description = description;
    }
}
```

Penjelasan:

- Kelas `Movie` memiliki properti untuk menyimpan informasi film: `Id` (unik), `Title`, `Director`, `Stars` (sebagai list string), dan `Description`.
 - Tipe data `string?` digunakan untuk menunjukkan bahwa nilai string bisa null (fitur `Nullable Reference Types` di `C# modern`).
 - Sebuah `Id` ditambahkan untuk mempermudah identifikasi unik setiap film, terutama untuk operasi `GET by ID` dan `DELETE`.
 - Konstruktor disediakan untuk memudahkan pembuatan objek `Movie`.
-

Bagian III: Controller `MoviesController.cs`

Controller ini adalah inti dari API, menangani request HTTP dan berinteraksi dengan data film.

Deklarasi Controller dan Penyimpanan Data:

```
using Microsoft.AspNetCore.Mvc;
using jmmodul9.Models; // Sesuaikan namespace jika berbeda
using System.Collections.Generic;
using System.Linq; // Diperlukan untuk .FirstOrDefault, .Any, dll.

namespace jmmodul9.Controllers // Sesuaikan namespace jika berbeda
{
    [ApiController]
    [Route("api/[controller]")] // Route menjadi /api/Movies
    public class MoviesController : ControllerBase
    {
        // Data disimpan sementara di memori
        private static List<Movie> _movies = new List<Movie>
        {
            // Contoh data awal (ID ditambahkan manual untuk contoh ini)
            new Movie { Id = 1, 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 { Id = 2, 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..." },
            // Tambahkan film lain jika ada
        };

        // Counter untuk ID unik (jika ID tidak di-generate otomatis)
        private static int _nextId = _movies.Count > 0 ? _movies.Max(m => m.Id) + 1 : 1;

        // ... Action Methods (Endpoints) ...
    }
}
```

Penjelasan:

- `[ApiController]` dan `[Route("api/[controller]")]` mengonfigurasi kelas sebagai API controller dengan route dasar `/api/Movies`.
 - `_movies`: List statis `List<Movie>` digunakan untuk menyimpan data film di memori. Data awal ditambahkan untuk pengujian.
 - `_nextId`: Variabel statis untuk membantu memberikan ID unik saat menambahkan film baru (jika tidak menggunakan database yang bisa auto-increment).
-

Bagian IV: Endpoint API

Controller ini menyediakan endpoint berikut:

1. Get All Movies (GET /api/Movies)

```
[HttpGet] // Menangani GET request ke /api/Movies
public ActionResult<IEnumerable<Movie>> GetMovies()
{
    // Mengembalikan semua film dalam list
    return Ok(_movies);
}
```

Penjelasan: Mengembalikan seluruh daftar film yang tersimpan dengan status HTTP 200 (OK).

2. Get Movie by ID (GET /api/Movies/{id})

```
[HttpGet("{id}")] // Menangani GET request ke /api/Movies/angka (misal: /api/Movies/1)
public ActionResult<Movie> GetMovie(int id)
{
    // Cari film berdasarkan ID
    var movie = _movies.FirstOrDefault(m => m.Id == id);

    if (movie == null)
    {
        // Jika tidak ditemukan, kembalikan 404 Not Found
        return NotFound();
    }
    // Jika ditemukan, kembalikan film tersebut dengan status 200 OK
    return Ok(movie);
}
```

Penjelasan: Mencari dan mengembalikan film dengan `id` yang cocok. Jika tidak ditemukan, mengembalikan status HTTP 404 (Not Found).

3. Add New Movie (POST /api/Movies)

```
[HttpPost] // Menangani POST request ke /api/Movies
public ActionResult<Movie> PostMovie([FromBody] Movie movie) // Data film dari body request
{
    if (!ModelState.IsValid) // Validasi model (jika ada aturan validasi)
    {
        return BadRequest(ModelState);
    }

    // Berikan ID baru dan tambahkan ke list
    movie.Id = _nextId++;
    _movies.Add(movie);
}
```

```
// Kembalikan status 201 Created, dengan lokasi resource baru dan data film yang ditambahkan
// Menggunakan nameof(GetMovie) untuk merujuk ke action GET by ID
return CreatedAtAction(nameof(GetMovie), new { id = movie.Id }, movie);
}
```

Penjelasan: Menerima data film baru dari body request, memberikan ID unik, menambahkannya ke list, dan mengembalikan status HTTP 201 (Created) beserta data film baru dan URI-nya.

4. Delete Movie by ID (DELETE /api/Movies/{id})

```
[HttpDelete("{id}")] // Menangani DELETE request ke /api/Movies/angka
public IActionResult DeleteMovie(int id)
{
    // Cari film yang akan dihapus
    var movie = _movies.FirstOrDefault(m => m.Id == id);
    if (movie == null)
    {
        // Jika tidak ditemukan, kembalikan 404 Not Found
        return NotFound();
    }

    // Hapus film dari list
    _movies.Remove(movie);

    // Kembalikan status 204 No Content (sukses tanpa body response)
    return NoContent();
}
```

Penjelasan: Menghapus film dengan `id` yang cocok. Jika berhasil, mengembalikan status HTTP 204 (No Content). Jika film tidak ditemukan, mengembalikan 404 (Not Found).

Bagian V: Konfigurasi dan Menjalankan (Program.cs)

File `Program.cs` berisi konfigurasi dasar untuk menjalankan aplikasi web API.

Isi `Program.cs` (Ringkasan):

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers(); // Mendaftarkan service controller

// Menambahkan konfigurasi Swagger/OpenAPI untuk dokumentasi dan pengujian
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger(); // Mengaktifkan middleware Swagger
    app.UseSwaggerUI(); // Mengaktifkan UI Swagger di /swagger
}

app.UseHttpsRedirection(); // Mengalihkan HTTP ke HTTPS
app.UseAuthorization(); // Mengaktifkan middleware otorisasi (jika digunakan)
app.MapControllers(); // Memetakan request ke controller

app.Run(); // Menjalankan aplikasi
```

Penjelasan:

- Kode ini menyiapkan dependency injection (`AddControllers`), menambahkan dukungan Swagger/OpenAPI (`AddEndpointsApiExplorer`, `AddSwaggerGen`), dan mengonfigurasi pipeline request HTTP (termasuk routing ke controller dan penggunaan Swagger UI di environment Development).

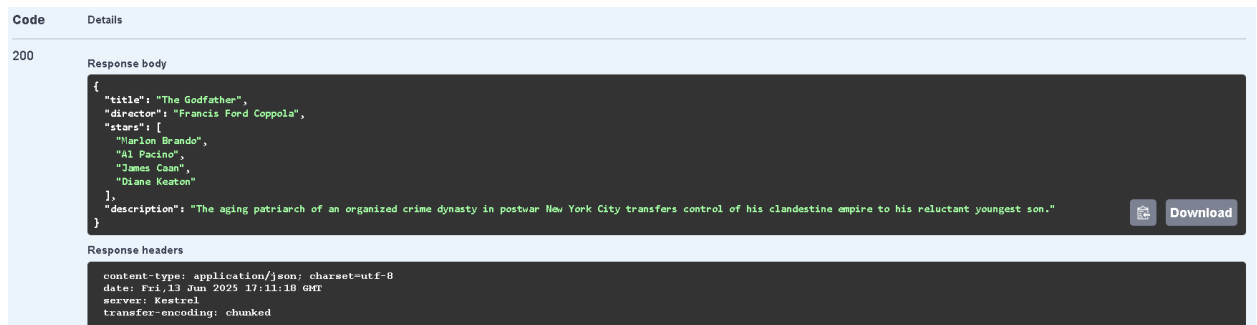
Bagian VI: Pengujian API

API dapat diuji menggunakan:

- Swagger UI:** Akses `/swagger` pada URL aplikasi saat berjalan (misalnya `https://localhost:xxxx/swagger`). UI ini memungkinkan pengujian interaktif semua endpoint.
- Postman/Insomnia:** Alat pengujian API pihak ketiga untuk mengirim request HTTP kustom (GET, POST, DELETE, dll.) ke endpoint API.

Contoh Hasil Pengujian:

- **Request:** GET /api/Movies/1**Response (200 OK):**



Code Details

200

Response body

```
{
  "title": "The Godfather",
  "director": "Francis Ford Coppola",
  "stars": [
    "Marlon Brando",
    "Al Pacino",
    "James Caan",
    "Diane Keaton"
  ],
  "description": "The aging patriarch of an organized crime dynasty in postwar New York City transfers control of his clandestine empire to his reluctant youngest son."
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 13 Jun 2025 17:11:10 GMT
server: Kestrel
transfer-encoding: chunked
```

Download

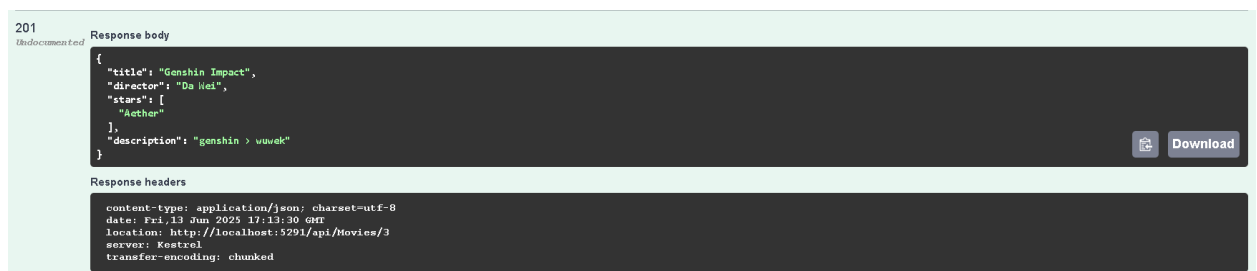
- **Request:** POST /api/Movies dengan Request Body:



Request body

```
{
  "title": "Genshin Impact",
  "director": "Da Wei",
  "stars": [
    "Aether"
  ],
  "description": "genshin > wuwei"
}
```

Response (201 Created):



201

Undocumented

Response body

```
{
  "title": "Genshin Impact",
  "director": "Da Wei",
  "stars": [
    "Aether"
  ],
  "description": "genshin > wuwei"
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Fri, 13 Jun 2025 17:13:30 GMT
location: http://localhost:5291/api/Movies/3
server: Kestrel
transfer-encoding: chunked
```

Download

- **Request:** DELETE /api/Movies/2**Response (204 No Content):** (Tidak ada body)

Laporan ini merangkum proses pembuatan API sederhana untuk data film menggunakan [ASP.NET](#) Core, dari model hingga controller dan pengujian endpoint.