

Nama : Muhammad Shafiq Rasuna

Nim : 2311104043

Kelas : SE0702

Link Github : https://github.com/rasunaaa/KPL_MUHAMMAD-SHAFIQ-RASUNA_2311104043_SE0702

1. MovieAPI.Controllers

```
1  using Microsoft.AspNetCore.Mvc;
2  using MovieAPI.Models;
3
4  namespace MovieAPI.Controllers
5  {
6      [ApiController]
7      [Route("api/[controller]")]
8      public class MoviesController : ControllerBase
9      {
10         private static List<Movie> movies = new List<Movie>
11         {
12             new Movie
13             {
14                 Title = "The Shawshank Redemption",
15                 Director = "Frank Darabont",
16                 Stars = new List<string>{ "Tim Robbins", "Morgan Freeman" },
17                 Description = "Two imprisoned men bond over a number of years..."
18             },
19             new Movie
20             {
21                 Title = "The Godfather",
22                 Director = "Francis Ford Coppola",
23                 Stars = new List<string>{ "Marlon Brando", "Al Pacino" },
24                 Description = "The aging patriarch of an organized crime dynasty..."
25             },
26             new Movie
27             {
28                 Title = "The Dark Knight",
29                 Director = "Christopher Nolan",
30                 Stars = new List<string>{ "Christian Bale", "Heath Ledger" },
31                 Description = "When the menace known as the Joker wreaks havoc..."
32             }
33         };
34
35         [HttpGet]
36         public ActionResult<List<Movie>> GetAllMovies() => movies;
37
38         [HttpGet("{id}")]
39         public ActionResult<Movie> GetMovieById(int id)
40         {
41             if (id < 0 || id >= movies.Count) return NotFound();
42             return movies[id];
43         }
44
45         [HttpPost]
46         public ActionResult AddMovie([FromBody] Movie newMovie)
47         {
48             movies.Add(newMovie);
49             return Ok();
50         }
51
52         [HttpDelete("{id}")]
53         public ActionResult DeleteMovie(int id)
54         {
55             if (id < 0 || id >= movies.Count) return NotFound();
56             movies.RemoveAt(id);
57             return Ok();
58         }
59     }
60 }
```

MoviesController adalah API controller di ASP.NET Core yang mengelola data film menggunakan list statik. Terdapat empat endpoint: GET untuk menampilkan semua film, GET by id untuk mengambil film berdasarkan indeks, POST untuk menambah film baru, dan DELETE untuk menghapus film berdasarkan indeks. Semua data disimpan sementara di memori (tanpa database).

2. WeatherForecastController.cs

```
modul9_04043 JurnalModul9_2311104043.Controllers.Weathe
1 using Microsoft.AspNetCore.Mvc;
2 using modul9_04043;
3
4 namespace JurnalModul9_2311104043.Controllers;
5
6 [ApiController]
7 [Route("[controller]")]
8 public class WeatherForecastController : ControllerBase
9 {
10     private static readonly string[] Summaries = new[]
11     {
12         "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"
13     };
14
15     private readonly ILogger<WeatherForecastController> _logger;
16
17     public WeatherForecastController(ILogger<WeatherForecastController> logger)
18     {
19         _logger = logger;
20     }
21
22     [HttpGet(Name = "GetWeatherForecast")]
23     public IEnumerable<WeatherForecast> Get()
24     {
25         return Enumerable.Range(1, 5).Select(index => new WeatherForecast
26         {
27             Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
28             TemperatureC = Random.Shared.Next(-20, 55),
29             Summary = Summaries[Random.Shared.Next(Summaries.Length)]
30         })
31         .ToArray();
32     }
33 }
```

Kode tersebut merupakan controller `WeatherForecastController` dalam proyek ASP.NET Core Web API yang berfungsi untuk menyediakan data prakiraan cuaca secara acak. Controller ini berada di namespace `JurnalModul9_2311104043.Controllers` dan menggunakan atribut `[ApiController]` serta `[Route("[controller]")]`, yang secara otomatis menetapkan route endpoint menjadi `WeatherForecast`. Controller ini memiliki dependency `ILogger` untuk logging aktivitas, meskipun tidak digunakan dalam kode saat ini.

Bagian utama dari controller adalah metode `Get()`, yang ditandai dengan atribut `[HttpGet(Name = "GetWeatherForecast")]`. Metode ini akan mengembalikan lima data prakiraan cuaca. Data ini dibuat menggunakan `Enumerable.Range(1, 5)` yang mengulangi proses sebanyak lima kali. Untuk setiap item, akan dibuat objek `WeatherForecast` dengan tiga properti: `Date` (tanggal dari hari ini + indeks hari), `TemperatureC` (suhu acak antara -20 hingga 55 derajat Celsius), dan `Summary` (deskripsi cuaca yang dipilih secara acak dari array `Summaries`, seperti "Freezing", "Hot", dll). Data dikembalikan dalam bentuk array dan biasanya digunakan untuk menampilkan contoh output JSON dari Web API. Kode ini sangat umum ditemukan sebagai template default ketika membuat proyek ASP.NET Core Web API baru.

3. Movie.cs

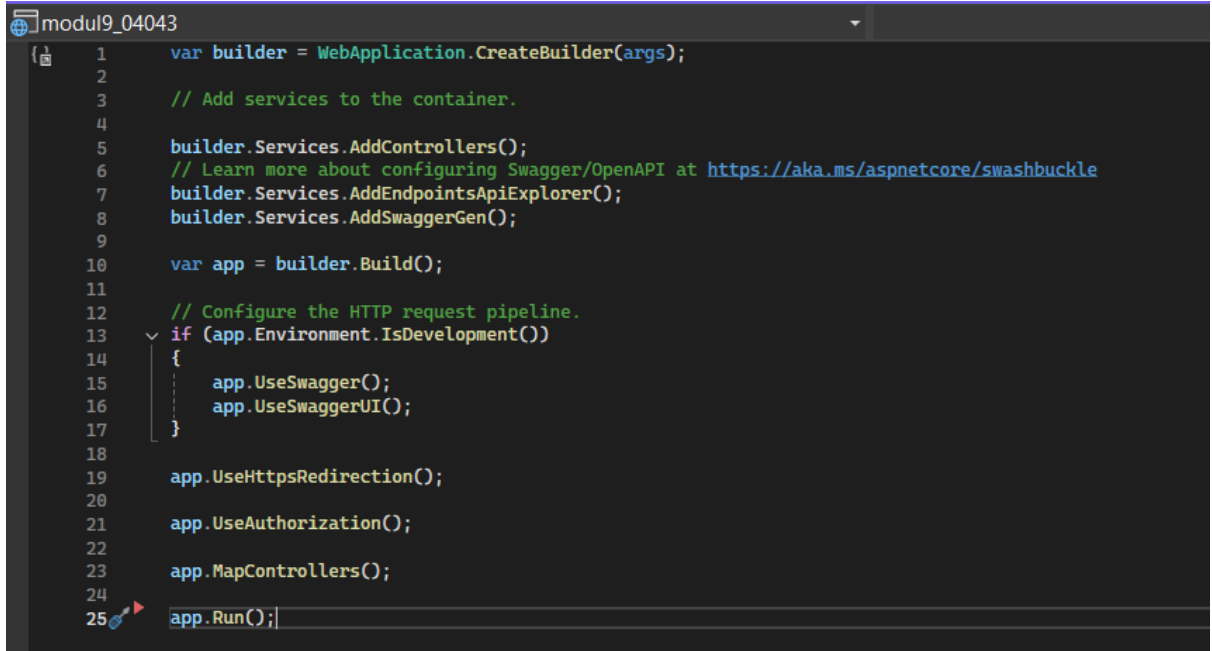
```
modul9_04043 JurnalModul9_2311104043.Controllers.Weat
1 using Microsoft.AspNetCore.Mvc;
2
3 namespace JurnalModul9_2311104043.Controllers;
4
5 [ApiController]
6 [Route("[controller]")]
7
8 public class WeatherForecastController : ControllerBase
9 {
10     private static readonly string[] Summaries = new[]
11     {
12         "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"
13     };
14
15     private readonly ILogger<WeatherForecastController> _logger;
16
17     public WeatherForecastController(ILogger<WeatherForecastController> logger)
18     {
19         _logger = logger;
20     }
21
22     [HttpGet(Name = "GetWeatherForecast")]
23     public IEnumerable<WeatherForecast> Get()
24     {
25         return Enumerable.Range(1, 5).Select(index => new WeatherForecast
26         {
27             Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
28             TemperatureC = Random.Shared.Next(-20, 55),
29             Summary = Summaries[Random.Shared.Next(Summaries.Length)]
30         })
31         .ToArray();
32     }
33 }
```

Kode di atas adalah implementasi **WeatherForecastController**, sebuah **API controller** di ASP.NET Core yang digunakan untuk menampilkan data prakiraan cuaca. Controller ini berada dalam namespace `JurnalModul9_2311104043.Controllers` dan menggunakan atribut `[ApiController]` serta `[Route("[controller]")]`, sehingga URL endpoint-nya menjadi `WeatherForecast`.

Controller ini memiliki array statis `Summaries` yang berisi kumpulan deskripsi cuaca, seperti "Freezing", "Hot", dan sebagainya. Pada metode `Get()` yang ditandai dengan `[HttpGet(Name = "GetWeatherForecast")]`, controller akan mengembalikan 5 data cuaca secara acak. Setiap data berisi tanggal (berurutan mulai dari hari ini +1), suhu dalam derajat Celcius (acak antara -20 hingga 55), dan ringkasan cuaca (dipilih secara acak dari `Summaries`). Data dikembalikan dalam bentuk array `WeatherForecast`.

Kode ini merupakan template standar dari proyek ASP.NET Core Web API dan berguna sebagai contoh dasar pembuatan endpoint dengan data dummy.

4. Program.cs



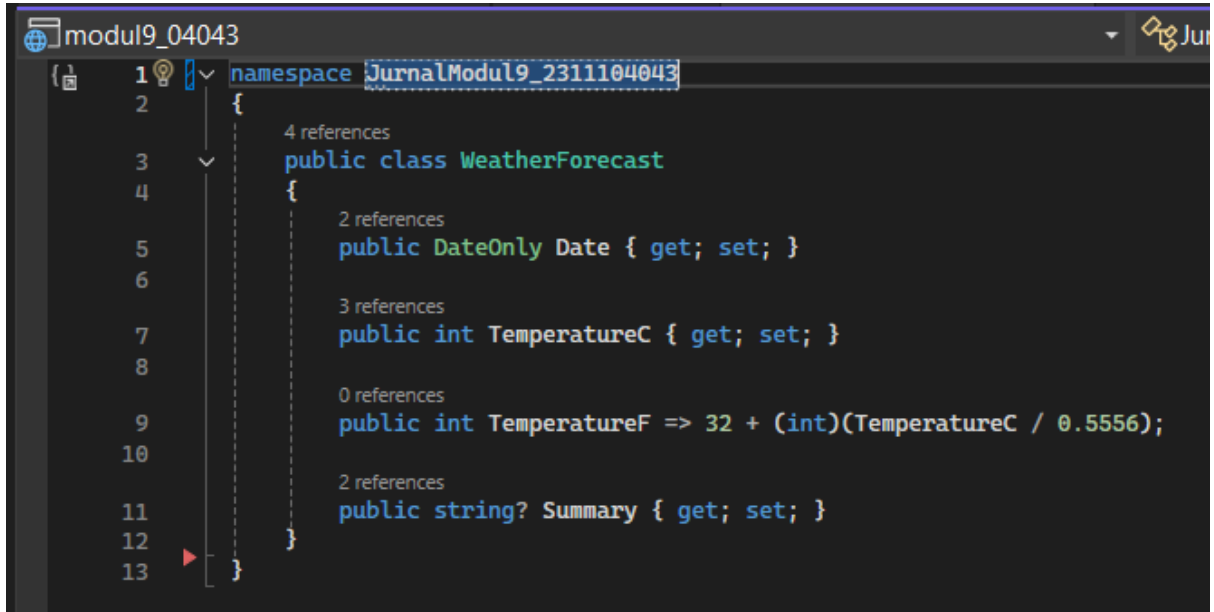
```
1  var builder = WebApplication.CreateBuilder(args);
2
3  // Add services to the container.
4
5  builder.Services.AddControllers();
6  // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
7  builder.Services.AddEndpointsApiExplorer();
8  builder.Services.AddSwaggerGen();
9
10 var app = builder.Build();
11
12 // Configure the HTTP request pipeline.
13 if (app.Environment.IsDevelopment())
14 {
15     app.UseSwagger();
16     app.UseSwaggerUI();
17 }
18
19 app.UseHttpsRedirection();
20
21 app.UseAuthorization();
22
23 app.MapControllers();
24
25 app.Run();
```

Kode tersebut memulai dengan membuat objek builder menggunakan `WebApplication.CreateBuilder(args)`, yang menyiapkan semua konfigurasi dasar untuk aplikasi web. Kemudian, beberapa layanan penting didaftarkan ke container DI (Dependency Injection), termasuk `AddControllers()` untuk mendukung penggunaan controller, serta `AddEndpointsApiExplorer()` dan `AddSwaggerGen()` yang digunakan untuk mendukung dokumentasi API menggunakan Swagger/OpenAPI.

Setelah konfigurasi selesai, aplikasi dibangun menggunakan `builder.Build()`. Pada tahap berikutnya, dilakukan konfigurasi pipeline HTTP. Jika aplikasi berjalan di lingkungan pengembangan (Development), maka dokumentasi Swagger akan diaktifkan melalui `app.UseSwagger()` dan `app.UseSwaggerUI()` agar API bisa diuji langsung melalui antarmuka web. Middleware `app.UseHttpsRedirection()` mengarahkan permintaan HTTP ke HTTPS, dan `app.UseAuthorization()` menyiapkan middleware untuk otorisasi (meskipun autentikasi belum dikonfigurasi di sini). Terakhir, `app.MapControllers()` akan memetakan route dari controller ke endpoint, dan `app.Run()` akan menjalankan aplikasi.

Secara keseluruhan, kode ini adalah template standar Program.cs yang disediakan saat membuat proyek ASP.NET Core Web API versi terbaru (.NET 6 ke atas) dengan arsitektur minimal hosting model.

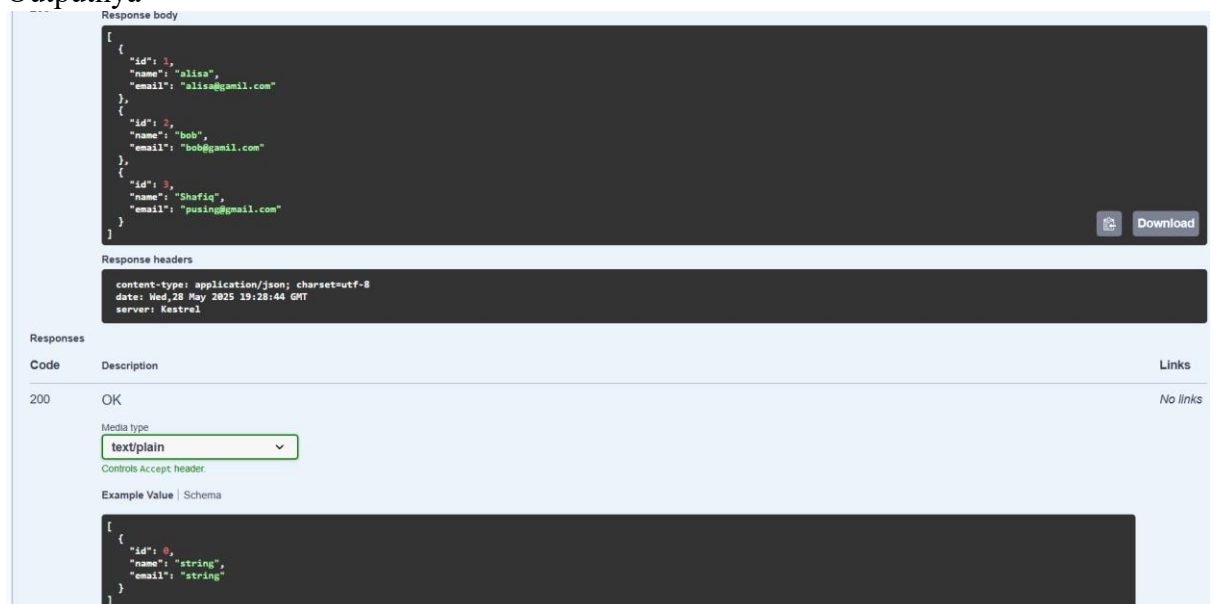
5. WeatherForecast.cs



```
1 namespace JurnalModul9_2311104043
2 {
3     4 references
4     public class WeatherForecast
5     {
6         2 references
7         public DateOnly Date { get; set; }
8
9         3 references
10        public int TemperatureC { get; set; }
11
12        0 references
13        public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
14
15        2 references
16        public string? Summary { get; set; }
17    }
18 }
```

Kode di atas merupakan definisi sebuah kelas bernama WeatherForecast yang berada dalam namespace JurnalModul9_2311104043. Kelas ini digunakan untuk merepresentasikan informasi prakiraan cuaca. Di dalam kelas terdapat beberapa properti, yaitu Date yang bertipe DateOnly untuk menyimpan tanggal prakiraan cuaca, TemperatureC yang menyimpan suhu dalam derajat Celsius, dan Summary yang berupa string opsional untuk mendeskripsikan kondisi cuaca, seperti "cerah" atau "berawan". Selain itu, terdapat properti TemperatureF yang bersifat read-only dan secara otomatis menghitung suhu dalam derajat Fahrenheit berdasarkan nilai suhu Celsius dengan rumus konversi yang sudah ditentukan. Dengan struktur ini, kelas WeatherForecast memudahkan penyimpanan dan pengelolaan data cuaca dalam aplikasi yang dikembangkan.

Outputnya



Response body

```
{
  {
    "id": 1,
    "name": "alisa",
    "email": "alisa@gmail.com"
  },
  {
    "id": 2,
    "name": "bob",
    "email": "bob@gmail.com"
  },
  {
    "id": 3,
    "name": "Shafiq",
    "email": "pusing@gmail.com"
  }
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Wed, 28 May 2025 19:28:44 GMT
server: Kestrel
```

Responses

| Code | Description | Links |
|------|-------------|----------|
| 200 | OK | No links |

Media type: text/plain

Controls: Accept header

Example Value | Schema

```
{
  {
    "id": 0,
    "name": "string",
    "email": "string"
  }
}
```

```
"name": "Shafiq",
"email": "pusing@gmail.com"
},

```

Request URL

https://localhost:7282/User

Server response

| Code | Details |
|------|---|
| 200 | <p>Response body</p> <pre>{ "id": 3, "name": "Shafiq", "email": "pusing@gmail.com" }</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Wed, 28 May 2025 19:26:23 GMT server: Kestrel</pre> |

Responses

| Code | Description | Links |
|------|-------------|---------|
| 200 | OK | No link |

Media type

text/plain

Controls Accept header:

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "email": "string"
}
```

```
curl -X 'GET' \
  'https://localhost:7282/User/1' \
  -H 'accept: text/plain'

```

Request URL

https://localhost:7282/User/1

Server response

| Code | Details |
|------|---|
| 200 | <p>Response body</p> <pre>{ "id": 1, "name": "alisa", "email": "alisa@gmail.com" }</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Wed, 28 May 2025 19:27:18 GMT server: Kestrel</pre> |

Responses

| Code | Description | Links |
|------|-------------|----------|
| 200 | OK | No links |

Media type

text/plain

Controls Accept header:

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "email": "string"
}
```

Curl

```
curl -X 'PUT' \
  'https://localhost:7282/User/3' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Shafiq",
    "email": "pusing2@gmail.com"
  }'

```

Request URL

https://localhost:7282/User/3

Server response

| Code | Details |
|------|--|
| 200 | <p>Response body</p> <pre>{ "id": 3, "name": "Shafiq", "email": "pusing2@gmail.com" }</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Wed, 28 May 2025 19:29:24 GMT server: Kestrel</pre> |

Responses

| Code | Description | Links |
|------|-------------|----------|
| 200 | OK | No links |

