



# Back End Development 1

## sesi 10

---



# **Best Practice<sup>+</sup> : RESTFUL API**

## BEST PRACTICE : CREATE RESTFUL API

Pada sesi kali ini, kita akan latihan menyiapkan RESTFUL API yang memiliki kemampuan untuk CRUD dan akan kita utilize dengan SQLite untuk store data.

Nah di sesi kali ini kita akan lakukan semua dalam VS Code dimana berbeda dengan cara-cara sebelumnya, disini kita akan explore dan create API dengan CLI.

Sebelum mulai pastikan 4 tools ini sudah diinstall ya :

1. Visual Studio code (<https://code.visualstudio.com/>)
2. Dotnet core SDK (<https://dotnet.microsoft.com/download>)
3. Postman (<https://www.postman.com/downloads/>)
4. DBeaver (<https://dbeaver.io/download/>)



## BEST PRACTICE : RESTFUL API - Sesi 10

# SET UP Environment

Pada sesi kali ini, kita akan latihan menyiapkan Environment terlebih dahulu dengan CLI untuk membuat App dan utilisasi dengan Package Entity & SQLite mulai dari :

### 1. Create our App ,

Tahap dimana kita inisiasi project dotnet caranya :

```
Maliks-MacBook-Air:Sesi10 swijaya$ dotnet new webapi -n "TodoApp" -lang "C#" -au none
```

```
Maliks-MacBook-Air:Sesi10 swijaya$ dotnet new webapi -n "TodoApp" -lang "C#" -au none
The template "ASP.NET Core Web API" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on TodoApp/TodoApp.csproj...
  Determining projects to restore...
  Restored /Users/swijaya/Downloads/C#/SourceCode/Sesi10/TodoApp/TodoApp.csproj (in 217 ms).
Restore succeeded.
```

Jangan lupa masuk ke folder TodoApp yang sudah kita buat :

```
Maliks-MacBook-Air:Sesi10 swijaya$ cd TodoApp
```

## 2. Utilize with Entity Framework & SQLite

Tambahkan package yang kita butuhkan untuk menggunakan Framework Entity & SQLite

```
Maliks-MacBook-Air:ToDoApp swijaya$ dotnet add package Microsoft.EntityFrameworkCore.Sqlite
```

```
Maliks-MacBook-Air:ToDoApp swijaya$ dotnet add package Microsoft.EntityFrameworkCore.Sqlite
Determining projects to restore...
Writing /var/folders/_1/jhrtgbls3m305qftdq4y0b580000gp/T/tmpu2DGV9.tmp
info : Adding PackageReference for package 'Microsoft.EntityFrameworkCore.Sqlite' into project
'/Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/ToDoApp.csproj'.
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.s
qlite/index.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.sq
lite/index.json 239ms
info : Restoring packages for /Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/ToDoApp.cs
proj...
info : Package 'Microsoft.EntityFrameworkCore.Sqlite' is compatible with all the specified fra
meworks in project '/Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/ToDoApp.csproj'.
info : PackageReference for package 'Microsoft.EntityFrameworkCore.Sqlite' version '5.0.10' ad
ded to file '/Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/ToDoApp.csproj'.
info : Committing restore...
info : Writing assets file to disk. Path: /Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoAp
p/obj/project.assets.json
log  : Restored /Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/ToDoApp.csproj (in 392 m
s).
Maliks-MacBook-Air:ToDoApp swijaya$
```





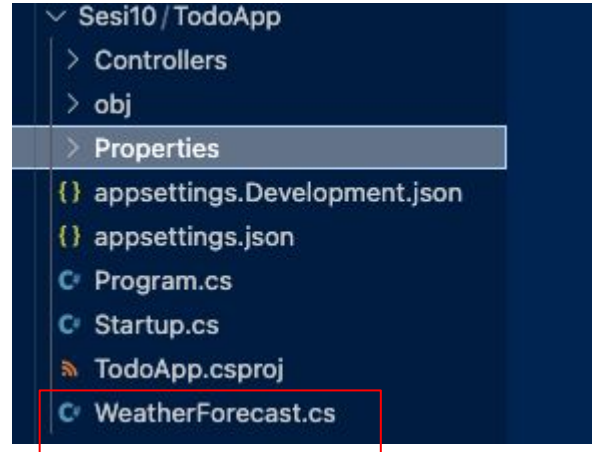
```
Maliks-MacBook-Air:TodoApp swijaya$ dotnet add package Microsoft.EntityFrameworkCore.Tools
```

```
Determining projects to restore...
Writing /var/folders/_1/jhrtgbls3m305qftdq4y0b580000gp/T/tmpShdmuf.tmp
info : Adding PackageReference for package 'Microsoft.EntityFrameworkCore.Tools' into project
'/Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/ToDoApp.csproj'.
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.t
ools/index.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.to
ols/index.json 1177ms
info : Restoring packages for /Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/ToDoApp.cs
proj...
info : Package 'Microsoft.EntityFrameworkCore.Tools' is compatible with all the specified fram
eworks in project '/Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/ToDoApp.csproj'.
info : PackageReference for package 'Microsoft.EntityFrameworkCore.Tools' version '5.0.10' add
ed to file '/Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/ToDoApp.csproj'.
info : Committing restore...
info : Generating MSBuild file /Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/obj/ToDoA
pp.csproj.nuget.g.props.
info : Writing assets file to disk. Path: /Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoA
pp/obj/project.assets.json
log  : Restored /Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp/ToDoApp.csproj (in 370 m
s).
```



Nah di sesi kali ini kita akan lakukan semua dalam VS Code dimana berbeda dengan cara-cara sebelumnya, disini kita akan explore dan create API dengan CLI.

Sekarang cek pada Folder Project ToDo :



Dapat dilihat dari struktur Project diatas jika SUDAH ada template default yang di generate oleh .NET Core untuk kita, yaitu WeatherForecast.cs

Selanjutnya kita bisa delete file ini dan kita gantikan dengan File Project kita.

## BEST PRACTICE : RESTFUL API - Sesi 10

# CONTROLLER FIRST

Hal Pertama yang harus kita lakukan adalah Buat Controller =TodoController.cs

```
1  using System.Threading.Tasks;
2  using Microsoft.AspNetCore.Mvc;
3  using Microsoft.EntityFrameworkCore;
4
5  namespace TodoApp.Controllers
6  {
7      [Route("api/[controller]")] // We define the routing that our controller going to use
8      [ApiController] // We need to specify the type of the controller to let .Net core know
9      public class TodoController : ControllerBase
10     {
11         [Route("TestRun")] // define the routing for this action
12         [HttpGet]
13
14         public ActionResult TestRun()
15         {
16             return Ok("success");
17         }
18     }
19 }
```





Coba jalankan project ini : `dotnet run`

```
Maliks-MacBook-Air:ToDoApp swijaya$ dotnet run
Building...
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoApp
```

URL Access : <https://localhost:5001/api/todo/testrun>



Seperti yang dilihat dari test api kita mendapat respons sukses.

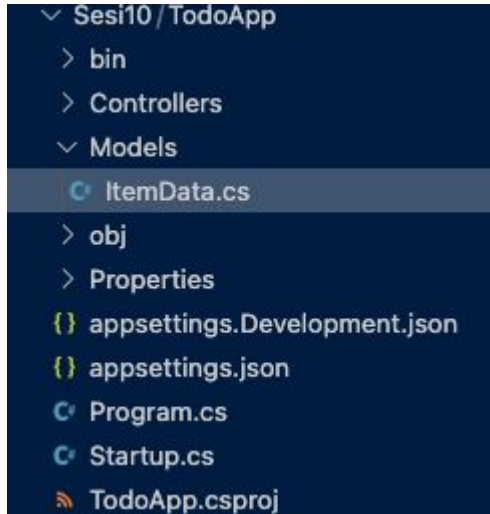
## BEST PRACTICE : RESTFUL API - Sesi 10

### MODEL

Masih Ingat dengan sesi sebelumnya pada bagian model ?

Ya kita akan implementasi hal yang sama, seperti :

1. Pada Folder Project ToDo : Tambahkan Folder Models

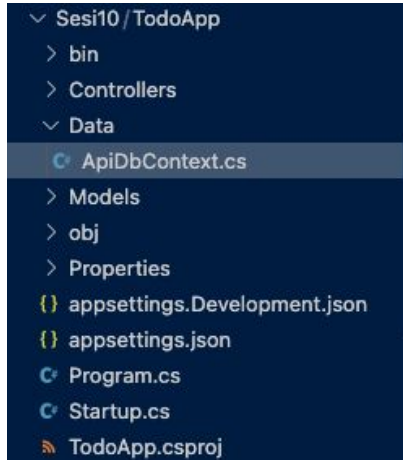


## 2. Tambahkan File ItemData.cs

```
1 namespace TodoApp.Models
2 {
3     public class ItemData
4     {
5         public int Id { get; set; }
6         public string Title { get; set; }
7         public string Description { get; set; }
8         public bool Done { get; set; }
9     }
10 }
```

Perlu Di Pahami :

Ketika kita sudah menambahkan model, kita memerlukan ApiDbContext.cs , didalam Folder Data di Project kita.



ApiDbContext.cs

```
1  using Microsoft.EntityFrameworkCore;
2  using TodoApp.Models;
3
4  namespace TodoApp.Data
5  {
6      public class ApiDbContext : DbContext
7      {
8          public virtual DbSet<ItemData> Items {get;set;}
9
10         public ApiDbContext(DbContextOptions<ApiDbContext> options)
11             : base(options)
12         {
13
14         }
15     }
16 }
```



HACKTIV8

Lalu kita perlu menyesuaikan string connection didalam appsetting.json.

```
1  {
2    "ConnectionStrings": {
3      "DefaultConnection": "DataSource=app.db;Cache=Shared"
4    },
5    "Logging": {
6      "LogLevel": {
7        "Default": "Information",
8        "Microsoft": "Warning",
9        "Microsoft.Hosting.Lifetime": "Information"
10     }
11   },
12   "AllowedHosts": "*"
13 }
```

Nah selanjutnya adalah update startup.cs agar kita bisa utilize DbContext App didalam Aplikasi kita.

Selanjutnya buka class startup dan tambahkan potongan kode berikut :

```
27     public void ConfigureServices(IServiceCollection services)
28     {
29         services.AddDbContext<ApiDbContext>(options =>
30             options.UseSqlite(
31                 Configuration.GetConnectionString("DefaultConnection")
32             ));
```

Nah selanjutnya kita tambahkan middleware DbContext yang kita butuhkan untuk menambahkan initial migrasi untuk buat database.

```
Maliks-MacBook-Air:TodoApp swijaya$ dotnet ef migrations add "Initial Migrations"
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
```

```
Maliks-MacBook-Air:TodoApp swijaya$ dotnet ef database update
Build started...
Build succeeded.
Done.
Maliks-MacBook-Air:TodoApp swijaya$
```



Pada tahap ini setelah update database selesai, kita bisa lihat akan ada folder baru namanya migrations yang terdiri dari script C# yang akan bertanggung jawab pada pembuatan database dan item table.

Kita bisa verifikasi pada database yang telah dibuat sejak kita lihat app.db file di direktori folder todo yang sama dengan yang kita lihat pada browser SQLite untuk verifikasi table yang sudah dibuat berhasil.

Sekarang kita akan melanjutkan build TodoController dan di-koneksikan ke ApiDbContext.



```

1  using System.Threading.Tasks;
2  using Microsoft.AspNetCore.Mvc;
3  using Microsoft.EntityFrameworkCore;
4  using TodoApp.Data;
5  using TodoApp.Models;
6
7  namespace TodoApp.Controllers
8  {
9      [Route("api/[controller]")] // api/todo
10     [ApiController]
11     public class TodoController : ControllerBase
12     {
13         private readonly ApiDbContext _context;
14
15         public TodoController(ApiDbContext context)
16         {
17             _context = context;
18         }
19
20         [HttpGet]
21         public async Task<IActionResult> GetItems()
22         {
23             var items = await _context.Items.ToListAsync();
24             return Ok(items);
25         }
26
27         [HttpPost]
28         public async Task<IActionResult> CreateItem(ItemData data)
29         {
30             if(ModelState.IsValid)
31             {
32                 await _context.Items.AddAsync(data);
33                 await _context.SaveChangesAsync();
34
35                 return CreatedAtAction("GetItem", new {data.Id}, data);
36             }
37
38             return new JsonResult("Something went wrong") {StatusCode = 500};
39         }
40     }

```



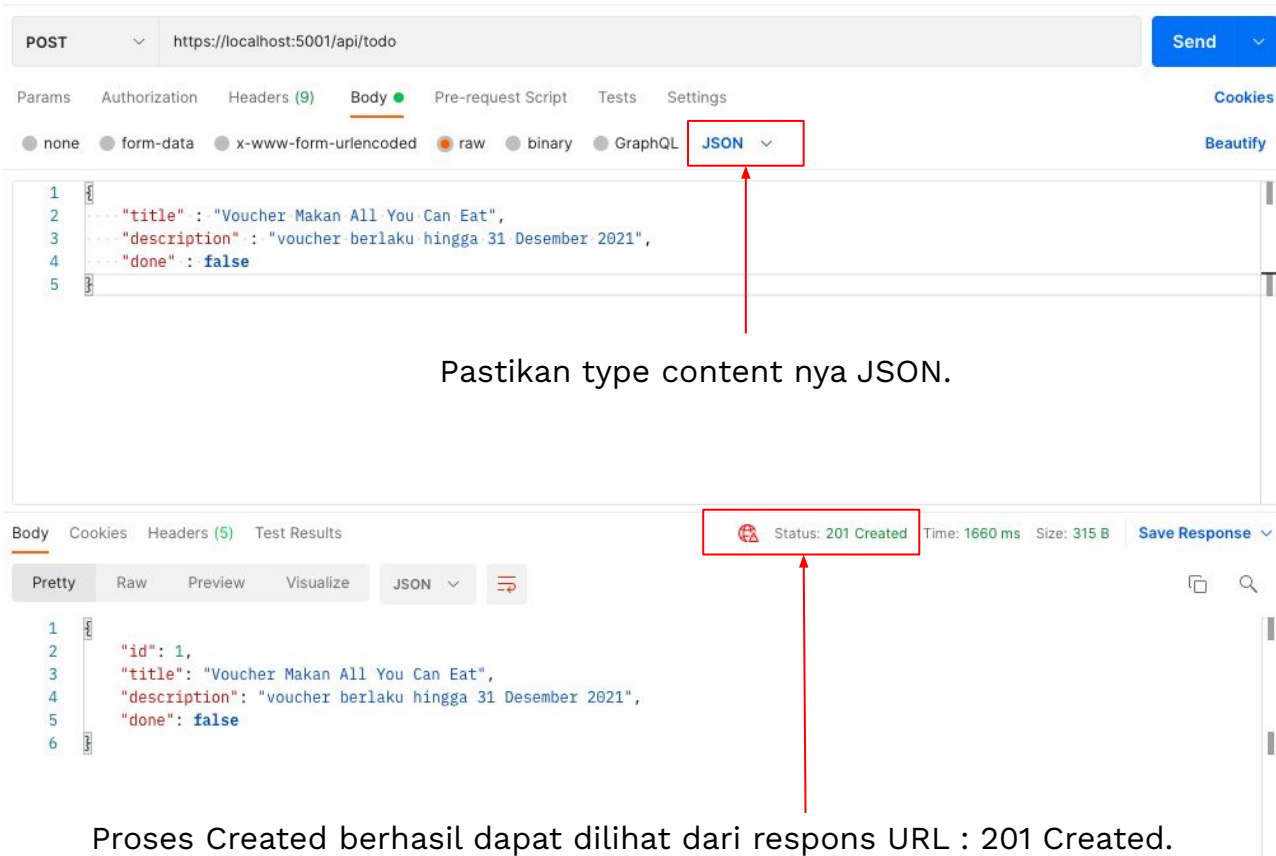
```

41 [HttpGet("{id}")]
42 public async Task<IActionResult> GetItem(int id)
43 {
44     var item = await _context.Items.FirstOrDefaultAsync(x => x.Id == id);
45
46     if(item == null)
47         return NotFound();
48
49     return Ok(item);
50 }
51
52 [HttpPut("{id}")]
53 public async Task<IActionResult> UpdateItem(int id, ItemData item)
54 {
55     if(id != item.Id)
56         return BadRequest();
57
58     var existItem = await _context.Items.FirstOrDefaultAsync(x => x.Id == id);
59
60     if(existItem == null)
61         return NotFound();
62
63     existItem.Title = item.Title;
64     existItem.Description = item.Description;
65     existItem.Done = item.Done;
66
67     // Implement the changes on the database level
68     await _context.SaveChangesAsync();
69
70     return NoContent();
71 }
72
73 [HttpDelete("{id}")]
74 public async Task<IActionResult> DeleteItem(int id)
75 {
76     var existItem = await _context.Items.FirstOrDefaultAsync(x => x.Id == id);
77
78     if(existItem == null)
79         return NotFound();
80
81     _context.Items.Remove(existItem);
82     await _context.SaveChangesAsync();
83
84     return Ok(existItem);
85 }
86 }
87

```



Lakukan Test setiap method yang sudah kita buat :



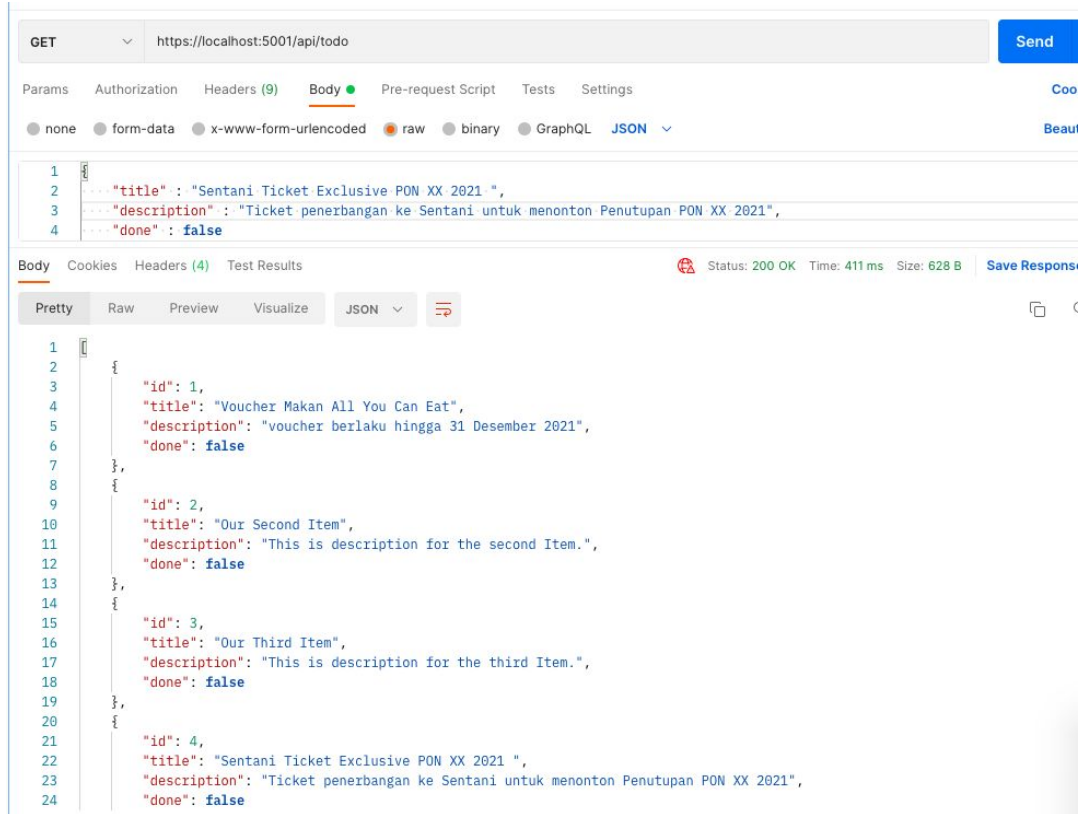
The screenshot shows a REST client interface with a POST request to `https://localhost:5001/api/todo`. The request body is a JSON object: `{ "title": "Voucher Makan All You Can Eat", "description": "voucher berlaku hingga 31 Desember 2021", "done": false }`. The response status is `201 Created`, with a time of `1660 ms` and a size of `315 B`. The response body is a JSON object: `{ "id": 1, "title": "Voucher Makan All You Can Eat", "description": "voucher berlaku hingga 31 Desember 2021", "done": false }`. Red arrows point to the `JSON` dropdown in the request body tab and the `Status: 201 Created` in the response status bar.

Pastikan type content nya JSON.

Proses Created berhasil dapat dilihat dari respons URL : 201 Created.

Tambahkan 3 Item Lagi pada method POST ini.

## Cek Item Todo gunakan method [GET]



GET <https://localhost:5001/api/todo> [Send](#)

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings [Coo](#)

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL [JSON](#) [Beau](#)

```
1 {
2   "title": "Sentani Ticket Exclusive PON XX 2021 ",
3   "description": "Ticket penerbangan ke Sentani untuk menonton Penutupan PON XX 2021",
4   "done": false
5 }
```

Body Cookies Headers (4) Test Results [Status: 200 OK](#) [Time: 411 ms](#) [Size: 628 B](#) [Save Respons](#)

[Pretty](#) [Raw](#) [Preview](#) [Visualize](#) [JSON](#) [v](#) [v](#)

```
1 {
2   {
3     "id": 1,
4     "title": "Voucher Makan All You Can Eat",
5     "description": "voucher berlaku hingga 31 Desember 2021",
6     "done": false
7   },
8   {
9     "id": 2,
10    "title": "Our Second Item",
11    "description": "This is description for the second Item.",
12    "done": false
13  },
14  {
15    "id": 3,
16    "title": "Our Third Item",
17    "description": "This is description for the third Item.",
18    "done": false
19  },
20  {
21    "id": 4,
22    "title": "Sentani Ticket Exclusive PON XX 2021 ",
23    "description": "Ticket penerbangan ke Sentani untuk menonton Penutupan PON XX 2021",
24    "done": false
25  }
26 }
```

Lalu bagaimana jika kita ingin menampilkan 1 item saja ?

Ya kita bisa tampilkan dengan memanggil id item nya, tapi perlu diketahui ini bisa dilakukan karena kita telah lebih dulu set id pada TodoController.cs

C# TodoController.cs ×

SourceCode > Sesi10 > TodoApp > Controllers > C# TodoController.cs

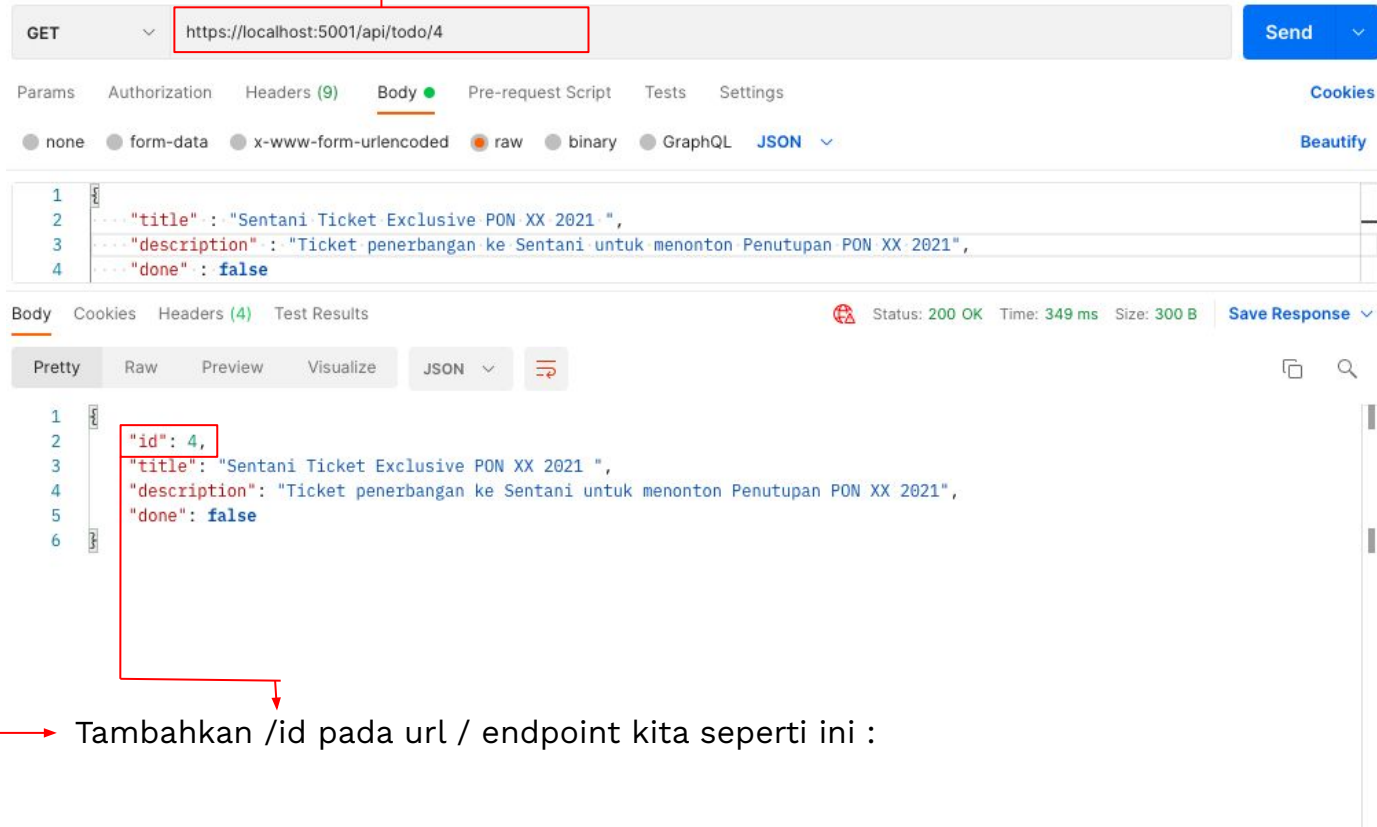
```
40
41 [HttpGet("{id}")]
42 public async Task<IActionResult> GetItem(int id)
43 {
44     var item = await _context.Items.FirstOrDefaultAsync(x => x.Id ==
45
46     if(item == null)
47         return NotFound();
48
49     return Ok(item);
50 }
```



HACKTIV8



Cek Item Todo gunakan method [GET] dan memanggil spesifik item based on ID



The screenshot shows a REST client interface with a GET request to `https://localhost:5001/api/todo/4`. The request body is a JSON object: `{ "title": "Sentani Ticket Exclusive PON XX 2021 ", "description": "Ticket penerbangan ke Sentani untuk menonton Penutupan PON XX 2021", "done": false }`. The response status is 200 OK, and the response body is a JSON object: `{ "id": 4, "title": "Sentani Ticket Exclusive PON XX 2021 ", "description": "Ticket penerbangan ke Sentani untuk menonton Penutupan PON XX 2021", "done": false }`. A red box highlights the `"id": 4,` in the response, and a red arrow points from it to the text below.

Tambahkan /id pada url / endpoint kita seperti ini :

Lalu bagaimana jika kita ingin mengubah 1 item dari data yang ada ?

Ya kita bisa EDIT dengan memanggil id item nya juga, hal ini bisa dilakukan karena kita telah lebih dulu set id pada TodoController.cs



**HACKTIV8**

C# TodoController.cs ×

SourceCode > Sesi10 > TodoApp > Controllers > C# TodoController.cs

```
52     [HttpPut("{id}")]
53     public async Task<IActionResult> UpdateItem(int id, ItemData item)
54     {
55         if(id != item.Id)
56             return BadRequest();
57
58         var existItem = await _context.Items.FirstOrDefaultAsync(x => x.Id == id);
59
60         if(existItem == null)
61             return NotFound();
62
63         existItem.Title = item.Title;
64         existItem.Description = item.Description;
65         existItem.Done = item.Done;
```



HACKTIV8

EDIT Item Todo gunakan method [PUT] dan memanggil spesifik item based on ID = 2.

Before

```
8 {
9   {
10    "id": 2,
11    "title": "Our Second Item",
12    "description": "This is description for the second item.",
13    "done": false
14  }
15 }
```

After

https://localhost:5001/api/todo/2

PUT https://localhost:5001/api/todo/2

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": 2,
3   "title": "Beli Merchandise PON XX 2021",
4   "description": "Kaos, Mug, Kipas, Tumbler",
5   "done": false
6 }
```

Body Cookies Headers (2) Test Results

Status: 204 No Content Time: 113 ms Size: 81 B Save Response

Pretty Raw Preview Visualize Text

1

## Cek Item Todo gunakan method [GET]

https://localhost:5001/api/todo

GET https://localhost:5001/api/todo

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (4) Test Results

Status: 200 OK Time: 1662 ms Size: 627 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "title": "Voucher Makan All You Can Eat",
5     "description": "voucher berlaku hingga 31 Desember 2021",
6     "done": false
7   },
8   {
9     "id": 2,
10    "title": "Beli Merchandise PON XX 2021 ",
11    "description": "Kaos, Mug, Kipas, Tumbler",
12    "done": false
13  },
14  {
15    "id": 3,
16    "title": "Our Third Item",
17    "description": "This is description for the third Item.",
18    "done": false
19  },
20  {
21    "id": 4,
22    "title": "Sentani Ticket Exclusive PON XX 2021 ",
23    "description": "Ticket penerbangan ke Sentani untuk menonton Penutupan PON XX 2021",
24    "done": false
25  }
26 ]
```

Sampai disini kita berhasil menjalankan UPDATE / EDIT dengan Method PUT.

Ada 1 Operasi lagi yang harus kita lakukan yaitu DELETE.

Perhatikan pada TodoController.js

```
C# TodoController.cs x
SourceCode > Sesi10 > TodoApp > Controllers > C# TodoController.cs

73     [HttpDelete("{id}")]
74     public async Task<IActionResult> DeleteItem(int id)
75     {
76         var existItem = await _context.Items.FirstOrDefaultAsync(x => x.I
77
78         if(existItem == null)
79             return NotFound();
80
81         _context.Items.Remove(existItem);
82         await _context.SaveChangesAsync();
83
84         return Ok(existItem);
85     }
86 }
87 }
```

Pada case ini kita akan hapus itemToDo dengan id = 3



BEFORE

https://localhost:5001/api/todo/3


Save


DELETE  https://localhost:5001/api/todo/3

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (4) Test Results  Status: 200 OK Time: 149 ms Size: 627 B

Pretty Raw Preview Visualize JSON  

```
1 {
2   {
3     "id": 1,
4     "title": "Voucher Makan All You Can Eat",
5     "description": "voucher berlaku hingga 31 Desember 2021",
6     "done": false
7   },
8   {
9     "id": 2,
10    "title": "Beli Merchandise PON XX 2021 ",
11    "description": "Kaos, Mug, Kipas, Tumbler",
12    "done": false
13  },
14  {
15    "id": 3,
16    "title": "Our Third Item",
17    "description": "This is description for the third Item.",
18    "done": false
19  },
20  {
21    "id": 4,
22    "title": "Sentani Ticket Exclusive PON XX 2021 ",
```



HACKTIV8



Ketika sudah berhasil, system akan mengembalikan URL Response 200 OK

AFTER

The screenshot displays a REST client interface. At the top, the URL `https://localhost:5001/api/todo/3` is entered. Below the URL bar, the HTTP method is set to **DELETE**. The **Body** tab is selected, showing the message "This request does not have a body". Below the body tab, various content types are listed: `none` (selected), `form-data`, `x-www-form-urlencoded`, `raw`, `binary`, and `GraphQL`. The **Send** button is visible. Below the request configuration, the **Body** tab of the response is selected. A red box highlights the response status bar, which shows: **Status: 200 OK**, **Time: 448 ms**, **Size: 250 B**, and a **Save Response** button. The response body is displayed in **JSON** format, showing the following data:

```
1 {
2   "id": 3,
3   "title": "Our Third Item",
4   "description": "This is description for the third Item.",
5   "done": false
6 }
```

Untuk Memastikan apakah item to do ke 3 sudah dihapus , maka tampilkan semua list toDO dengan method [GET]

Dapat dilihat jika Item ToDo dengan id = 3 sudah hilang dari list ToDo kita.

https://localhost:5001/api/todo

Save

Send

GET https://localhost:5001/api/todo

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies


Body Cookies Headers (4) Test Results

Status: 200 OK Time: 54 ms Size: 524 B Save Response

Pretty Raw Preview Visualize JSON

```
1  {
2    {
3      "id": 1,
4      "title": "Voucher Makan All You Can Eat",
5      "description": "voucher berlaku hingga 31 Desember 2021",
6      "done": false
7    },
8    {
9      "id": 2,
10     "title": "Beli Merchandise PON XX 2021 ",
11     "description": "Kaos, Mug, Kipas, Tumbler",
12     "done": false
13   },
14   {
15     "id": 4,
16     "title": "Sentani Ticket Exclusive PON XX 2021 ",
17     "description": "Ticket penerbangan ke Sentani untuk menonton Penutupan PON XX 2021",
18     "done": false
19   }
20 }
```





# **ASP .NET 5<sup>+</sup> Authentication with JWT**

## BEST PRACTICE : RESTFUL API - Sesi 10

# Json Web Authentication

Pada sesi kali ini, kita akan belajar untuk mengamankan Rest API kita dengan menggunakan *JsonWebToken*. *JsonWebToken* adalah sebuah token berbentuk string panjang yang digunakan untuk pertukaran informasi antara 2 belah pihak atau client dan server.

JsonWebToken biasa digunakan pada aplikasi web yang berbasis SPA atau Single Page Application dengan alur sebagai berikut:

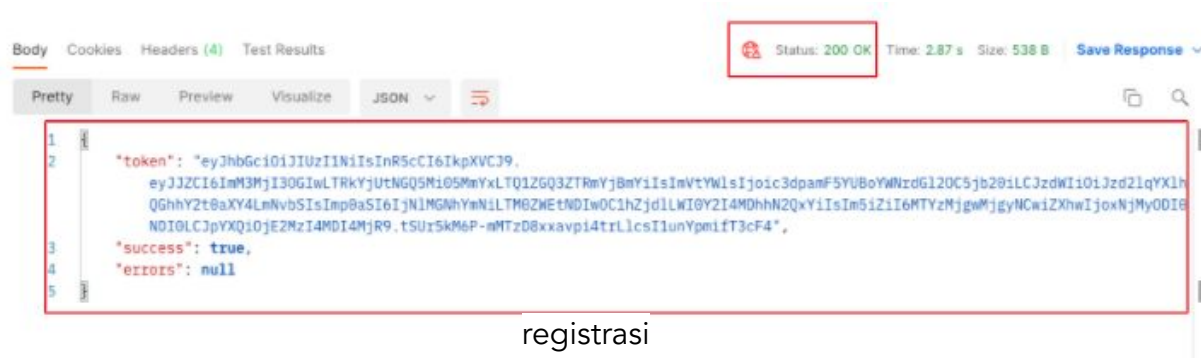
- User melakukan login melalui client, biasanya user hanya perlu menginput email dan password saja.
- Setelah user klik tombol submit, maka client akan mengirimkan data yang di input oleh user kepada server.
- Setelah data diterima oleh server, maka server akan memeriksa terlebih dahulu apakah data yang dikirimkan oleh client merupakan data yang valid atau tidak.
- Jika datanya valid, maka server akan mengirimkan data user tersebut kepada client, namun data user nya telah dijadikan sebuah token dengan menggunakan *JWT* ( *JsonWebToken* ).
- Lalu client akan menyimpan token yang dikirimkan oleh server pada local storage atau session storage, dan jika client hendak mengirimkan request kepada server kembali, maka client perlu mengirimkan token tersebut kepada server melalui request headers.
- Lalu ketika token tersebut sudah oleh server, maka server akan melakukan proses autentikasi terhadap token tersebut yang dikirimkan oleh client.

Berikut merupakan link resmi dari JWT <https://jwt.io/>

## BEST PRACTICE : RESTFUL API - Sesi 10

### Registration & Login API

Pada project kali ini kita akan belajar bagaimana menggunakan JWT pada saat registrasi user dan juga login.



Nah, untuk implement authentication kita perlu beberapa package yang harus di install, yaitu :

## 1. JwtBearer

```
dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer
```

```
Maliks-MacBook-Air:ToDoAppWithJWT swijaya$ dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer
Determining projects to restore...
Writing /var/folders/_1/jhrtgbls3m305qftdq4y0b580000gp/T/tmpJP0B9A.tmp
info : Adding PackageReference for package 'Microsoft.AspNetCore.Authentication.JwtBearer' into project '/Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoAppWithJWT/ToDoAppWithJWT.csproj'.
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.aspnetcore.authentication.jwtbearer/index.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.aspnetcore.authentication.jwtbearer/index.json 867ms
info : Restoring packages for /Users/swijaya/Downloads/C#/SourceCode/Sesi10/ToDoAppWithJWT/ToDoAppWithJWT.csproj...
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.authentication.jwtbearer/index.json
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.authentication.jwtbearer/index.json 238ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.authentication.jwtbearer/5.0.10/microsoft.aspnetcore.authentication.jwtbearer.5.0.10.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.authentication.jwtbearer/5.0.10/microsoft.aspnetcore.authentication.jwtbearer.5.0.10.nupkg 21ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.identitymodel.protocols.openidconnect/index.json
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.identitymodel.protocols.openidconnect/index.json 239ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.identitymodel.protocols.openidconnect/6.7.1/microsoft.identitymodel.protocols.openidconnect.6.7.1.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.identitymodel.protocols.openidconnect/6.7.1/microsoft.identitymodel.protocols.openidconnect.6.7.1.nupkg 25ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.identitymodel.protocols/index.json
info : GET https://api.nuget.org/v3-flatcontainer/system.identitymodel.tokens.jwt/index.json
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.identitymodel.protocols/index.json 237ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.identitymodel.protocols/6.7.1/microsoft.identitymodel.protocols.6.7.1.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.identitymodel.protocols/6.7.1/microsoft.identitymodel.protocols.6.7.1.nupkg 30ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.identitymodel.logging/index.json
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.identitymodel.tokens/index.json
info : OK https://api.nuget.org/v3-flatcontainer/system.identitymodel.tokens.jwt/index.json 945ms
info : GET https://api.nuget.org/v3-flatcontainer/system.identitymodel.tokens.jwt/6.7.1/system.identitymodel.tokens.jwt.6.7.1.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/system.identitymodel.tokens.jwt/6.7.1/system.identitymodel.tokens.jwt.6.7.1.nupkg 23ms
```





Nah, untuk implement authentication kita perlu beberapa package yang harus di install, yaitu :

## 2. Identity.EntityFrameworkCore

```
dotnet add package Microsoft.AspNetCore.Identity.EntityFrameworkCore
```

```
Maliks-MacBook-Air:TodoAppWithJWT swijaya$ dotnet add package Microsoft.AspNetCore.Identity.EntityFrameworkCore
Determining projects to restore...
Writing /var/folders/_1/jhrtgbls3m305qftdq4y0b580000qp/T/tmpUMvM0h.tmp
info : Adding PackageReference for package 'Microsoft.AspNetCore.Identity.EntityFrameworkCore' into project '/Users/swijaya/Downloads/C#/SourceCode/Sesi10/TodoAppWithJWT/TodoAppWithJWT.csproj'.
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.aspnetcore.identity.entityframeworkcore/index.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.aspnetcore.identity.entityframeworkcore/index.json 1189ms
info : Restoring packages for /Users/swijaya/Downloads/C#/SourceCode/Sesi10/TodoAppWithJWT/TodoAppWithJWT.csproj...
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.identity.entityframeworkcore/index.json
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.identity.entityframeworkcore/index.json 947ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.identity.entityframeworkcore/5.0.10/microsoft.aspnetcore.identity.entityframeworkcore.5.0.10.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.identity.entityframeworkcore/5.0.10/microsoft.aspnetcore.identity.entityframeworkcore.5.0.10.nupkg 30ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.identity.stores/index.json
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.extensions.identity.stores/index.json 869ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.identity.stores/5.0.10/microsoft.extensions.identity.stores.5.0.10.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.extensions.identity.stores/5.0.10/microsoft.extensions.identity.stores.5.0.10.nupkg 21ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.identity.core/index.json
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.dependencyinjection/index.json
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.extensions.dependencyinjection/index.json 323ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.dependencyinjection/5.0.0/microsoft.extensions.dependencyinjection.5.0.0.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.extensions.dependencyinjection/5.0.0/microsoft.extensions.dependencyinjection.5.0.0.nupkg 27ms
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.extensions.identity.core/index.json 865ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.identity.core/5.0.10/microsoft.extensions.identity.core.5.0.10.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.extensions.identity.core/5.0.10/microsoft.extensions.identity.core.5.0.10.nupkg 28ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.cryptography.keyderivation/index.json
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.cryptography.keyderivation/index.json 865ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.cryptography.keyderivation/5.0.10/microsoft.aspnetcore.cryptography.keyderivation.5.0.10.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.cryptography.keyderivation/5.0.10/microsoft.aspnetcore.cryptography.keyderivation.5.0.10.nupkg 23ms
```



Nah, untuk implement authentication kita perlu beberapa package yang harus di install, yaitu :

## 2. Identity.UI

```
dotnet add package Microsoft.AspNetCore.Identity.UI
```

```
Maliks-MacBook-Air:TodoAppWithJWT swijaya$ dotnet add package Microsoft.AspNetCore.Identity.UI
Determining projects to restore...
Writing /var/folders/_1/jhrtgbls3m305qftdq4y0b580000gp/T/tmpFwRmiG.tmp
info : Adding PackageReference for package 'Microsoft.AspNetCore.Identity.UI' into project '/Users/swijaya/Downloads/C#/SourceCode/Sesi10/TodoAppWithJWT/TodoAppWithJWT.csproj'
info : GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.aspnetcore.identity.ui/index.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.aspnetcore.identity.ui/index.json 1156ms
info : Restoring packages for /Users/swijaya/Downloads/C#/SourceCode/Sesi10/TodoAppWithJWT/TodoAppWithJWT.csproj...
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.identity.ui/index.json
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.identity.ui/index.json 940ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.identity.ui/5.0.10/microsoft.aspnetcore.identity.ui.5.0.10.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.aspnetcore.identity.ui/5.0.10/microsoft.aspnetcore.identity.ui.5.0.10.nupkg 50ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.fileproviders.embedded/index.json
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.extensions.fileproviders.embedded/index.json 862ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.fileproviders.embedded/5.0.10/microsoft.extensions.fileproviders.embedded.5.0.10.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.extensions.fileproviders.embedded/5.0.10/microsoft.extensions.fileproviders.embedded.5.0.10.nupkg 23ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.fileproviders.abstractions/index.json
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.extensions.fileproviders.abstractions/index.json 874ms
info : GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.fileproviders.abstractions/5.0.0/microsoft.extensions.fileproviders.abstractions.5.0.0.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/microsoft.extensions.fileproviders.abstractions/5.0.0/microsoft.extensions.fileproviders.abstractions.5.0.0.nupkg 41ms
info : Installed Microsoft.Extensions.FileProviders.Abstractions 5.0.0 from https://api.nuget.org/v3/index.json with content hash iuZiIz3mteEb+nsUqpGXKx2cGF+cv6gWPd5jqQI4hzqdiJ6I94ddLjKhQ0uRWl1ueHwocIw30xbSHGhQj0zjdQ==.
```



## BEST PRACTICE : RESTFUL API - Sesi 10

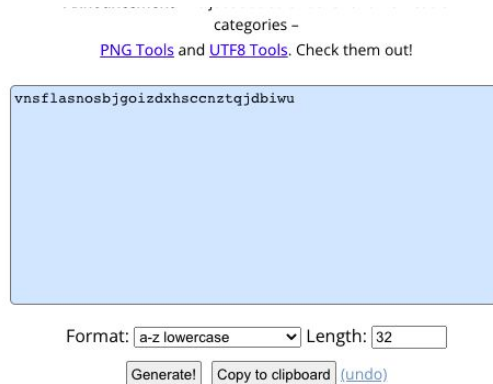
# GENERATE JWT SECRET

<https://www.browserling.com/tools/random-string>



Format:  Length:

Ubah Length pada url diatas dari 128 menjadi 32, dan klik generate.



categories -  
[PNG Tools](#) and [UTF8 Tools](#). Check them out!

vnsflasnosbjgoizdxhscenztqjdbiwu

Format:  Length:

[\(undo\)](#)

Lakukan copy pada key secret yang sudah di generated.

## BEST PRACTICE : RESTFUL API - Sesi 10

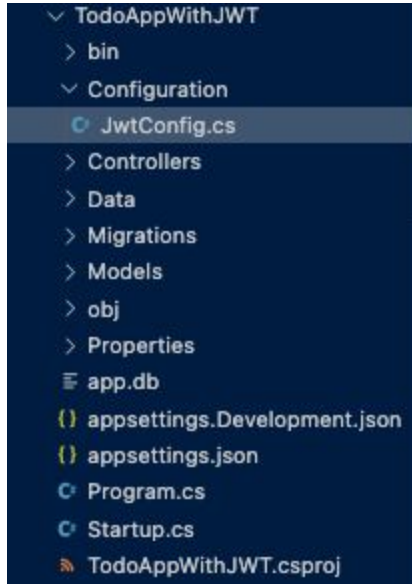
### appsettings.json

```
1  {
2    "ConnectionStrings": {
3      "DefaultConnection" : "DataSource=app.db; Cache=Shared"
4    },
5    "Logging": {
6      "LogLevel": {
7        "Default": "Information",
8        "Microsoft": "Warning",
9        "Microsoft.Hosting.Lifetime": "Information"
10     }
11  },
12  "JwtConfig": {
13    "Secret": "vnsflasnosbjgoizdxhsccnztqjdbiwu"
14  },
15  "AllowedHosts": "*"
16 }
```

Update setting pada appsettings.json dengan menambahkan JWT Settings pada line 12 & 13 dimana kita perlu JWT Secret token didalamnya yang sudah kita generate pada langkah sebelumnya.

## BEST PRACTICE : RESTFUL API - Sesi 10

### Create JwtConfig.cs



Buat Folder Configuration lalu buat file bernama JwtConfig.cs

```
1 namespace TodoAppWithJWT.Configuration
2 {
3     public class JwtConfig
4     {
5         public string Secret { get; set; }
6     }
7 }
```

## BEST PRACTICE : RESTFUL API - Sesi 10

### Startup Class

Sekarang kita harus update class startup kita, didalam method ConfigureServices kita tambahkan code dibawah ini untuk inject JwtConfiguration App kita.

```
34     public void ConfigureServices(IServiceCollection services)
35     {
36         services.Configure<JwtConfig>(Configuration.GetSection("JwtConfig"));
37     }
```



Tambahkan konfigurasi pada class startup pada konfigurasi core ASP.NET middleware dan IOC Container.

```
43     services.AddAuthentication(options => {
44         options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
45         options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
46         options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
47     })
48     .AddJwtBearer(jwt => {
49         var key = Encoding.ASCII.GetBytes(Configuration["JwtConfig:Secret"]);
50
51         jwt.SaveToken = true;
52         jwt.TokenValidationParameters = new TokenValidationParameters {
53             ValidateIssuerSigningKey = true,
54             IssuerSigningKey = new SymmetricSecurityKey(key),
55             ValidateIssuer = false,
56             ValidateAudience = false,
57             ValidateLifetime = true,
58             RequireExpirationTime = false
59         };
60     });
61
62     services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
63         .AddEntityFrameworkStores<ApiDbContext>();
64
```

Tambahkan Method Configure dengan menambahkan Authentikasi

```
85     app.UseAuthentication();
```

## BEST PRACTICE : RESTFUL API - Sesi 10

### ApiDbContext.cs

Langkah selanjutnya adalah mengupdate ApiDbContext kita dengan memanfaatkan Identity provider yang disediakan Asp.Net , untuk kita navigasi ke ApiDbContext kita di folder Data dan kita update class ApiDbContext sebagai berikut :

```
17 public class ApiDbContext : IdentityDbContext
```

dengan mewarisi dari IdentityDbContext pada DbContext, EntityFramework akan tahu bahwa kita menggunakan otentikasi dan itu akan menggunakan tabel default identity.

Untuk Menghasilkan tabel identitas di database kita, Kita perlu menyiapkan skrip migrasi dan menjalankannya. untuk melakukan itu di dalam terminal kita perlu mengetik yang berikut:

1. dotnet ef migrations add "Addig authentication to our Api"

```
Application is shutting down...
Maliks-MacBook-Air:ToDoAppWithJWT swijaya$ dotnet ef migrations add "Adding authentication to our Api"
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
```

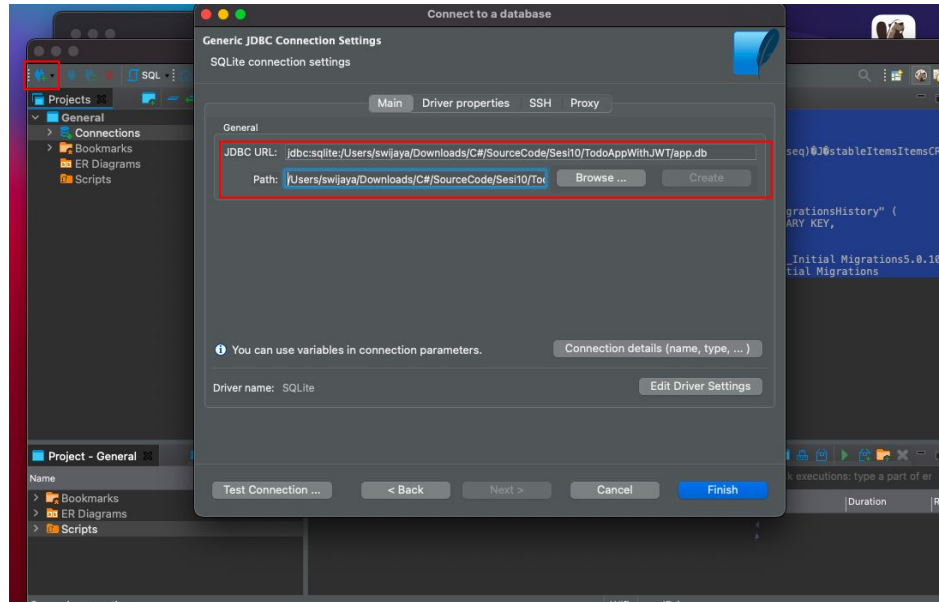




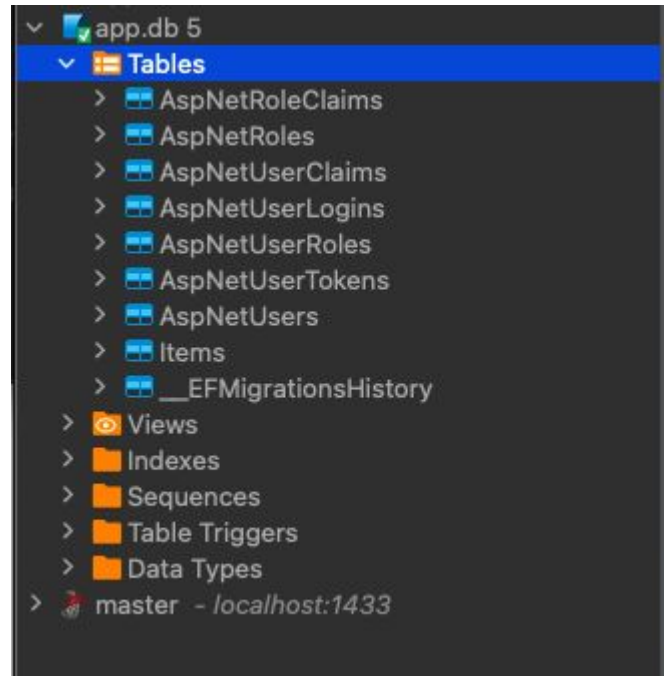
## 2. dotnet ef database update

```
Maliks-MacBook-Air:TodoAppWithJWT swijaya$ dotnet ef database update  
Build started...  
Build succeeded.  
Done.
```

Ketika migrasi kita berhasil kita bisa buka file database bernama app.db dengan Dbeaver dan kita bisa lihat Identity Table yang dibuat dengan Entity Framework.



Kita bisa lihat apa yang di generate oleh Entity Framework sewaktu kita migrasi dan update database.

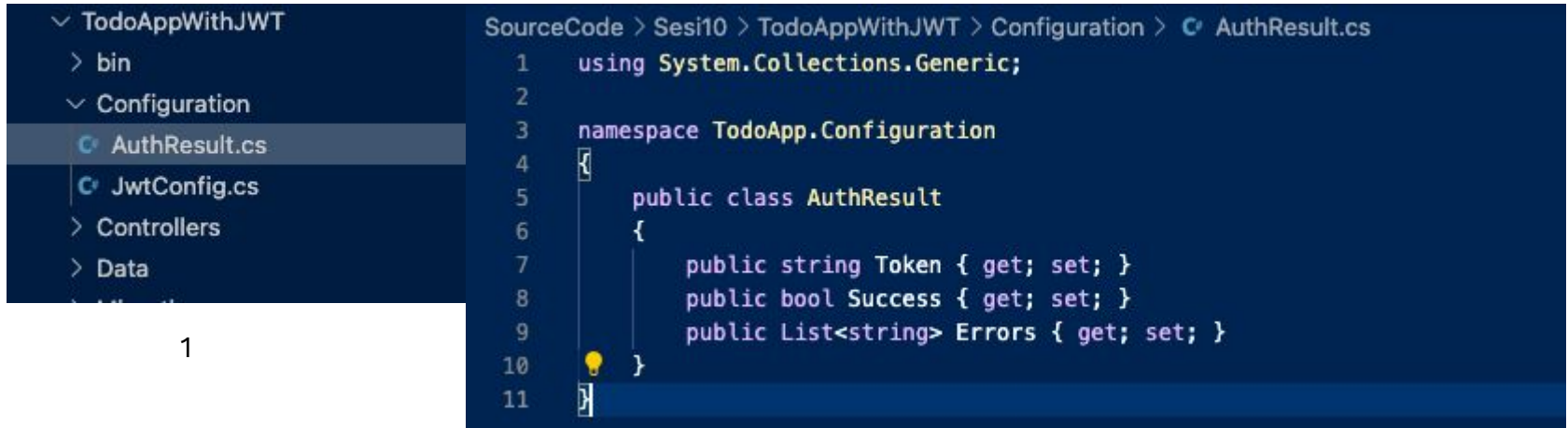


## BEST PRACTICE : RESTFUL API - Sesi 10

### DTO ( Data Transfer Objects)

Langkah Selanjutnya adalah kita harus set-up controllers dan build proses registrasi untuk User,

Buat file baru bernama : AuthResult.cs pada folder Configurations kita



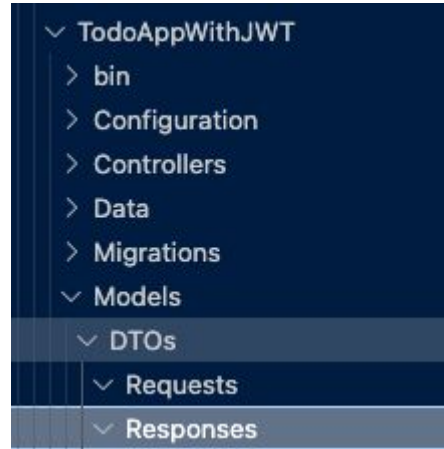
The screenshot shows a Visual Studio IDE with a dark theme. On the left, the 'Solution Explorer' pane displays the project structure for 'ToDoAppWithJWT'. The 'Configuration' folder is expanded, and 'AuthResult.cs' is highlighted. Below it, 'JwtConfig.cs' is also visible. On the right, the 'Source Code' pane shows the content of 'AuthResult.cs'. The code defines a namespace 'ToDoApp.Configuration' and a public class 'AuthResult' with three public properties: 'Token' (string), 'Success' (bool), and 'Errors' (List<string>). The code is as follows:

```
SourceCode > Sesi10 > ToDoAppWithJWT > Configuration > AuthResult.cs
1  using System.Collections.Generic;
2
3  namespace ToDoApp.Configuration
4  {
5      public class AuthResult
6      {
7          public string Token { get; set; }
8          public bool Success { get; set; }
9          public List<string> Errors { get; set; }
10     }
11 }
```

1

2

Untuk meng-organisir DTO's, Pada Folder Model buat folder bernama DTO dan buat lagi 2 folder bernama : Requests & responses.

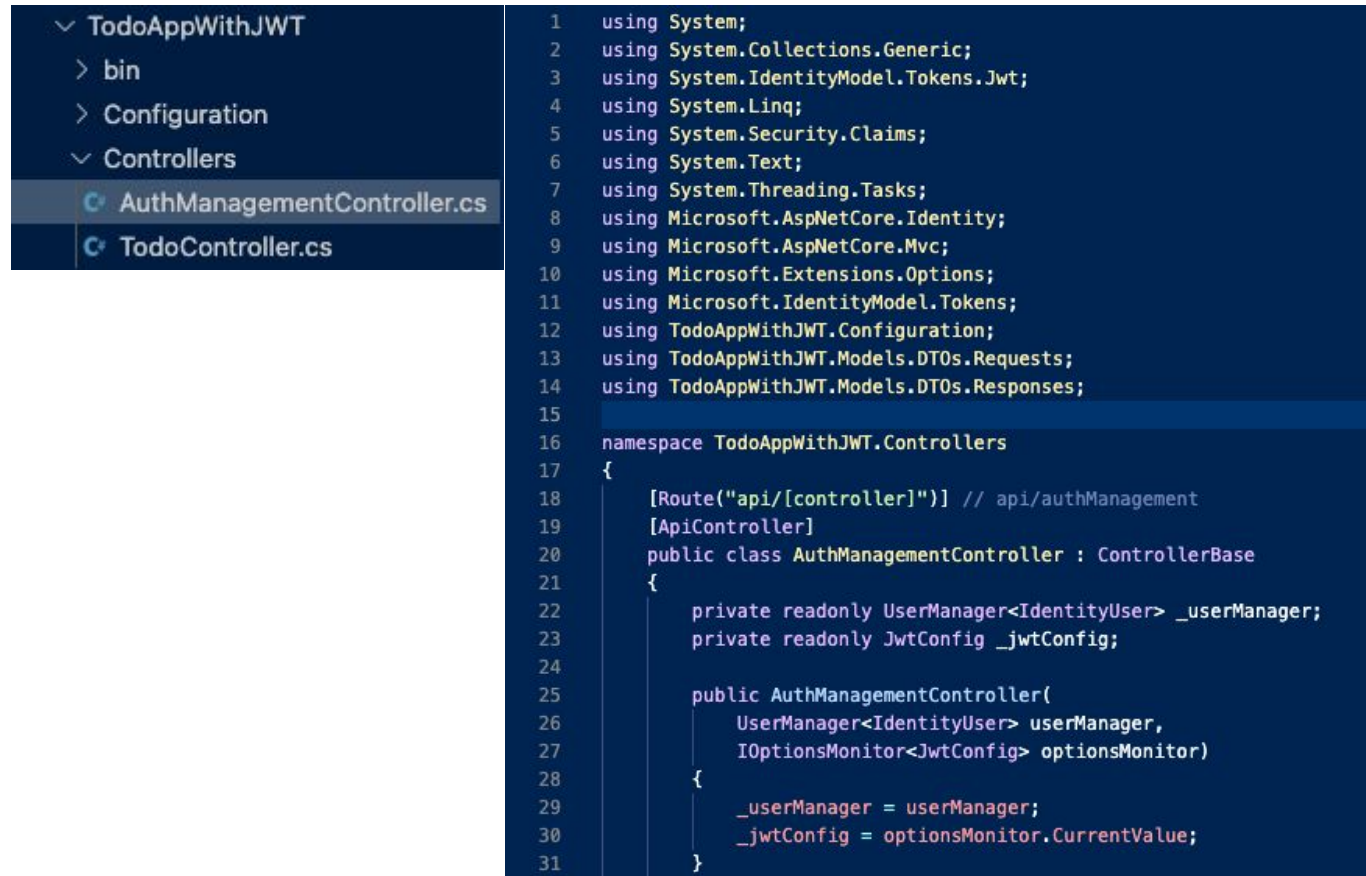


```
1  using System.ComponentModel.DataAnnotations;
2
3  namespace TodoAppWithJWT.Models.DTOs.Requests
4  {
5      public class UserRegistrationDto
6      {
7          [Required]
8          public string Username { get; set; }
9          [Required]
10         [EmailAddress]
11         public string Email { get; set; }
12         [Required]
13         public string Password { get; set; }
14     }
15 }
```



```
1  using TodoAppWithJWT.Configuration;
2
3  namespace TodoAppWithJWT.Models.DTOs.Responses
4  {
5      public class RegistrationResponse : AuthResult
6      {
7      }
8  }
9  }
```

Sekarang kita perlu menambahkan controller user registration pada folder Controller dengan membuat file class baru bernama AuthManagementController.cs



The image shows a Visual Studio interface. On the left, the 'Solution Explorer' displays a project named 'TodoAppWithJWT'. Under the 'Controllers' folder, two files are listed: 'AuthManagementController.cs' (selected) and 'TodoController.cs'. The main editor area shows the code for 'AuthManagementController.cs'. The code includes several 'using' statements for System, System.Collections.Generic, System.IdentityModel.Tokens.Jwt, System.Linq, System.Security.Claims, System.Text, System.Threading.Tasks, Microsoft.AspNetCore.Identity, Microsoft.AspNetCore.Mvc, Microsoft.Extensions.Options, Microsoft.IdentityModel.Tokens, and the project's Configuration and Models namespaces. The code defines a namespace 'TodoAppWithJWT.Controllers' containing a class 'AuthManagementController' that inherits from 'ControllerBase'. The class has two private readonly fields: '\_userManager' of type 'UserManager<IdentityUser>' and '\_jwtConfig' of type 'JwtConfig'. The constructor 'AuthManagementController' takes 'userManager' and 'optionsMonitor' as parameters and assigns them to the private fields. The class is decorated with the '[Route("api/[controller]")] // api/authManagement' attribute and the '[ApiController]' attribute.

```
1 using System;
2 using System.Collections.Generic;
3 using System.IdentityModel.Tokens.Jwt;
4 using System.Linq;
5 using System.Security.Claims;
6 using System.Text;
7 using System.Threading.Tasks;
8 using Microsoft.AspNetCore.Identity;
9 using Microsoft.AspNetCore.Mvc;
10 using Microsoft.Extensions.Options;
11 using Microsoft.IdentityModel.Tokens;
12 using TodoAppWithJWT.Configuration;
13 using TodoAppWithJWT.Models.DTOs.Requests;
14 using TodoAppWithJWT.Models.DTOs.Responses;
15
16 namespace TodoAppWithJWT.Controllers
17 {
18     [Route("api/[controller]")] // api/authManagement
19     [ApiController]
20     public class AuthManagementController : ControllerBase
21     {
22         private readonly UserManager<IdentityUser> _userManager;
23         private readonly JwtConfig _jwtConfig;
24
25         public AuthManagementController(
26             UserManager<IdentityUser> userManager,
27             IOptionsMonitor<JwtConfig> optionsMonitor)
28         {
29             _userManager = userManager;
30             _jwtConfig = optionsMonitor.CurrentValue;
31         }
32     }
33 }
```

```

32
33 [HttpPost]
34 [Route("Register")]
35 public async Task<IActionResult> Register([FromBody] UserRegistrationDto user)
36 {
37     if(ModelState.IsValid)
38     {
39         // We can utilise the model
40         var existingUser = await _userManager.FindByEmailAsync(user.Email);
41
42         if(existingUser != null)
43         {
44             return BadRequest(new RegistrationResponse(){
45                 Errors = new List<string>() {
46                     "Email already in use"
47                 },
48                 Success = false
49             });
50         }
51
52         var newUser = new IdentityUser() { Email = user.Email, UserName = user.Username};
53         var isCreated = await _userManager.CreateAsync(newUser, user.Password);
54         if(isCreated.Succeeded)
55         {
56             var jwtToken = GenerateJwtToken( newUser);
57
58             return Ok(new RegistrationResponse() {
59                 Success = true,
60                 Token = jwtToken
61             });
62         } else {
63             return BadRequest(new RegistrationResponse(){
64                 Errors = isCreated.Errors.Select(x => x.Description).ToList(),
65                 Success = false
66             });
67         }
68     }

```





```

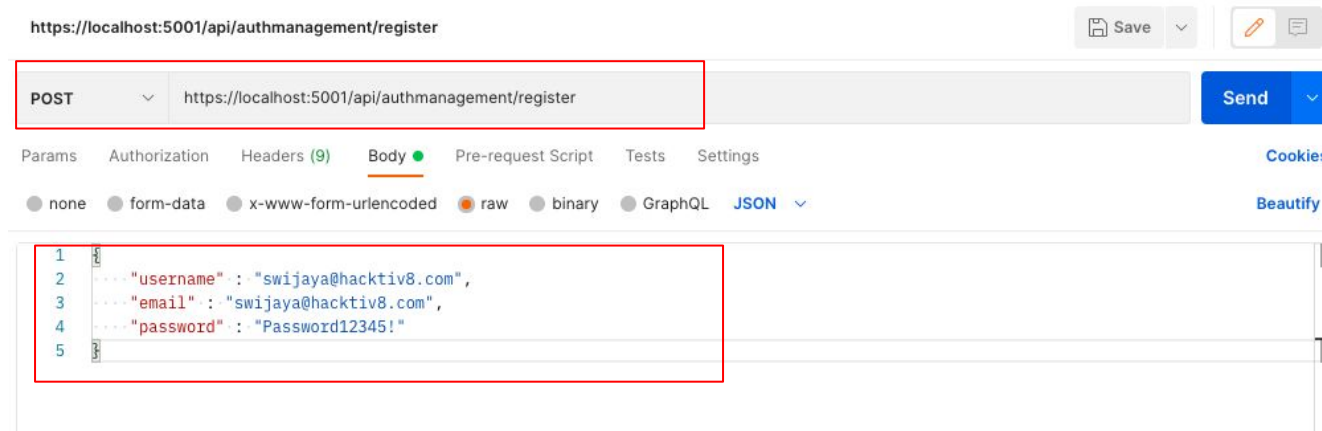
70     return BadRequest(new RegistrationResponse(){
71         Errors = new List<string>() {
72             "Invalid payload"
73         },
74         Success = false
75     });
76
77
78     private string GenerateJwtToken(IdentityUser user)
79     {
80         var jwtTokenHandler = new JwtSecurityTokenHandler();
81
82         var key = Encoding.ASCII.GetBytes(_jwtConfig.Secret);
83
84         var tokenDescriptor = new SecurityTokenDescriptor
85         {
86             Subject = new ClaimsIdentity(new []
87             {
88                 new Claim("Id", user.Id),
89                 new Claim(JwtRegisteredClaimNames.Email, user.Email),
90                 new Claim(JwtRegisteredClaimNames.Sub, user.Email),
91                 new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
92             }),
93             Expires = DateTime.UtcNow.AddHours(6),
94             SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
95         };
96
97         var token = jwtTokenHandler.CreateToken(tokenDescriptor);
98         var jwtToken = jwtTokenHandler.WriteToken(token);
99
100        return jwtToken;
101    }
102 }
103

```



Action Registrations telah selesai dibuat , sekarang kita bisa test di postman dan lihat implementasi jwt token untuk proses registrasi.

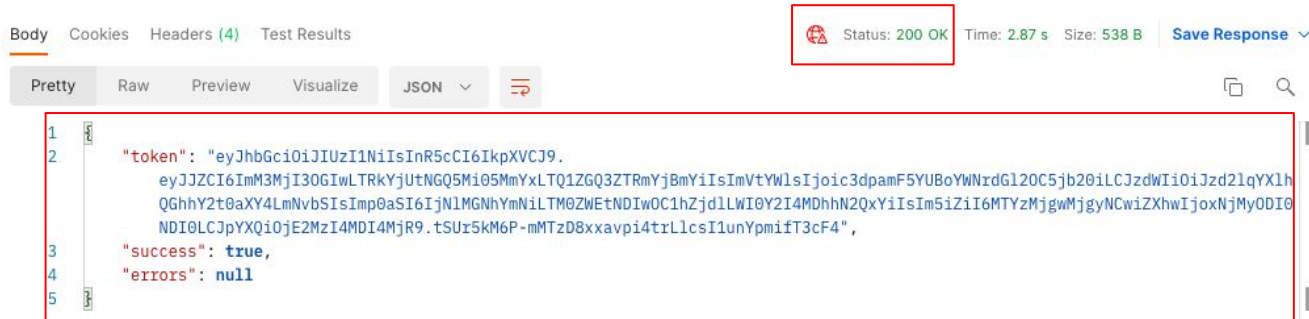
request



The image shows a Postman interface for a POST request. The URL bar at the top displays `https://localhost:5001/api/authmanagement/register`. Below the URL bar, the request method is set to **POST**. The request body is configured as **JSON** and contains the following data:

```
1 {
2   "username": "swijaya@hactiv8.com",
3   "email": "swijaya@hactiv8.com",
4   "password": "Password12345!"
5 }
```

response

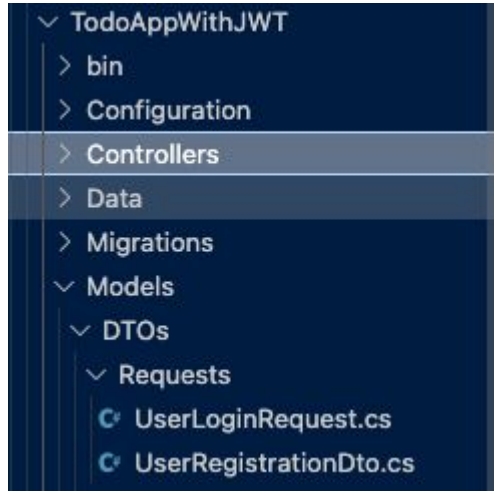


The image shows the response of the POST request in Postman. The status bar at the top indicates a **Status: 200 OK**. The response body is displayed in **JSON** format and contains the following data:

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImM3MjI3OGIwLTQ1ZGQ3ZTRmYjBmYiIsImVtYWlsIjoic3dpamF5YUBoYWNrdG12OC5jb20iLCJzdWIiOiJzd2lqYX1hQGhhY2t0aXY4LmNvbSIsImp0aSI6IjNlMGNhYmNiLT0wZWVtNDIwOC1hZjd1LWl0Y2I4MDhhN2QxYiIsIm5iZiI6MTYzMjgwMjgyNCwiZXhwIjojODI0NDI0LCJpYXQiOjE2MzI4MDI4MjR9.tSUz5kM6P-mMTzD8xxavpi4trllcsI1unYpmifT3cF4",
3   "success": true,
4   "errors": null
5 }
```

Setelah berhasil pada proses registrasi akan kita lanjutkan pembuatan action loginrequest.cs

Models/Dto/Requests/UserLoginRequest.cs



```
1 using System.ComponentModel.DataAnnotations;
2
3 namespace TodoAppWithJWT.Models.DTOs.Requests
4 {
5     public class UserLoginRequest
6     {
7         [Required]
8         [EmailAddress]
9         public string Email { get; set; }
10        [Required]
11        public string Password { get; set; }
12    }
13 }
```

Setelah berhasil pada proses registrasi akan kita lanjutkan pembuatan action login di AuthManagementController.cs

```
78 [HttpPost]
79 [Route("Login")]
80 public async Task<IActionResult> Login([FromBody] UserLoginRequest user)
81 {
82     if(ModelState.IsValid)
83     {
84         var existingUser = await _userManager.FindByEmailAsync(user.Email);
85
86         if(existingUser == null) {
87             return BadRequest(new RegistrationResponse(){
88                 Errors = new List<string>() {
89                     "Invalid login request"
90                 },
91                 Success = false
92             });
93         }
94
95         var isCorrect = await _userManager.CheckPasswordAsync(existingUser, user.Password);
96
97         if(!isCorrect) {
98             return BadRequest(new RegistrationResponse(){
99                 Errors = new List<string>() {
100                     "Invalid login request"
101                 },
102                 Success = false
103             });
104         }
105
106         var jwtToken = GenerateJwtToken(existingUser);
107
108         return Ok(new RegistrationResponse() {
109             Success = true,
110             Token = jwtToken
111         });
112     }
113
114     return BadRequest(new RegistrationResponse(){
115         Errors = new List<string>() {
116             "Invalid payload"
117         },
118         Success = false
119     });
120 }
```



Nah buka kembali postmannya dan kita eksekusi action login :

The screenshot displays the Postman interface for a REST client. At the top, the URL bar shows `https://localhost:5001/api/authmanagement/login`. Below the URL bar, the **POST** method is selected, and the same URL is entered in the request URL field. The **Body** tab is active, showing a JSON payload with the following content:

```
1 {
2   "email": "swijaya@hactiv8.com",
3   "password": "Password12345!"
4 }
```

Below the request body, the **Body** tab of the response is selected, showing a JSON response with the following content:

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJJ2ZCI6ImM3mji30GIWLRkYjUtnGQ5Mi05MmYxLTQ1ZGQ3ZTRmYjBmYiIsInVtYWlsIjoic3dpamF5YUBoYWNRdGl2OC5jb20iLCJzdWIiOiJzd2lqYXlhQGhhY2t0aXY4LmNvbSIsImp0aSI6ImIzZmZjMmUzLTJlY2ItNGU1OC1hNTI4LTE3ODJiOWYwZWY0ZSI6Im5iZiI6MTYzMjgwMzkwNSwiZXhwIjojNjMyODI1NTA1LCJpYXQiOiJlZmZlI4MDM5MDV9.tyiTwtAeFw5jf8Wk9Wuu4lyexwFSQ0GwkRSBgCkmUJE",
3   "success": true,
4   "errors": null
5 }
```

The status bar at the bottom indicates a **Status: 200 OK**, **Time: 3.37 s**, and **Size: 538 B**. The **Save Response** button is visible.



**HACTIV8**

Terakhir, Penerapan JWT adalah authorize / otorisasi dimana hanya user yang telah login dan bisa meng-akses itemData yang kita buat diawal.

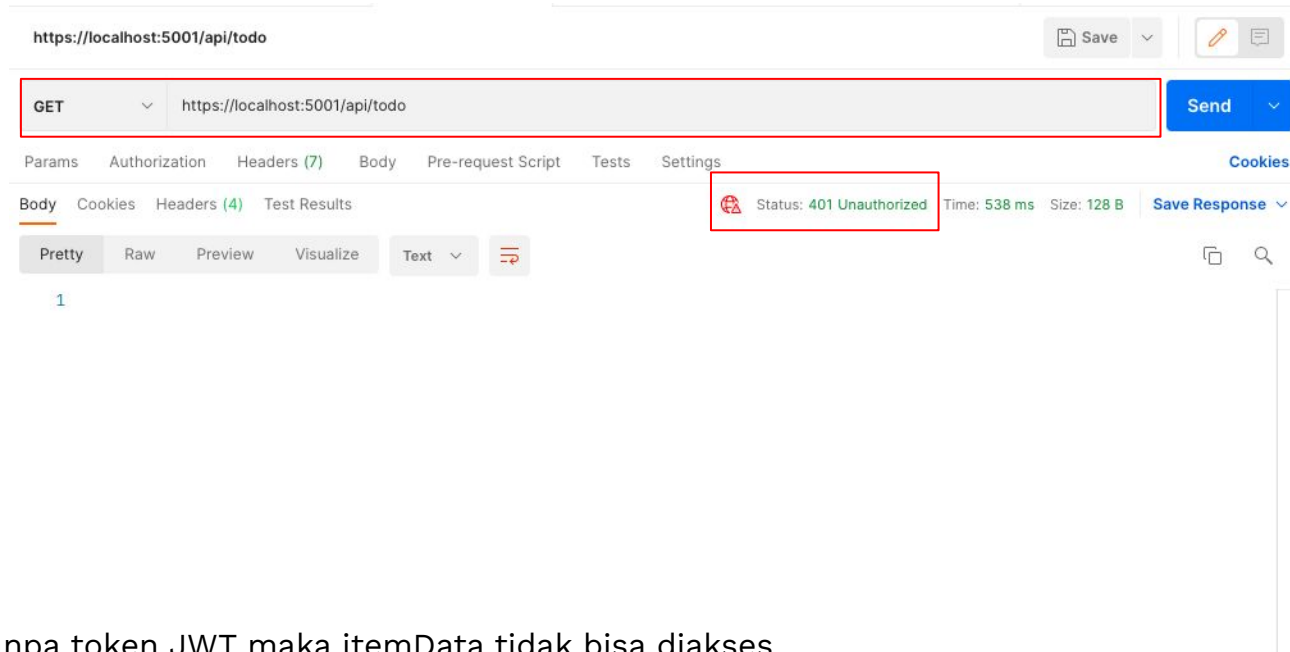
Buka file TodoController.cs dan tambahkan :

```
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
```

```
1  using System.Threading.Tasks;
2  using Microsoft.AspNetCore.Authentication.JwtBearer;
3  using Microsoft.AspNetCore.Authorization;
4  using Microsoft.AspNetCore.Mvc;
5  using Microsoft.EntityFrameworkCore;
6  using TodoAppWithJWT.Data;
7  using TodoAppWithJWT.Models;
8
9  namespace TodoAppWithJWT.Controllers
10 {
11     [Route("api/[controller]")] // api/todo
12     [ApiController]
13     [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
14     public class TodoController : ControllerBase
```



Silahkan akses [GET] api/todo kita



Nah tanpa token JWT maka itemData tidak bisa diakses,

Lalu bagaimana cara mengaksesnya ?

Pilih tab Headers dan tambahkan Authorization = Bearer {your-token}

https://localhost:5001/api/todo

GET https://localhost:5001/api/todo Send

Params Authorization **Headers (8)** Body Pre-request Script Tests Settings Cookies

<input checked="" type="checkbox"/>	Accept ⓘ	*/*	
<input checked="" type="checkbox"/>	Accept-Encoding ⓘ	gzip, deflate, br	
<input checked="" type="checkbox"/>	Connection ⓘ	keep-alive	
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...	
	Key	Value	Description

Body Cookies Headers (4) Test Results

Status: 200 OK Time: 2.57 s Size: 150 B Save Response

```
1 {
2   {
3     "id": 1,
4     "title": "Voucher Makan All You Can Eat",
5     "description": "voucher berlaku hingga 31 Desember 2021",
6     "done": false
7   },
8   {
9     "id": 2,
10    "title": "Beli Merchandise PON XX 2021 ",
11    "description": "Kaos, Mug, Kipas, Tumbler",
12    "done": false
13  },
14  {
15    "id": 4,
16    "title": "Sentani Ticket Exclusive PON XX 2021 ",
17    "description": "Ticket penerbangan ke Sentani untuk menonton Penutupan PON XX 2021",
18    "done": false
19  }
20 }
```





Good NEWS :

Sejak menggunakan .NET 5 ketika kita membuat dokumentasi webapi project Swagger yang sudah ter-integrasi dengan Aplikasi kita, nah cara melihat tampilan swagger bisa di cek pada :

`http://localhost:5000/swagger/index.html`

