# Best Practice “Create Web API: RESTFUL API”

Dalam Membuat Web API dapat menggunakan dotnet 5.0. Berikut Aplikasi yang diperlukan dalam serangkaian pembuatan web API menggunakan dotnet 5.0:

1. .NET 5.0:
   - .Net SDK 5.0.406
   - ASP.NET Core Runtime 5.0.15
   - .NET Desktop Runtime 5.0.15
   - .NET Runtime 5.0.15

   (https://dotnet.microsoft.com/en-us/download/dotnet/5.0)
2. Visual Studio Code: Aplikasi IDE/editor kode selama pembuatan web API (https://code.visualstudio.com/docs/?dv=win)
3. Insomnia: aplikasi untuk mencoba API (https://insomnia.rest/download)
4. Navicat: aplikasi untuk membuka database termasuk SQLite

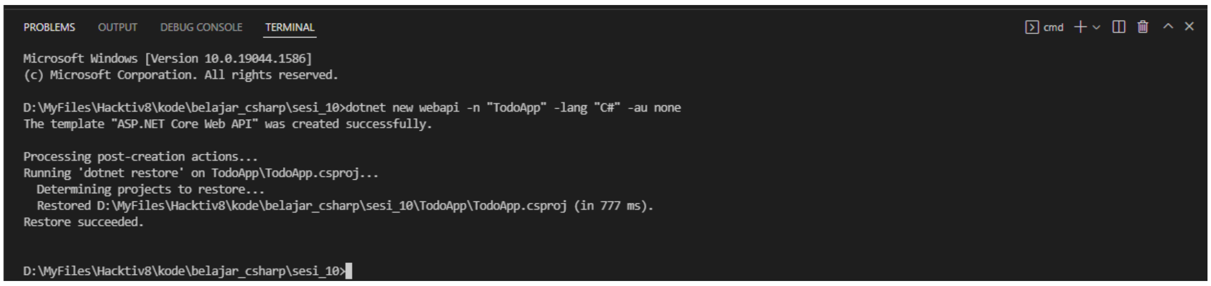
## I. Membuat dan Menjalankan ‘Web API TodoApp’ dari dotnet 5.0

1. Membuat Web API dapat dilakukan dengan menjalankan syntax pada Folder Project

Syntax:

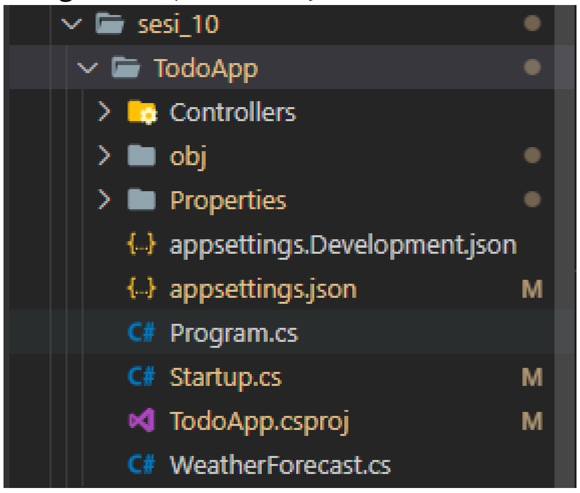
| Syntax | dotnet new webapi | -n “TodoApp” | -lang “C#” | -au none |
|---|---|---|---|---|
| Keterangan | (1) | (2) | (3) | (4) |

Keterangan:
1. Membuat project baru dotnet dengan template webapi
2. Penamaan project
3. Penggunaan bahasa C#
4. Authorization = none



```
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10>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 D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp\TodoApp.csproj (in 777 ms).
Restore succeeded.


D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10>
```

Dengan menjalankan syntax di atas akan mengenerate file sebagai berikut:

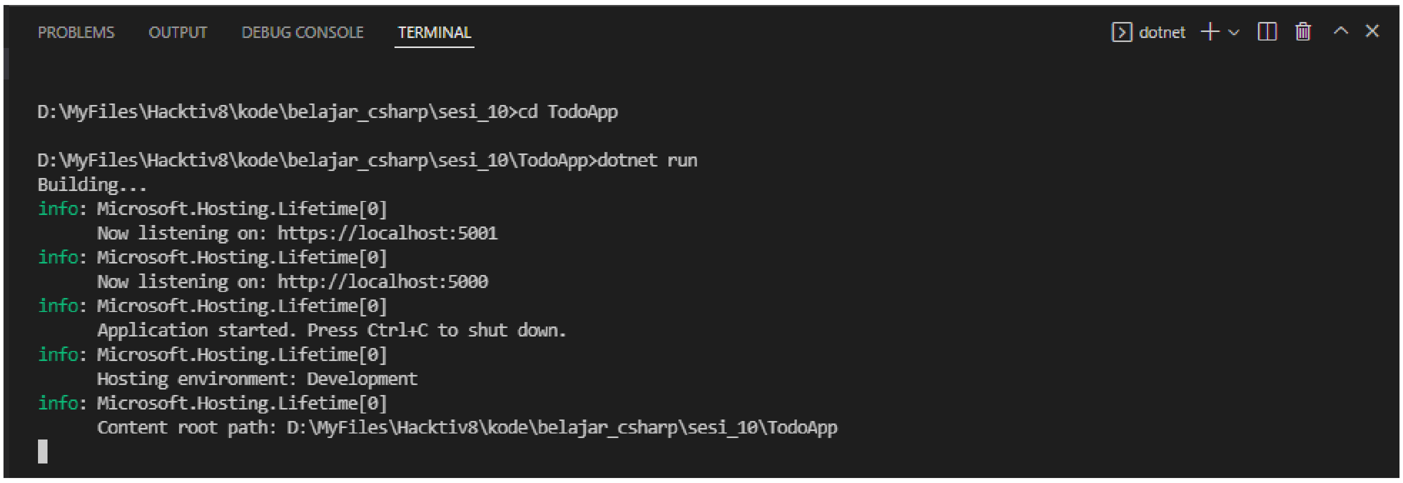


```
sesi_10
  TodoApp
    Controllers
    obj
    Properties
    appsettings.Development.json
    appsettings.json                M
    Program.cs
    Startup.cs                      M
    TodoApp.csproj                  M
    WeatherForecast.cs
```

Setelah itu masuk ke dalam folder project TodoApp (1) dan mencoba menjalankannya (2)

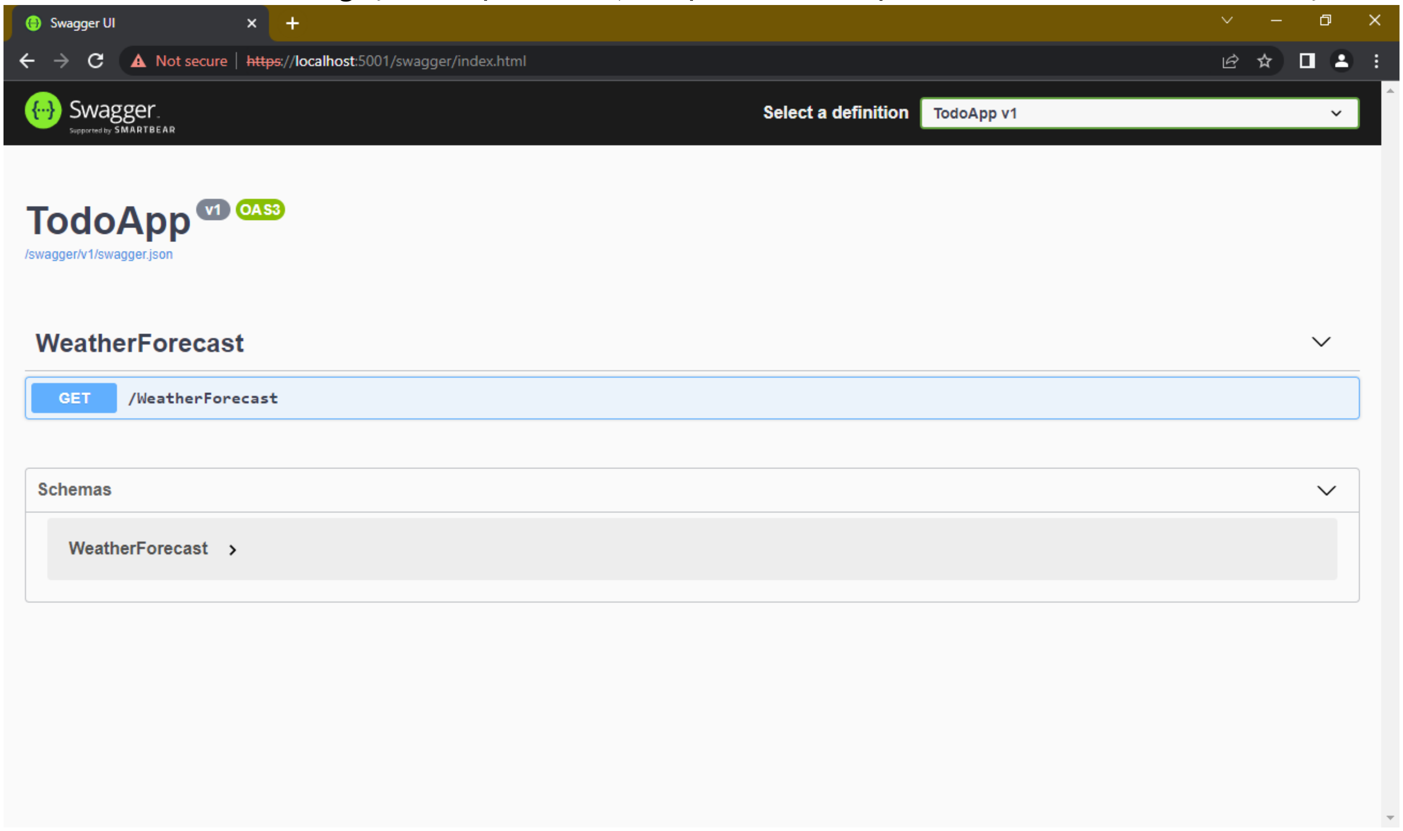
| Syntax 1 | cd TodoApp |
|---|---|

| Syntax 2 | dotnet run |
|---|---|



```
D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10>cd TodoApp

D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>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: D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp
```

Berdasarkan Hasil running syntax, aplikasi berjalan pada host ‘http://localhost:5001’ Berikut hasilnya:

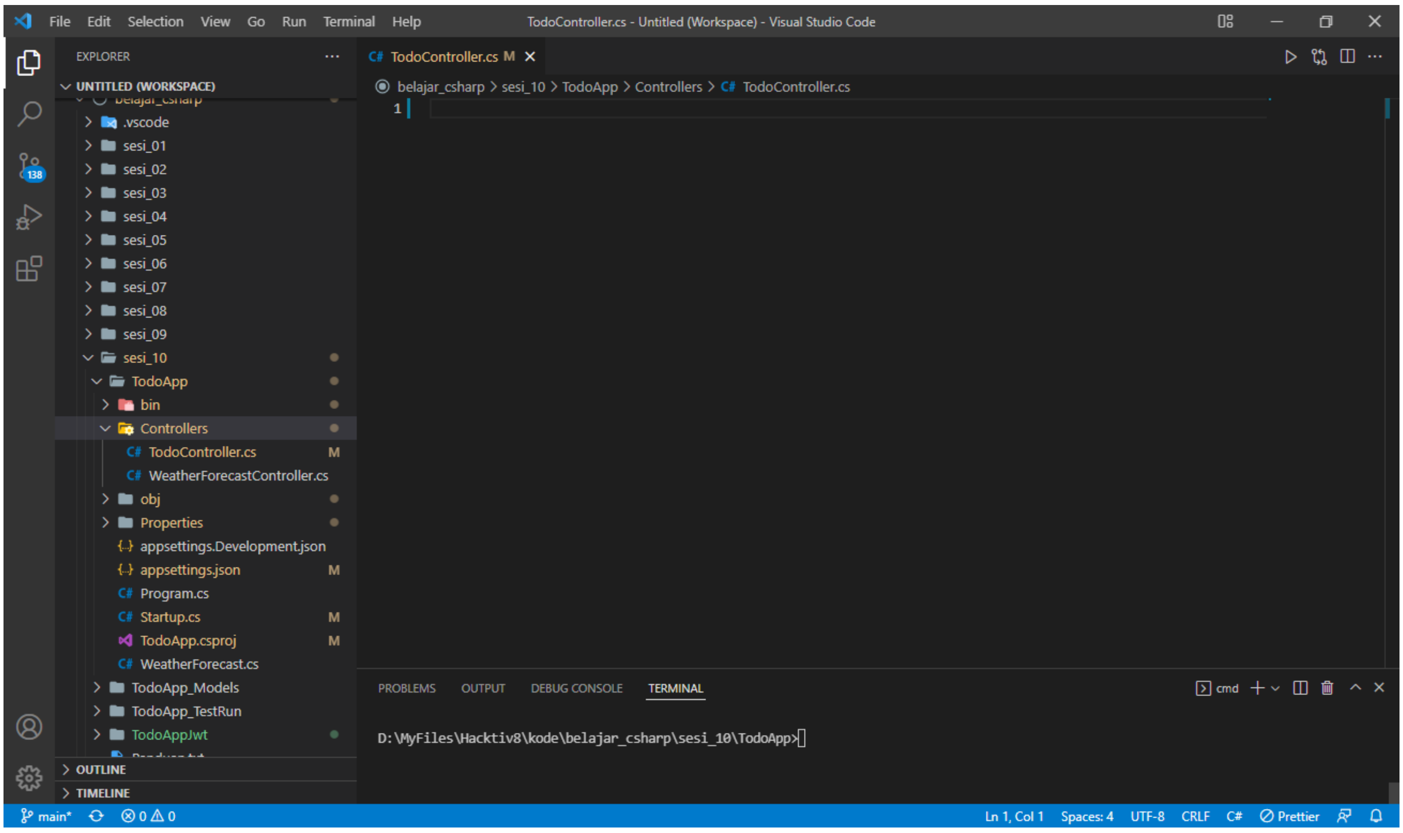


Untuk menghentikannya dapat menekan ‘Ctrl+C’. Dengan ini Project web API TodoApp sudah siap untuk dilakukan tahap selanjutnya.

## II. Menambahkan Controller Todo pada Project TodoApp sebagai TestRun

Dalam folder project TodoApp sudah terdapat folder controller maka dari itu dapat langsung menambahkan file controller pada folder ini

1. Pada ‘TodoApp>Controllers’, tambahkan ‘TodoController.cs’.



```
D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>
```

2. Lalu isi dengan kode seperti dibawah ini

```
using Microsoft.AspNetCore.Mvc;
namespace TodoApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class TodoController: ControllerBase
    {
        [Route("TestRun")]
        [HttpGet]

        public ActionResult TestRun(){
            return Ok("success");
        }
    }
}
```

---

Penjelasan Library:

| Library | `using Microsoft.AspNetCore.Mvc;` |
|---|---|
| Kode | `[Route("api/[controller]")]`<br>`[ApiController]`<br><br>`[Route("TestRun")]`<br>`[HttpGet]`<br><br>`Ok("success");` |

Berdasarkan

```
[Route("api/[controller]")]
```

Maka rute berada pada http://localhost/api/todo dimana todo ini merupakan nama controller berdasarkan ‘TodoController.cs’
Sedangkan setelah itu terdapat lagi fungsi Route

```
[Route("TestRun")]
```

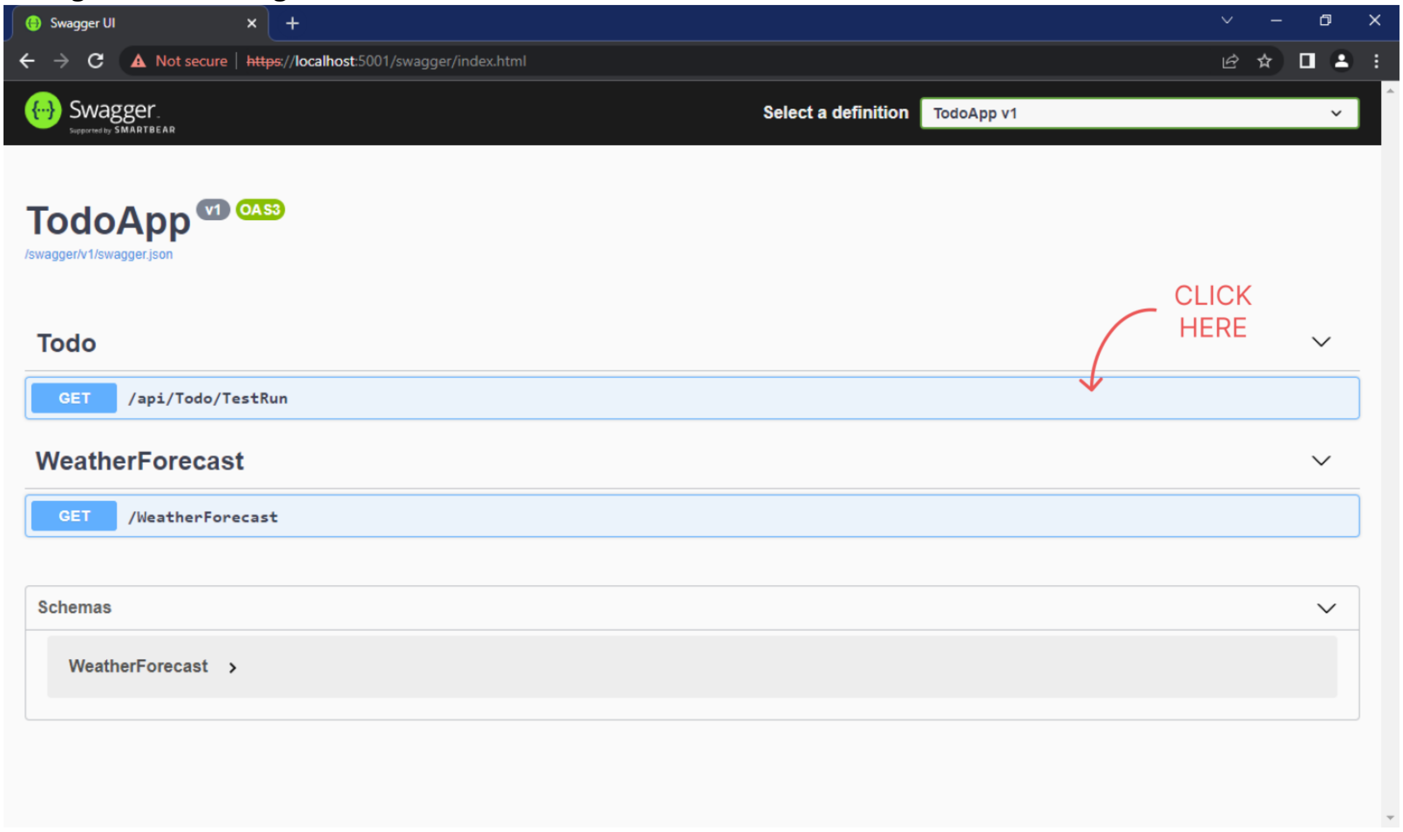
Akan mendefinisikan rute pada http://localhost/api/todo/testrun

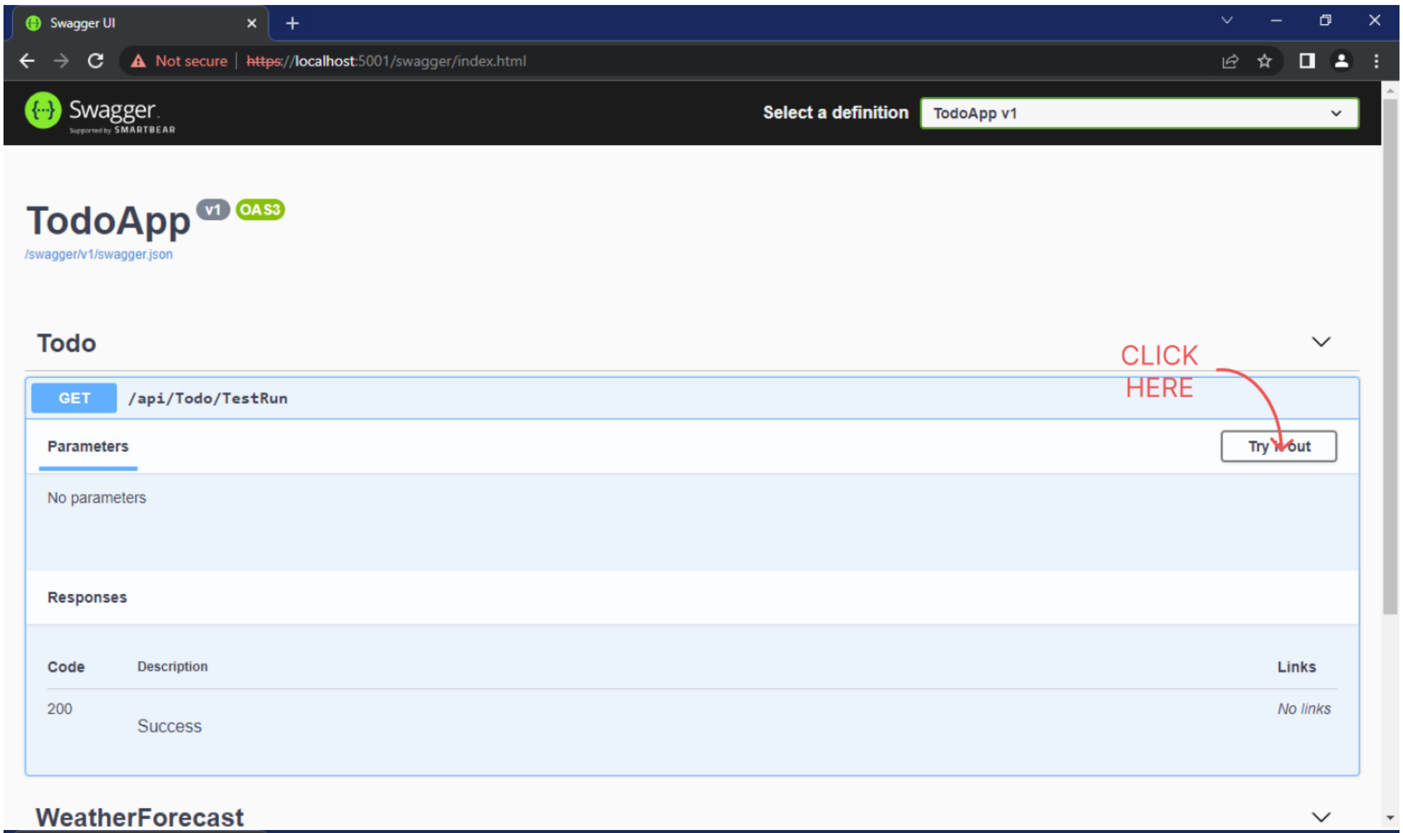
```
[HttpGet]
public ActionResult TestRun(){
      return Ok("success");
}
```

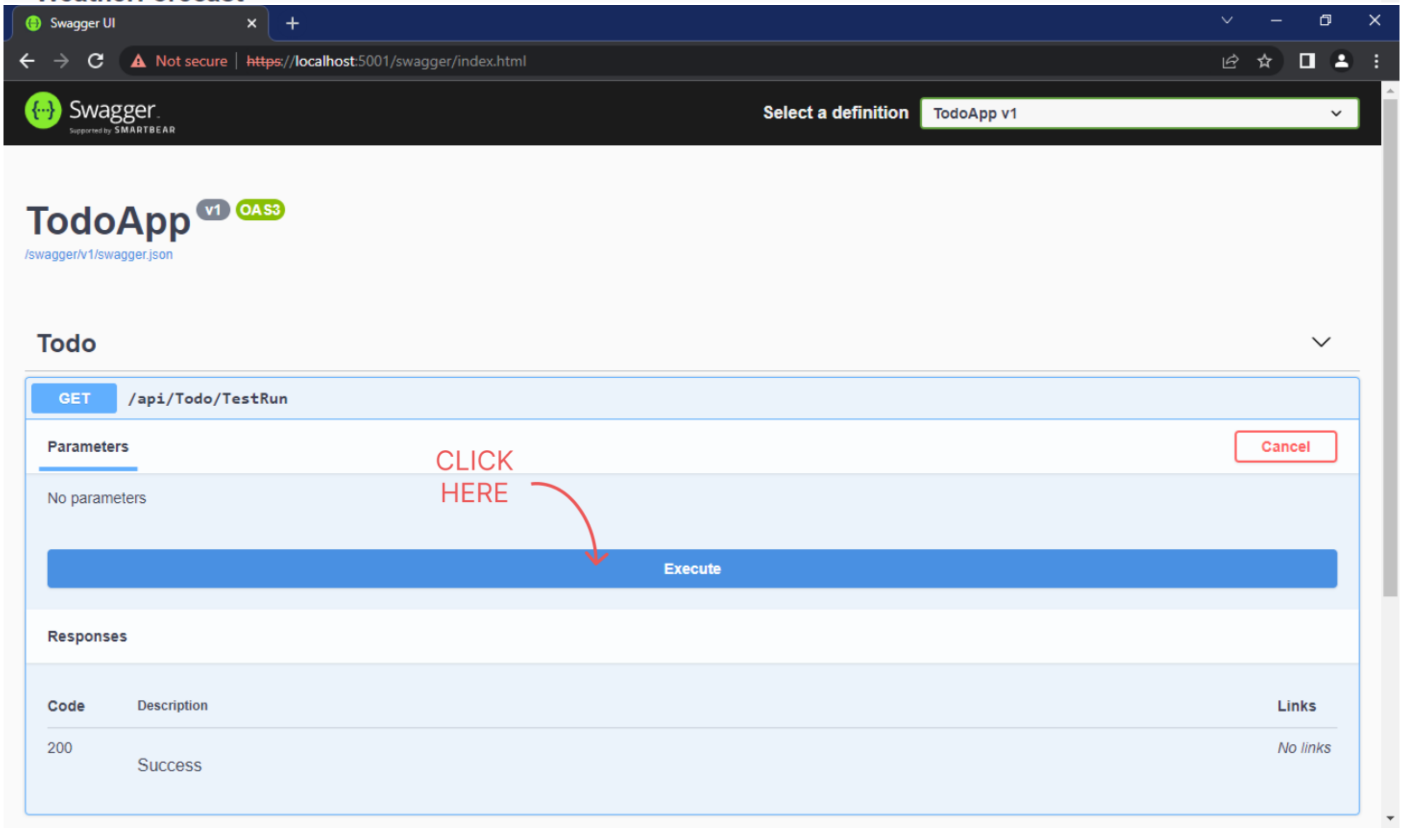
TestRun() akan berjalan pada http://localhost/api/todo/testrun dengan method ‘GET’

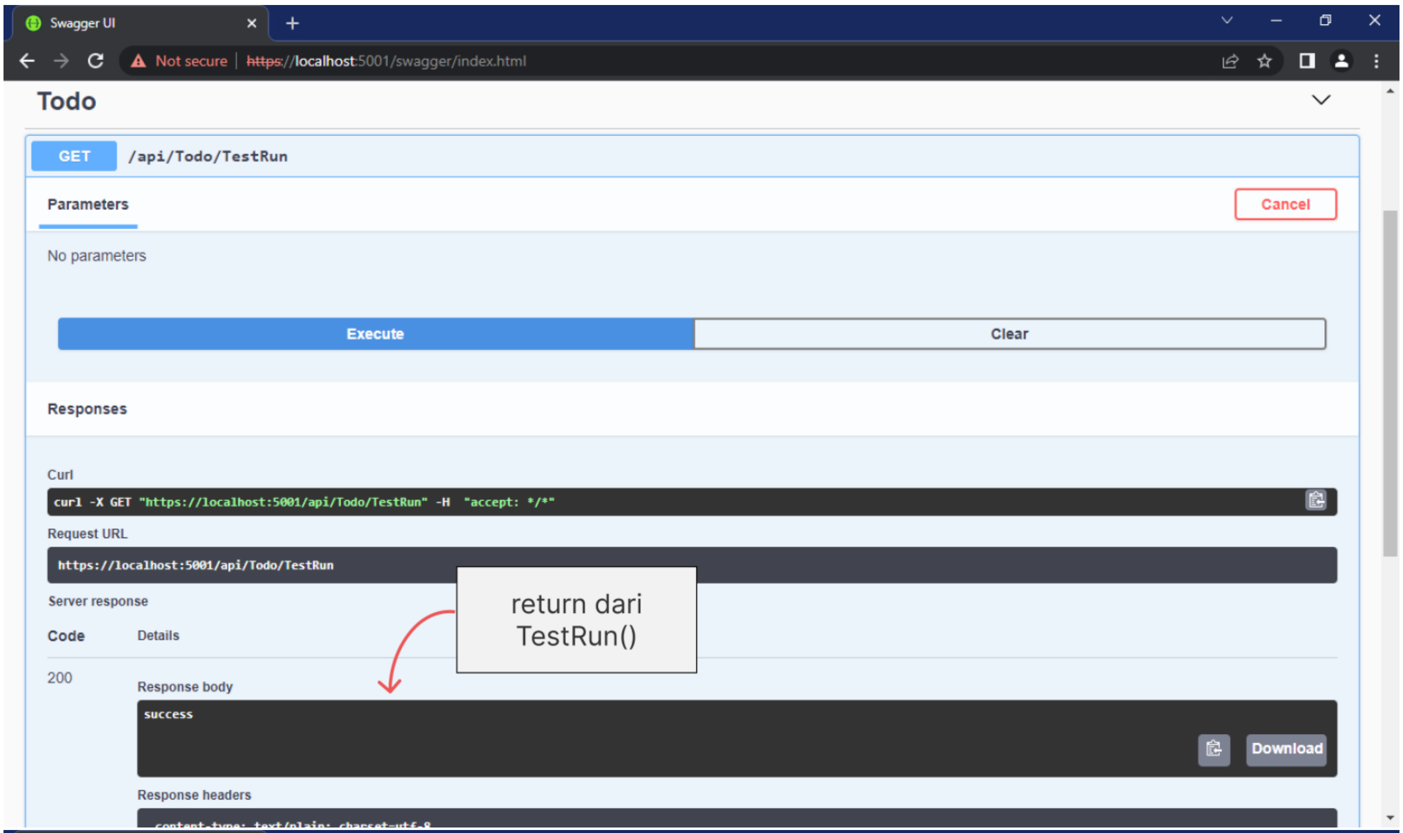
```
return Ok("success");
```

merupakan Method dari `[ApiController]`
Sebagai hasil running :




Swagger UI
Not secure https://localhost:5001/swagger/index.html
Swagger
Select a definition
TodoApp v1
TodoApp v1 OAS3
/swagger/v1/swagger.json
Todo
GET /api/Todo/TestRun
Parameters
Try it out
CLICK HERE
No parameters
Responses
Code Description Links
200 Success No links
WeatherForecast

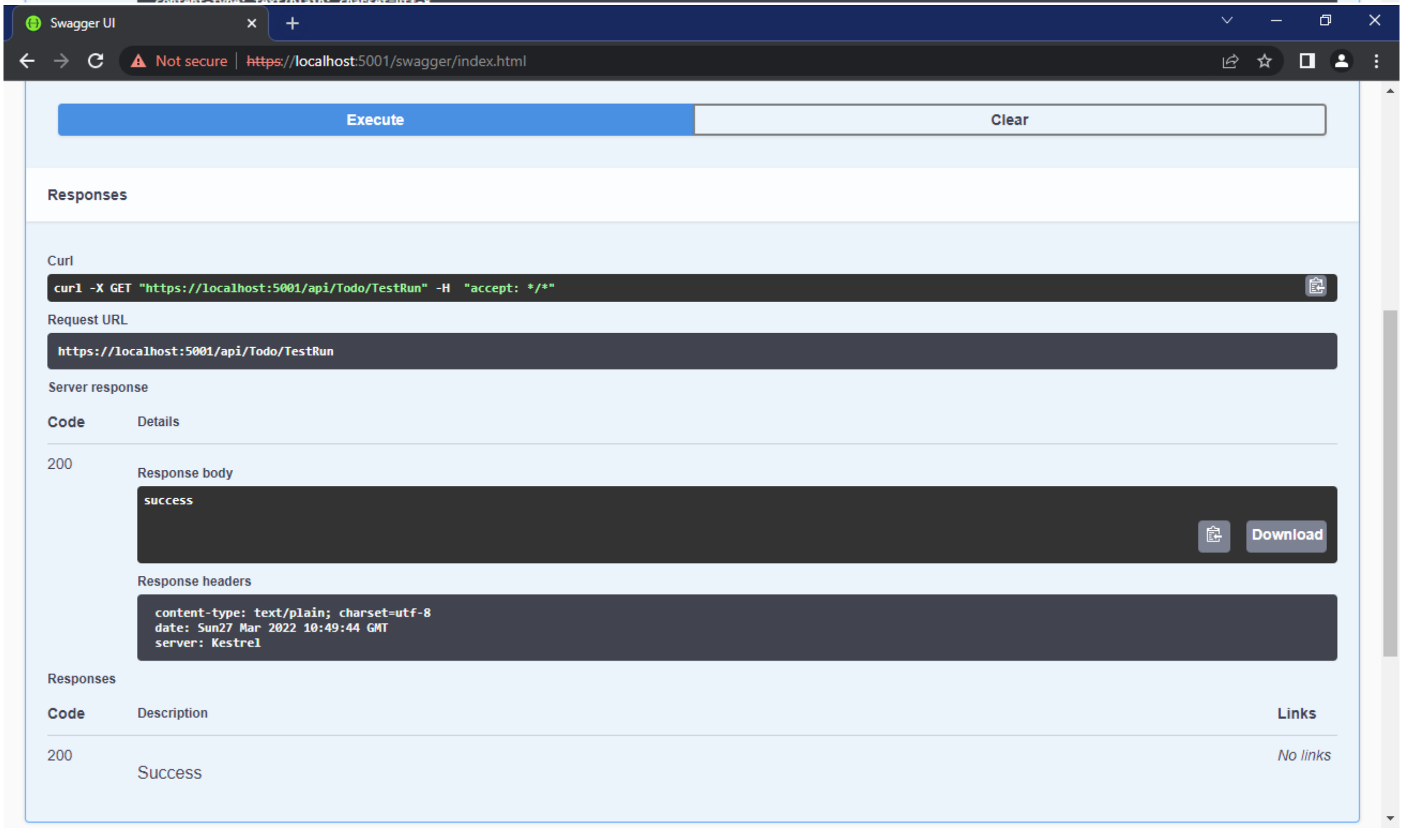

Swagger UI
Not secure https://localhost:5001/swagger/index.html
Swagger
Select a definition
TodoApp v1
TodoApp v1 OAS3
/swagger/v1/swagger.json
Todo
GET /api/Todo/TestRun
Parameters
Cancel
CLICK HERE
No parameters
Execute
Responses
Code Description Links
200 Success No links



Curl
```
curl -X GET "https://localhost:5001/api/Todo/TestRun" -H  "accept: */*"
```
Request URL
```
https://localhost:5001/api/Todo/TestRun
```
Server response

Code Details

200 Response body
```
success
```
Response headers



Curl
```
curl -X GET "https://localhost:5001/api/Todo/TestRun" -H  "accept: */*"
```
Request URL
```
https://localhost:5001/api/Todo/TestRun
```
Server response

Code Details

200 Response body
```
success
```
Response headers
```
content-type: text/plain; charset=utf-8
date: Sun27 Mar 2022 10:49:44 GMT
server: Kestrel
```
Responses

| Code | Description | Links |
|---|---|---|
| 200 | Success | *No links* |

Dengan ini TodoController berfungsi dengan baik dan siap untuk tahap selanjutnya.

## III. Migrasi dan Update Database SQLite 'Items' sebagai database 'Web API TodoApp'

Pada TodoApp ini akan menggunakan SQLite sebagai database. Di dalam database akan ada tabel Item yang berisi data:

| | |
|---|---|
| Id | INT |
| Title | STRING |
| Description | STRING |
| Done | BOOL |

Dikarenakan menggunakan SQLite dan migrations maka diperlukan package tambahan:

dotnet add package Microsoft.EntityFrameworkCore.Sqlite -v 5.0.15
dotnet add package Microsoft.EntityFrameworkCore.Tools -v 5.0.15

1. Pada 'TodoApp' tambahkan folder **'Models'**.
2. Pada 'TodoApp>Models' tambahkan file **'ItemData.cs'**, di dalam file ini definisikan `class ItemData`

```
namespace TodoApp.Models
{
    public class ItemData
    {
        public int Id {get; set;}
        public string Title {get; set;}
        public string Description {get; set;}
        public bool Done {get; set;}
    }
}
```

Langkah ini diperlukan untuk mendefinisikan class ItemData yang nantinya akan dimanfaatkan sebagai object data untuk Item

3. Pada 'TodoApp' tambahkan folder 'Data'
4. Pada 'TodoApp>Data' tambahkan file 'ApiDBContext.cs'

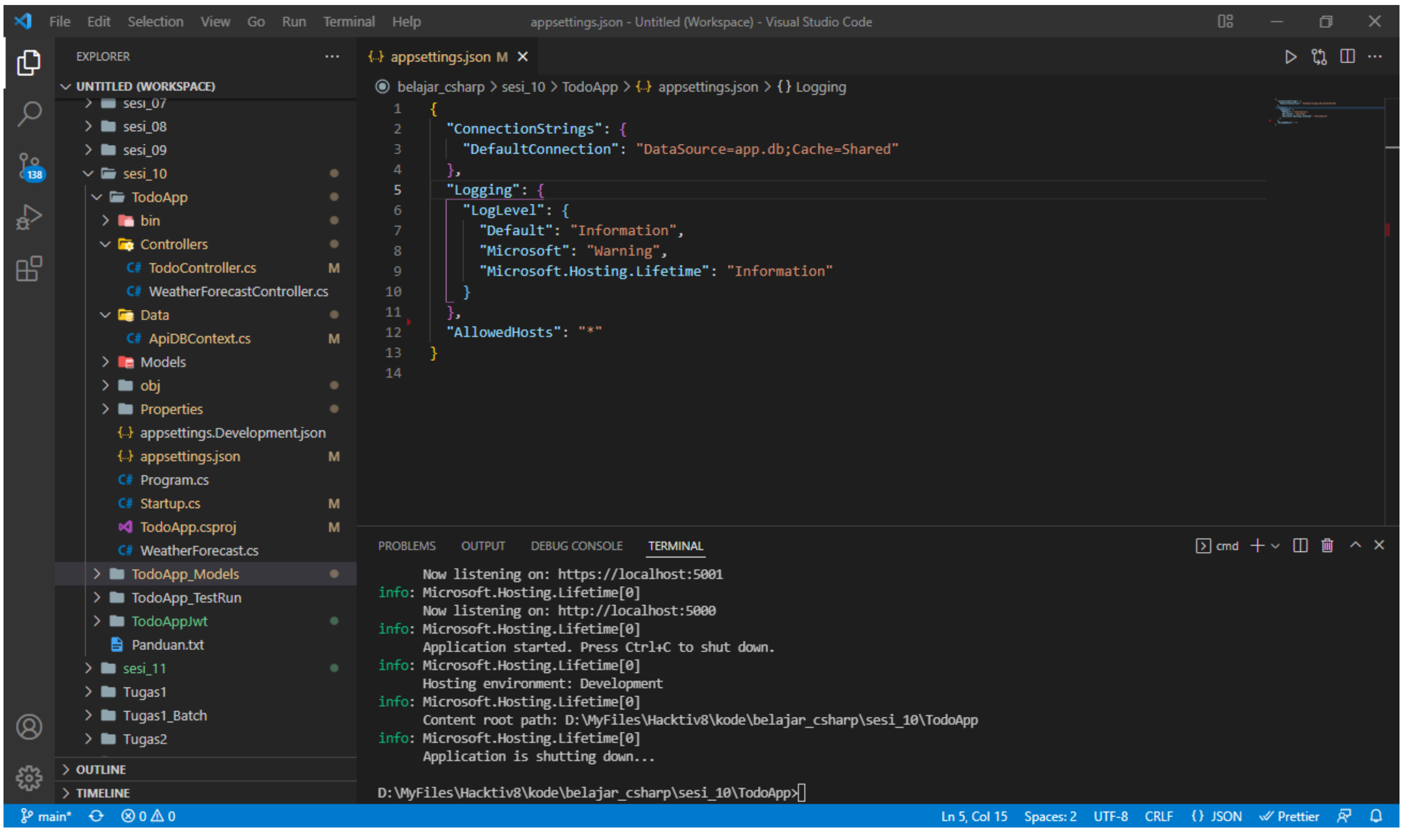
```
using Microsoft.EntityFrameworkCore;
using TodoApp.Models;

namespace TodoApp.Data
{
    public class ApiDbContext : DbContext
    {
        public virtual DbSet<ItemData> Items {get;set;}
        public ApiDbContext(DbContextOptions<ApiDbContext> options):base(options)
        {

        }
    }
}
```

- `DbContext` diambil dari `Microsoft.EntityFrameworkCore`
- `ItemData` diambil dari `TodoApp.Models`

5. Pada appsettings.json tambahkan

```
"ConnectionStrings": {
    "DefaultConnection": "DataSource=app.db;Cache=Shared"
  },
```

Jadi seperti ini :



```json
{
  "ConnectionStrings": {
    "DefaultConnection": "DataSource=app.db;Cache=Shared"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

```
      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: D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp
info: Microsoft.Hosting.Lifetime[0]
      Application is shutting down...

D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>
```

6. Pada 'startup.cs' pada bagian atas tambahkan

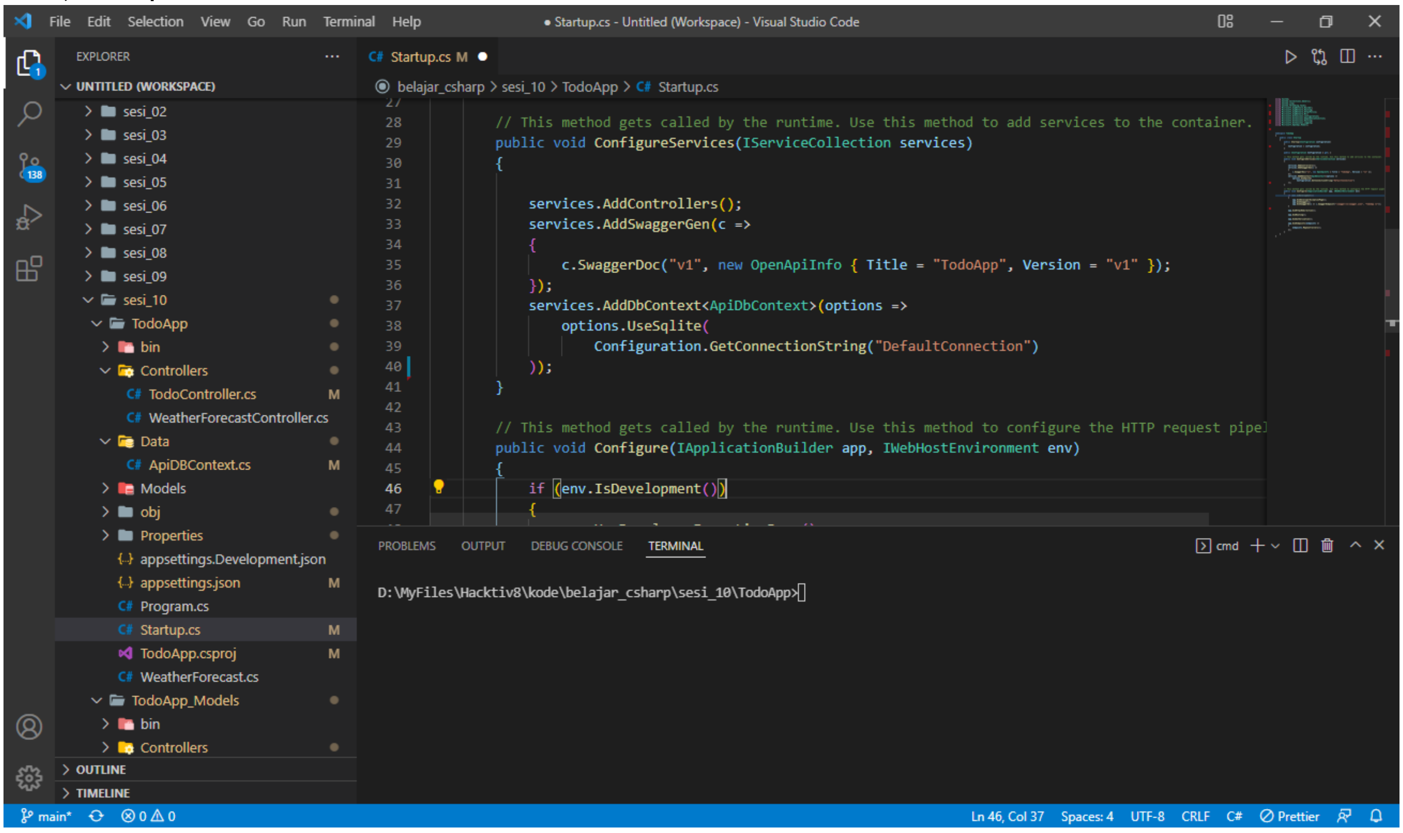
```
using Microsoft.EntityFrameworkCore;
```

untuk menggunakan `Microsoft.EntityFrameworkCore.Tools` dan `Microsoft.EntityFrameworkCore.Sqlite`

7. method 'ConfigurationServices' tambahkan

```
services.AddDbContext<ApiDbContext>(options =>
                options.UseSqlite(
                    Configuration.GetConnectionString("DefaultConnection")
            ));
```

Menjadi seperti ini:

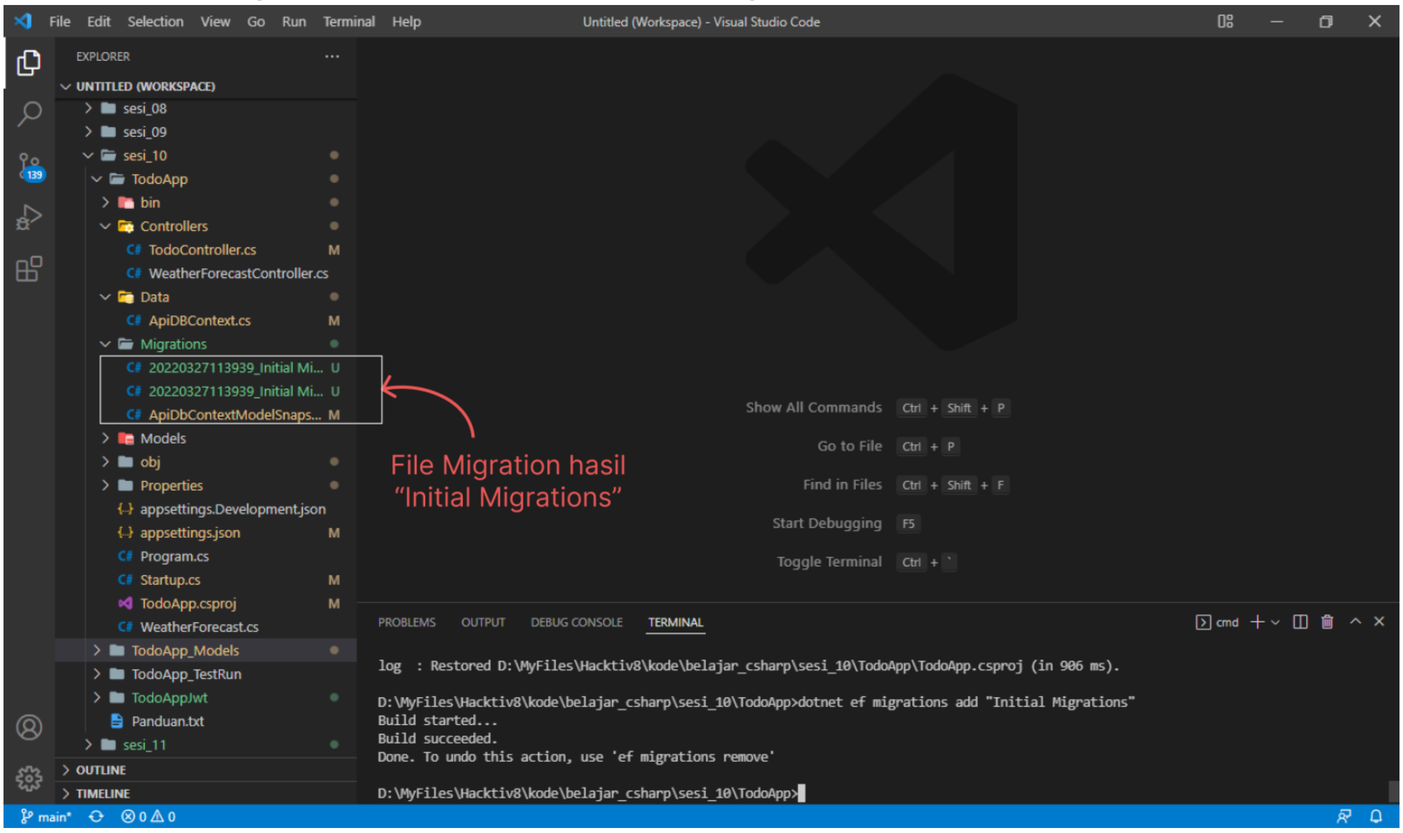


```
        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {

            services.AddControllers();
            services.AddSwaggerGen(c =>
            {
                c.SwaggerDoc("v1", new OpenApiInfo { Title = "TodoApp", Version = "v1" });
            });
            services.AddDbContext<ApiDbContext>(options =>
                options.UseSqlite(
                    Configuration.GetConnectionString("DefaultConnection")
                ));
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipel
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
```

```
D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>
```

8. Pada terminal jalankan untuk membuat migrations

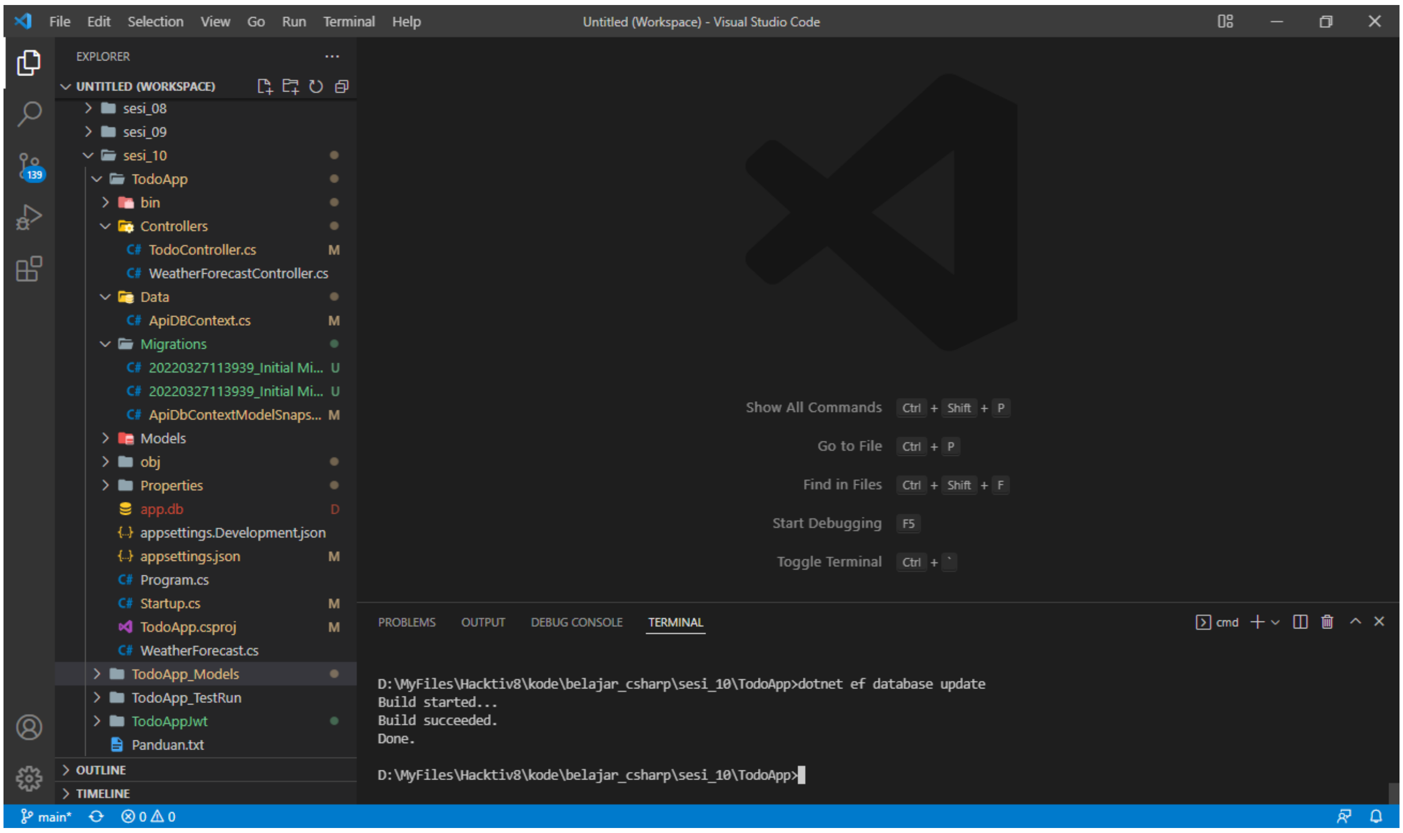
```
dotnet ef migrations add "Initial Migrations"
```


File Migration hasil
"Initial Migrations"
20220327113939_Initial Mi... U
20220327113939_Initial Mi... U
ApiDbContextModelSnaps... M

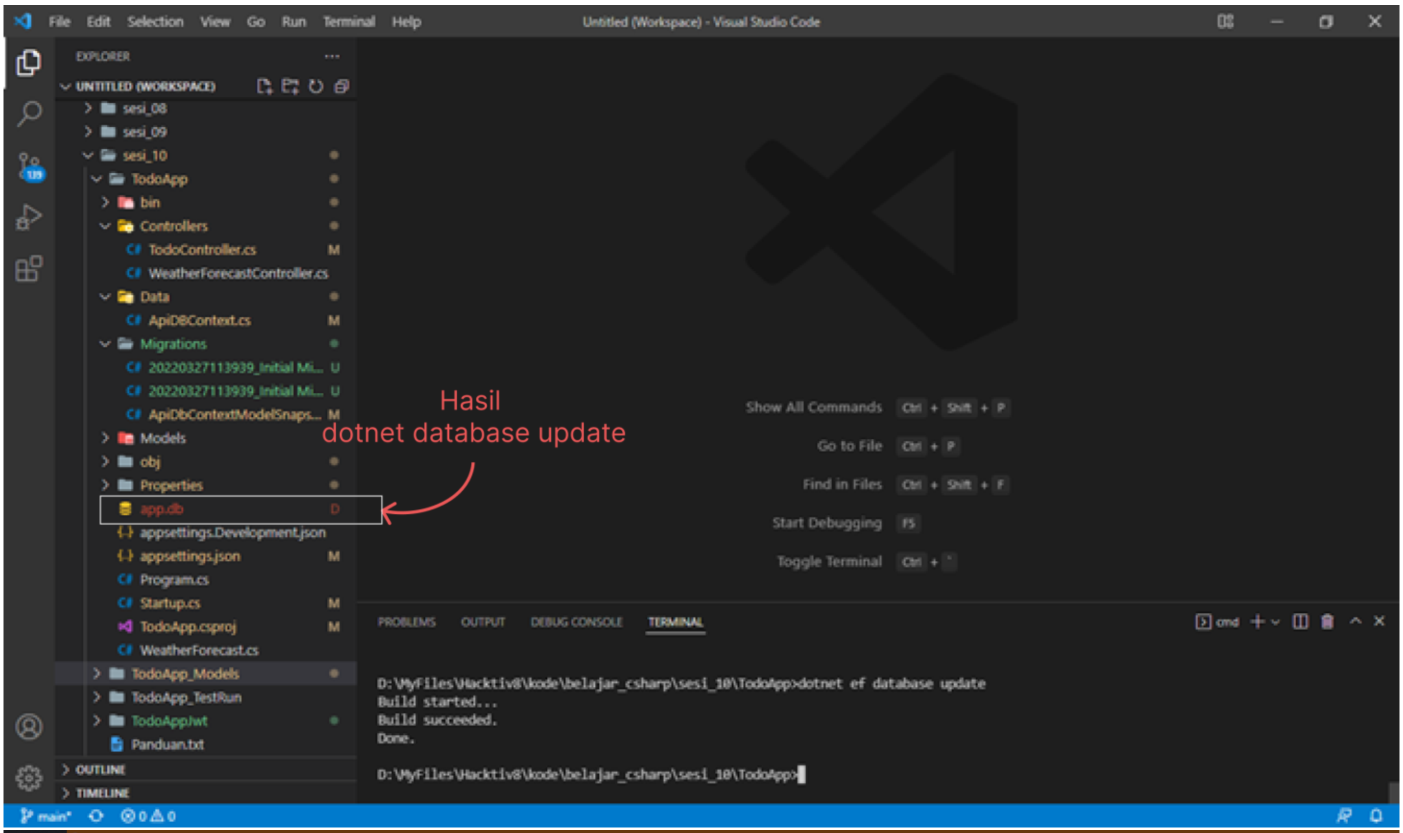
```
log  : Restored D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp\TodoApp.csproj (in 906 ms).

D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>dotnet ef migrations add "Initial Migrations"
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'

D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>
```

9. Lalu jalankan

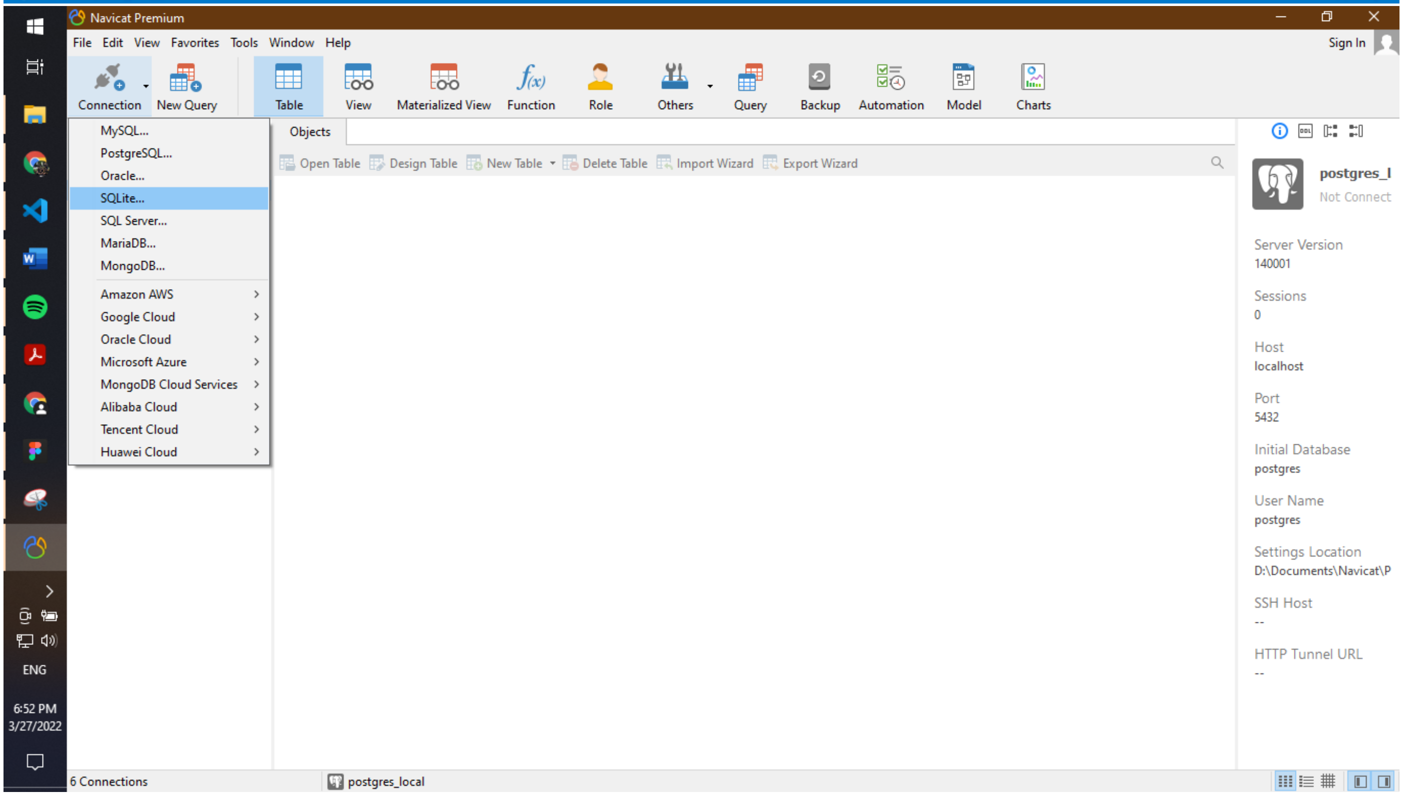
```
dotnet ef database update
```



```
D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>dotnet ef database update
Build started...
Build succeeded.
Done.

D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>
```

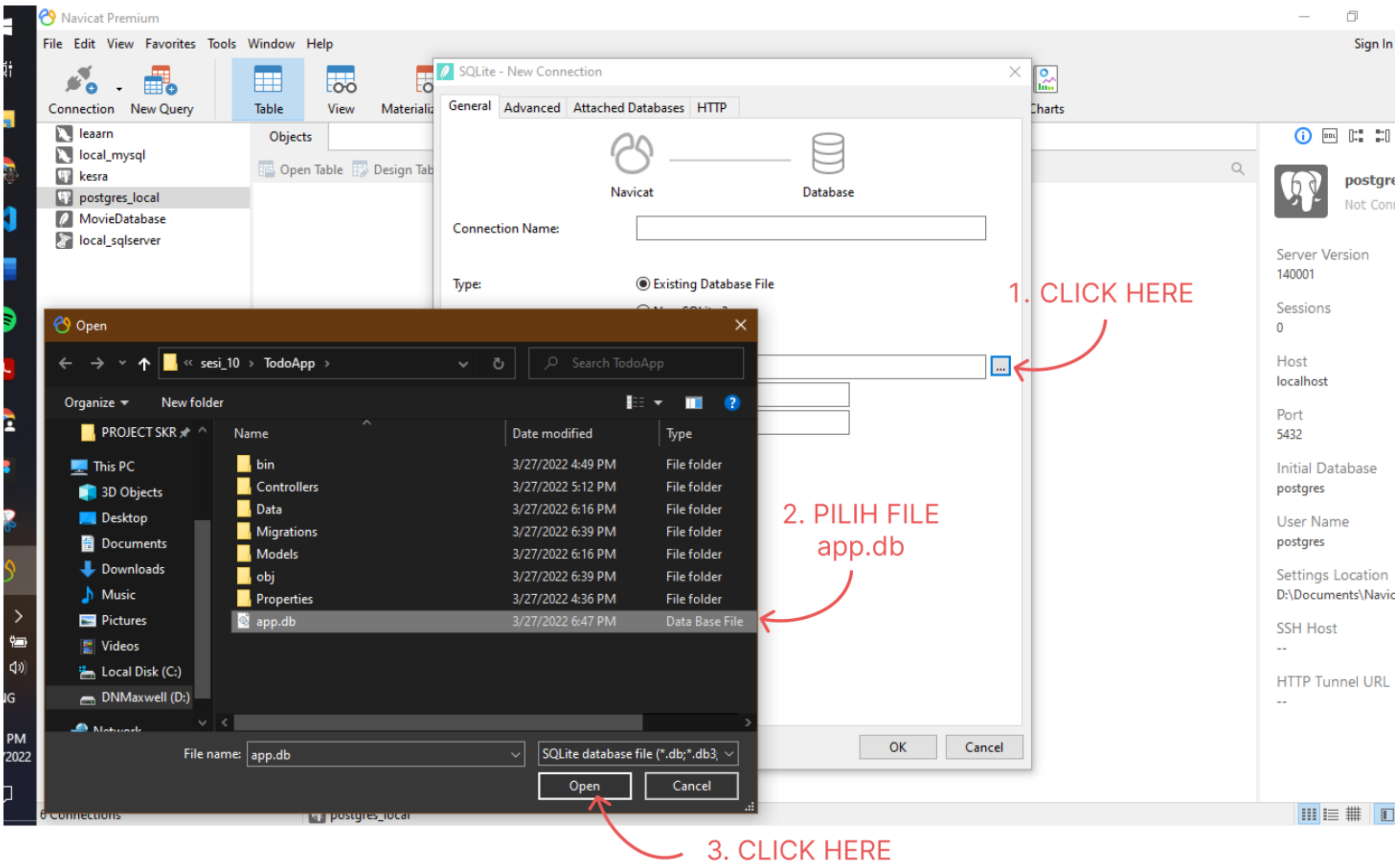


```
D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>dotnet ef database update
Build started...
Build succeeded.
Done.

D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>
```



Server Version
140001
Sessions
0
Host
localhost
Port
5432
Initial Database
postgres
User Name
postgres
Settings Location
D:\Documents\Navicat\P
SSH Host
--
HTTP Tunnel URL
--

6 Connections    postgres_local



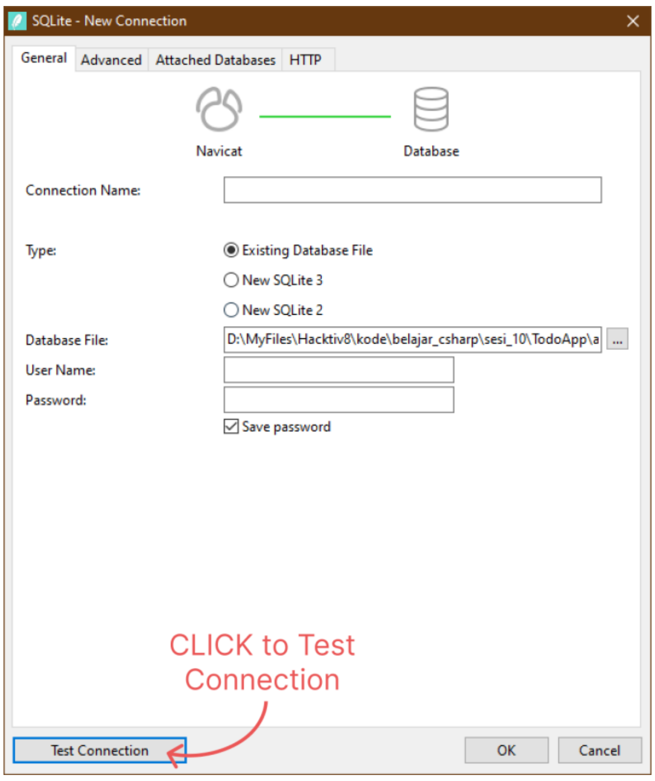
Connection Name:

Type: (●) Existing Database File

| Name | Date modified | Type |
|---|---|---|
| bin | 3/27/2022 4:49 PM | File folder |
| Controllers | 3/27/2022 5:12 PM | File folder |
| Data | 3/27/2022 6:16 PM | File folder |
| Migrations | 3/27/2022 6:39 PM | File folder |
| Models | 3/27/2022 6:16 PM | File folder |
| obj | 3/27/2022 6:39 PM | File folder |
| Properties | 3/27/2022 4:36 PM | File folder |
| app.db | 3/27/2022 6:47 PM | Data Base File |

File name: app.db

SQLite database file (*.db;*.db3;



Connection Name:

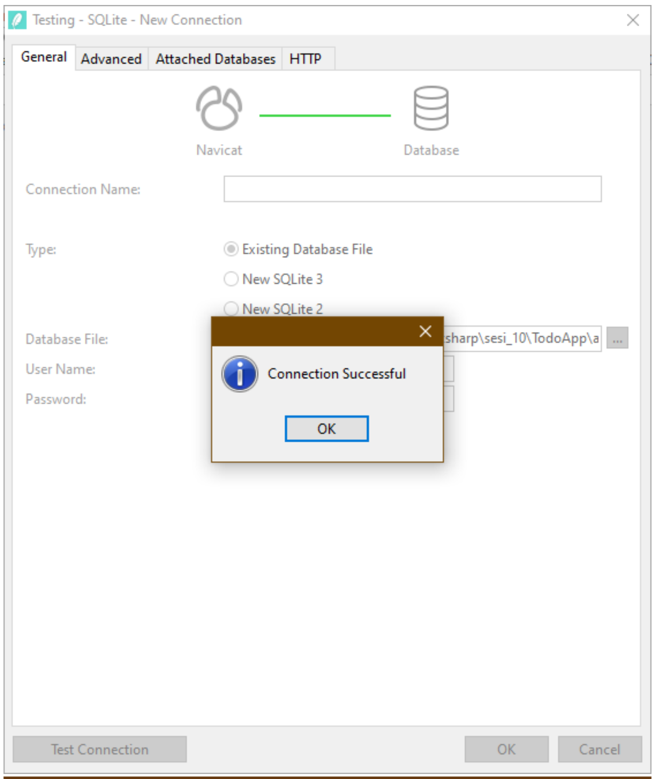
Type: [✓ Existing Database File] New SQLite 3 New SQLite 2

Database File: D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp\a

User Name:

Password:

[✓ Save password]


Testing - SQLite - New Connection

General | Advanced | Attached Databases | HTTP

Connection Name:

Type: (●) Existing Database File ( ) New SQLite 3 ( ) New SQLite 2

Database File: sharp\sesi_10\TodoApp\a

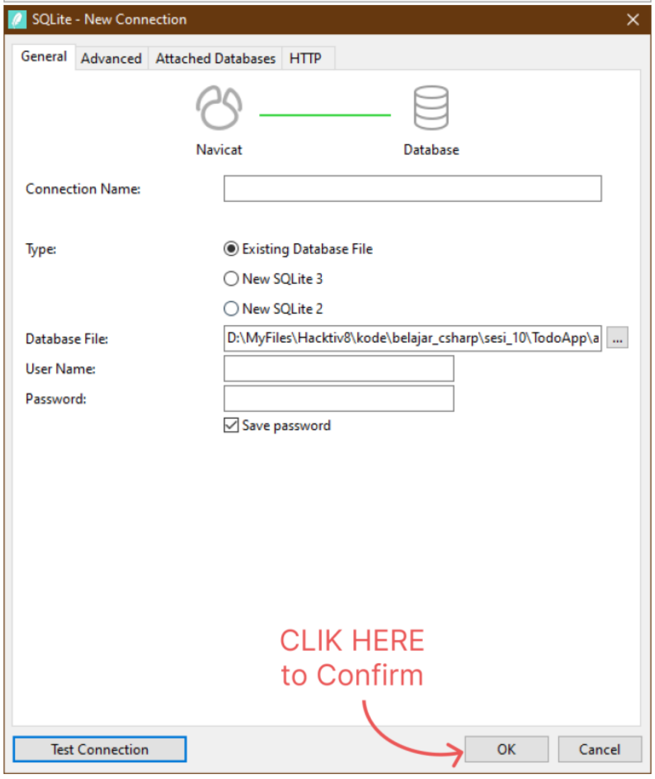
User Name:

Password:

Connection Successful

OK

Test Connection | OK | Cancel


SQLite - New Connection

General | Advanced | Attached Databases | HTTP

Connection Name:

Type: (●) Existing Database File ( ) New SQLite 3 ( ) New SQLite 2

Database File: D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp\a

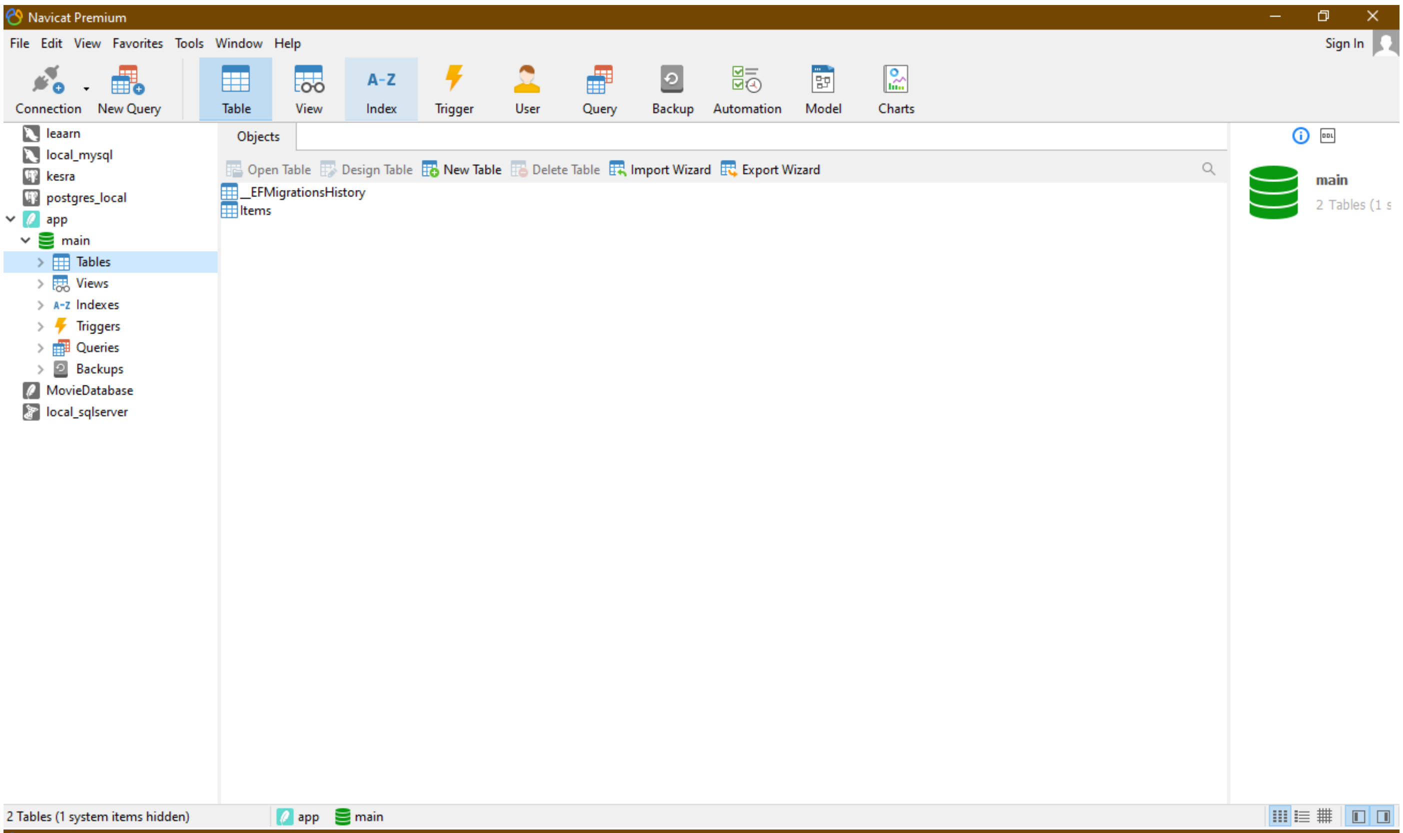
User Name:

Password:

☑ Save password

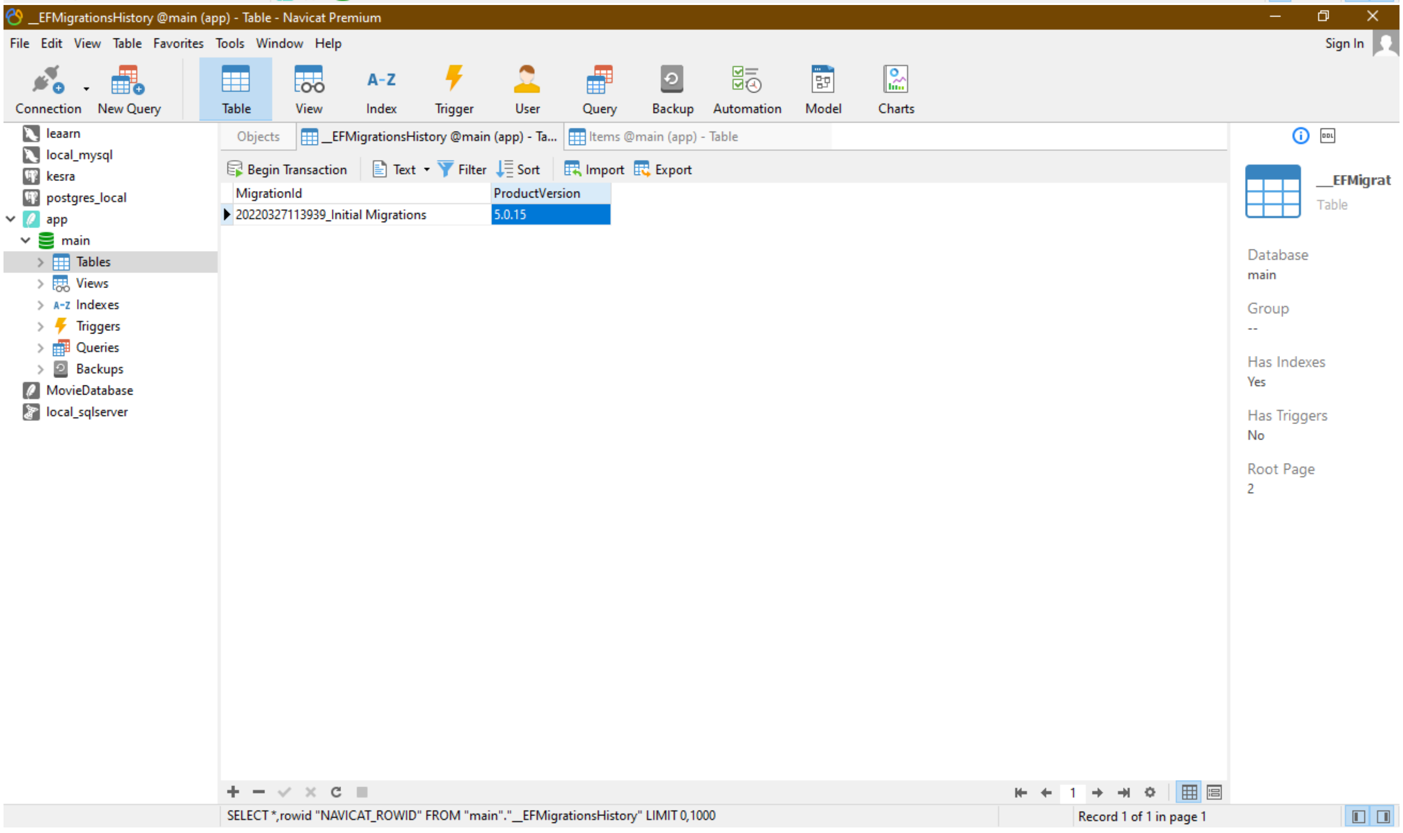
CLIK HERE
to Confirm

Test Connection | OK | Cancel



__EFMigrationsHistory
Items

2 Tables (1 system items hidden)



| MigrationId | ProductVersion |
|---|---|
| 20220327113939_Initial Migrations | 5.0.15 |

Database
main

Group
--

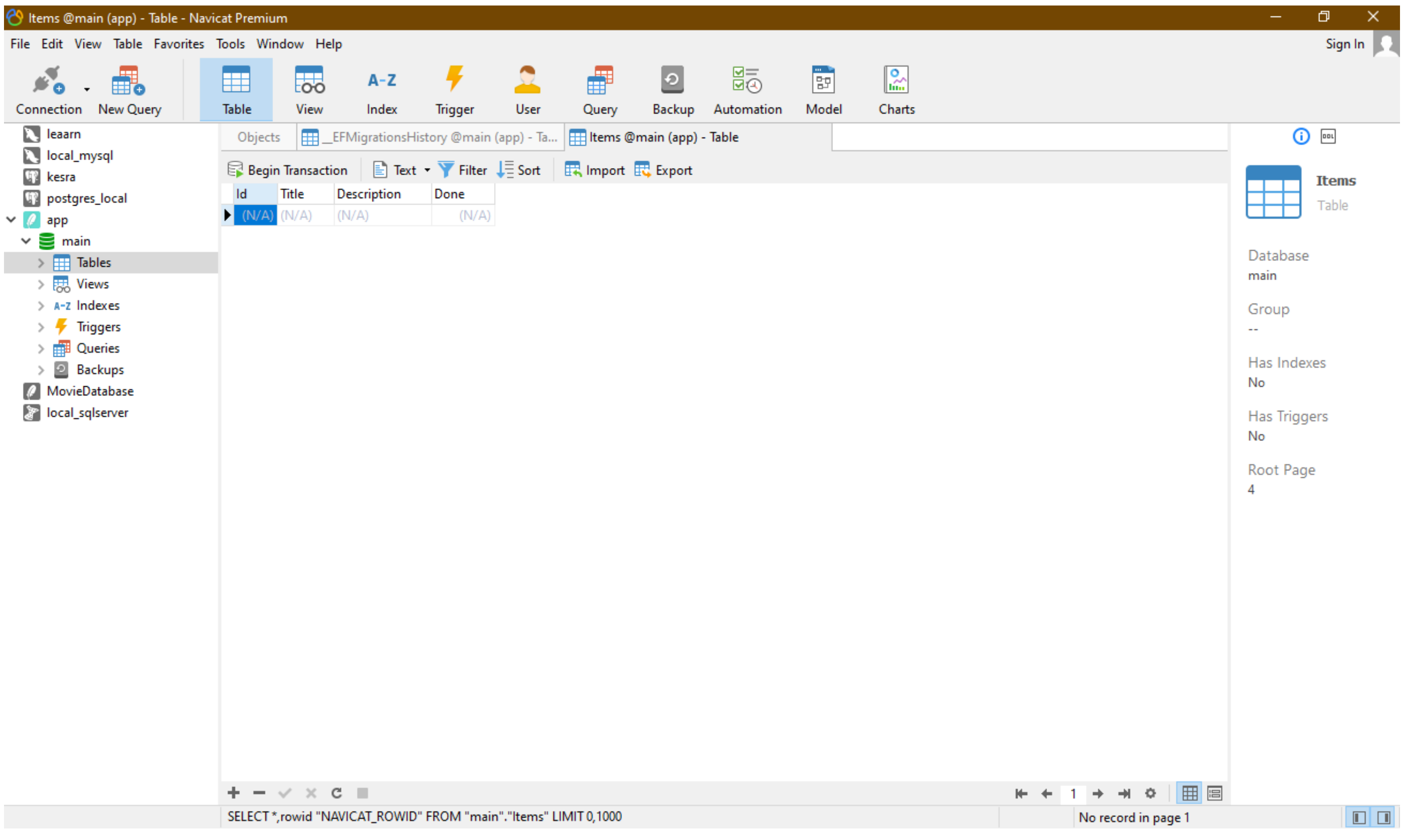
Has Indexes
Yes

Has Triggers
No

Root Page
2

```
SELECT *,rowid "NAVICAT_ROWID" FROM "main"."__EFMigrationsHistory" LIMIT 0,1000
```

Record 1 of 1 in page 1



| Id | Title | Description | Done |
|---|---|---|---|
| (N/A) | (N/A) | (N/A) | (N/A) |

```
SELECT *,rowid "NAVICAT_ROWID" FROM "main"."Items" LIMIT 0,1000
```

No record in page 1

## IV. Penambahan Model GET dan POST pada 'Web API TodoApp'

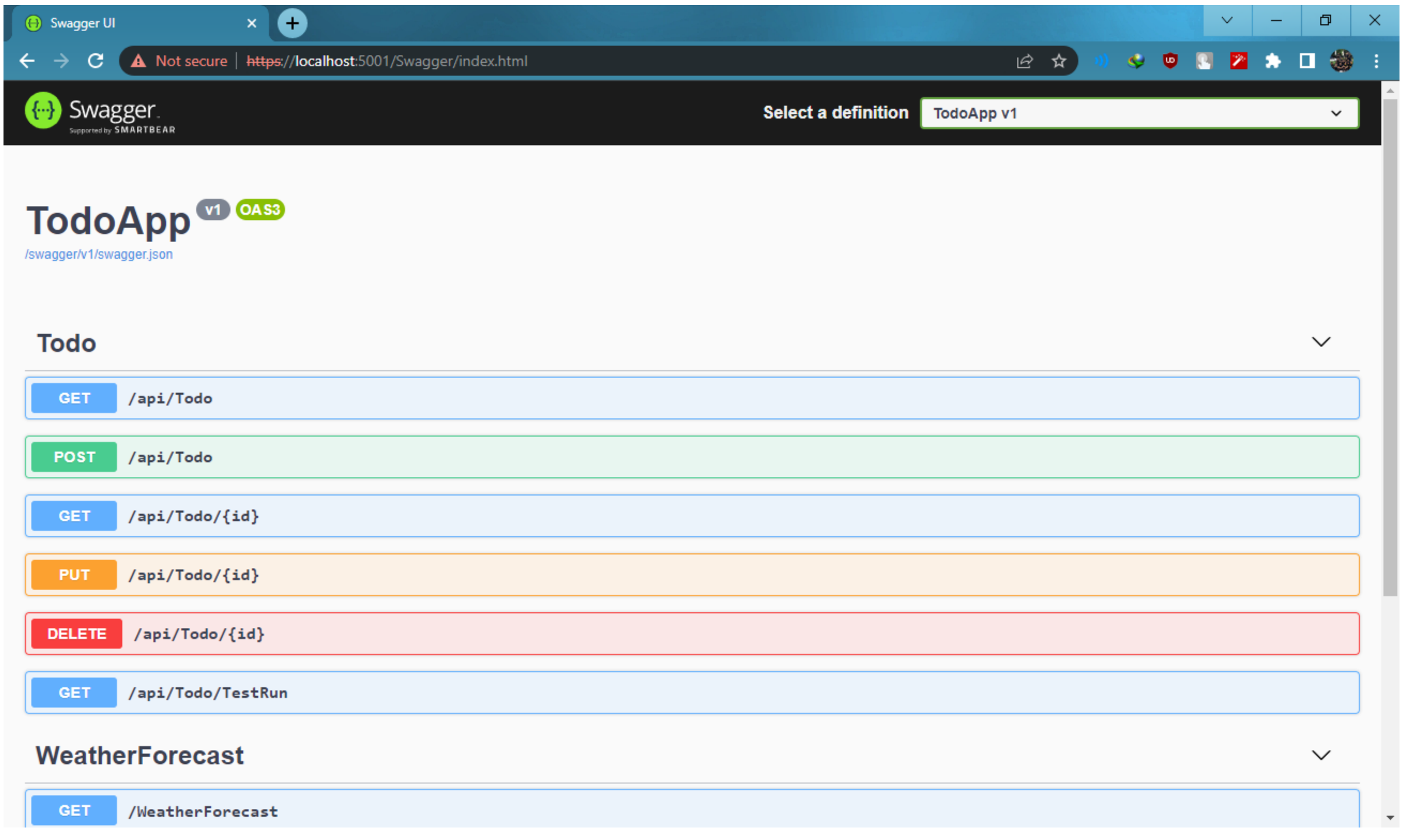
1. Pada **'Controller>TodoController'**, bagian atas (include), tambahkan

```
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using TodoApp.Data;
using TodoApp.Models;
```

`System.Threading.Tasks` untuk Task
`Microsoft.EntityFrameworkCore` untuk sqlite

2. Pada **'Controller>TodoController'**, bagian method TodoController: ControllerBase

```
private readonly ApiDbContext _context;
        public TodoController(ApiDbContext context){
            _context = context;
        }

        [HttpGet]
        public async Task<IActionResult> GetItems(){
            var items = await _context.Items.ToListAsync();
            return Ok(items);
        }

        [HttpPost]
        public async Task<IActionResult> CreateItem(ItemData data){
            if(ModelState.IsValid){
                await _context.Items.AddAsync(data);
                await _context.SaveChangesAsync();

                return CreatedAtAction(nameof(GetItems), new {id = data.Id}, data);
            }
                return new JsonResult("Something went wrong") {StatusCode = 500};
        }

        [HttpGet("{id}")]
        public async Task<IActionResult> GetItems(int id){
            var items = await _context.Items.FirstOrDefaultAsync(x=> x.Id == id);

            if(items== null)
                return NotFound();
            return Ok(items);
        }
        [HttpPut("{id}")]
        public async Task<IActionResult> UpdateItem(int id, ItemData item)
        {
            if(id!=item.Id){
                return BadRequest();
            }
            var existItem = await _context.Items.FirstOrDefaultAsync(x=> x.Id == id);

            if(existItem == null)
                return NotFound();

            existItem.Title = item.Title;
            existItem.Description = item.Description;
            existItem.Done = item.Done;


            await _context.SaveChangesAsync();
            return NoContent();


        }

        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteItem(int id){

            var existItem = await _context.Items.FirstOrDefaultAsync(x=> x.Id == id);

            if(existItem == null)
                return NotFound();

            _context.Items.Remove(existItem);
            return Ok(existItem);
        }
```


Swagger UI

localhost:5001/Swagger/index.html

Select a definition: TodoApp v1

TodoApp v1 OAS3

/swagger/v1/swagger.json

Todo

GET /api/Todo

POST /api/Todo

GET /api/Todo/{id}

PUT /api/Todo/{id}

DELETE /api/Todo/{id}

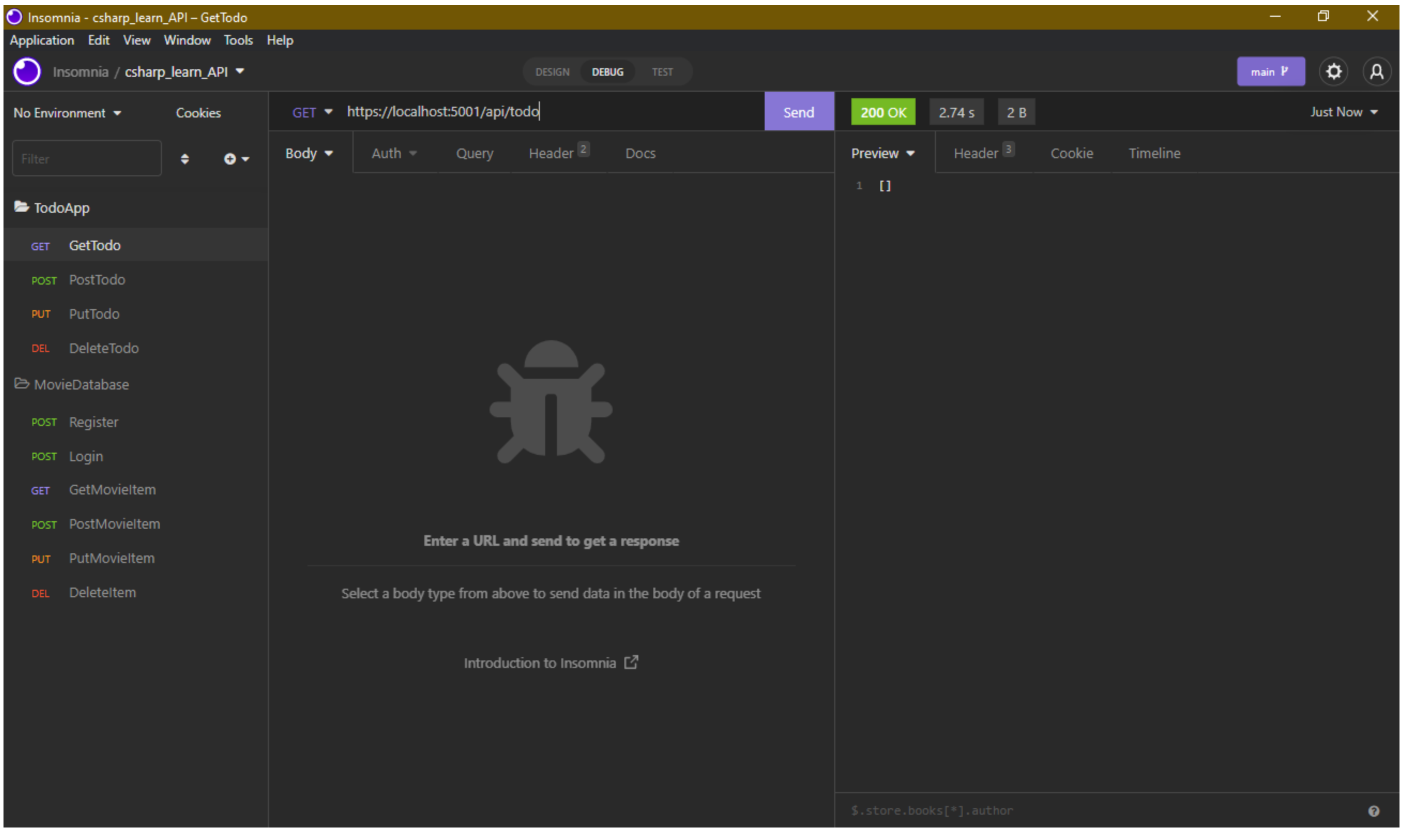
GET /api/Todo/TestRun

WeatherForecast

GET /WeatherForecast

## Request Menggunakan Insomnia

- GET


Insomnia - csharp_learn_API – GetTodo

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

200 OK 2.74 s 2 B Just Now

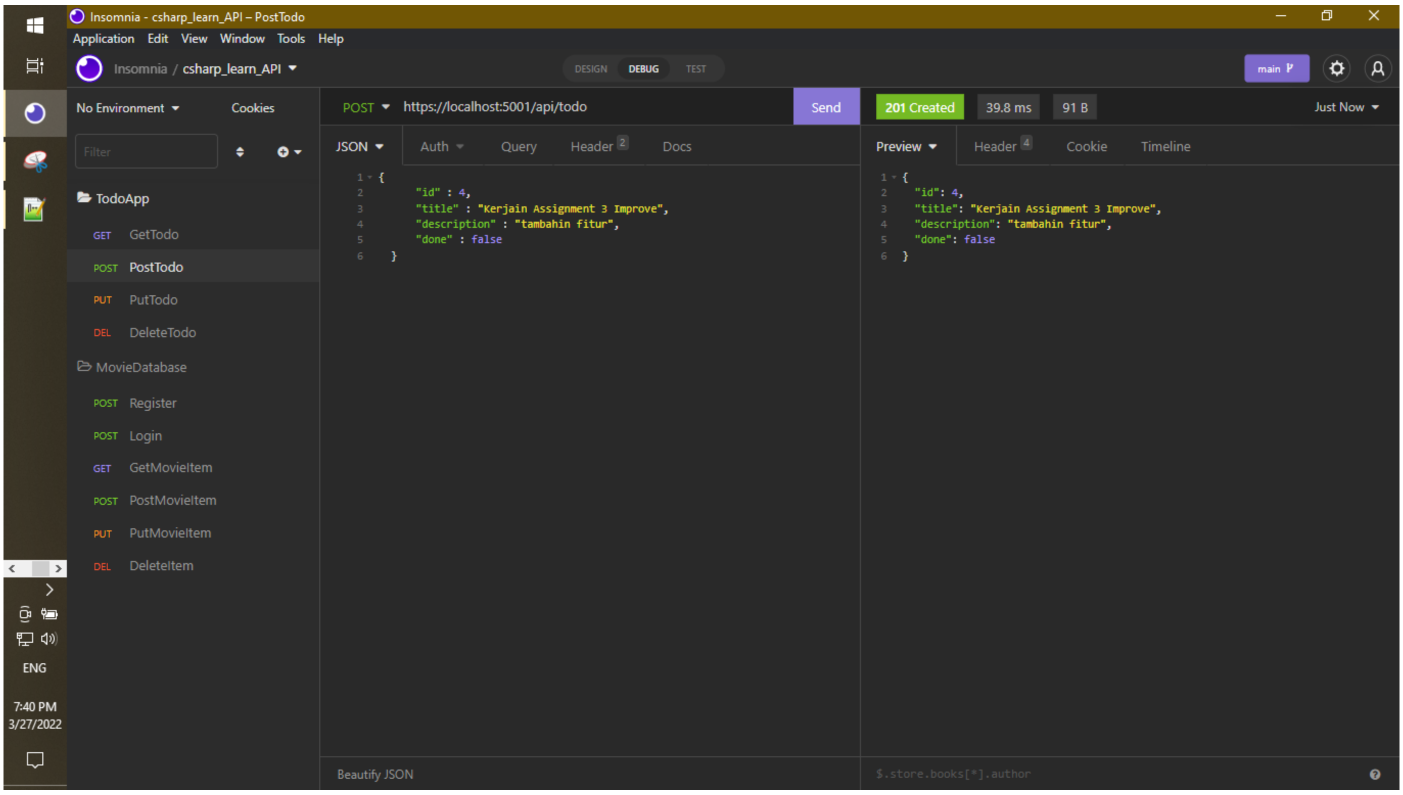
```
[]
```

TodoApp: GET GetTodo, POST PostTodo, PUT PutTodo, DEL DeleteTodo

MovieDatabase: POST Register, POST Login, GET GetMovieItem, POST PostMovieItem, PUT PutMovieItem, DEL DeleteItem

- POST

Body Input JSON:

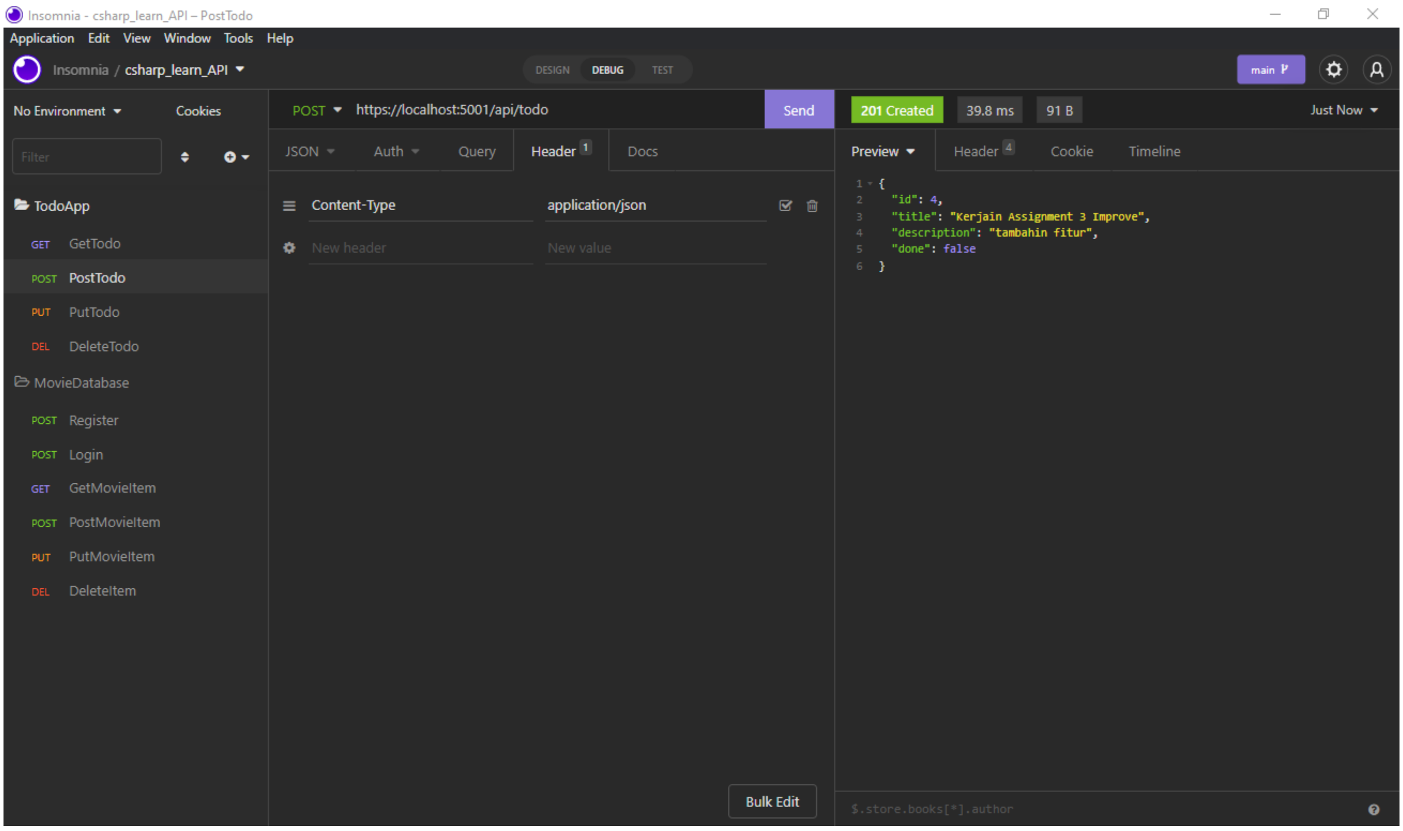

POST https://localhost:5001/api/todo

```
{
    "id" : 4,
    "title" : "Kerjain Assignment 3 Improve",
    "description" : "tambahin fitur",
    "done" : false
 }
```

201 Created 39.8 ms 91 B

```
{
  "id": 4,
  "title": "Kerjain Assignment 3 Improve",
  "description": "tambahin fitur",
  "done": false
}
```

## Header:


POST https://localhost:5001/api/todo

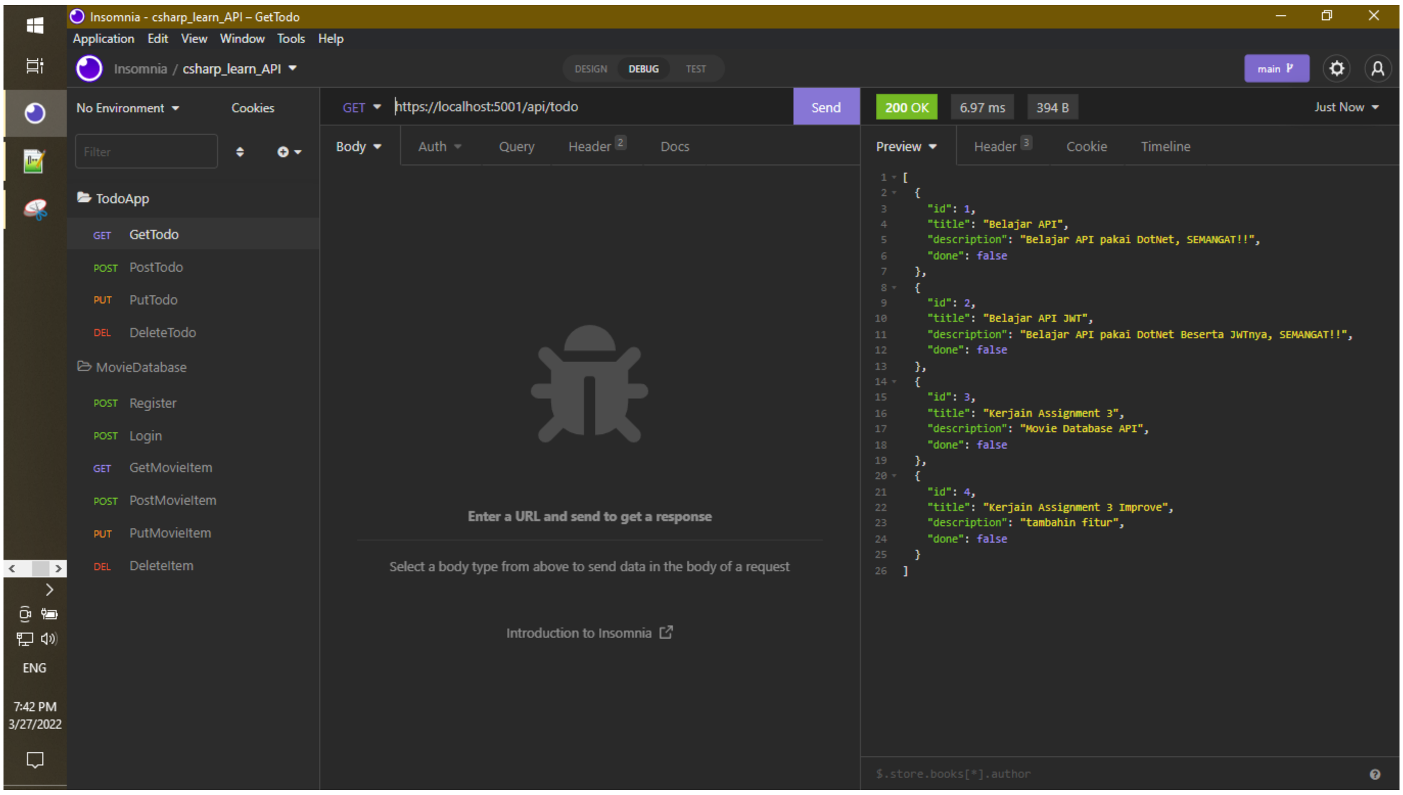
| Content-Type | application/json |
|---|---|

201 Created 39.8 ms 91 B

```
{
  "id": 4,
  "title": "Kerjain Assignment 3 Improve",
  "description": "tambahin fitur",
  "done": false
}
```

- PUT

## GET sebelum PUT



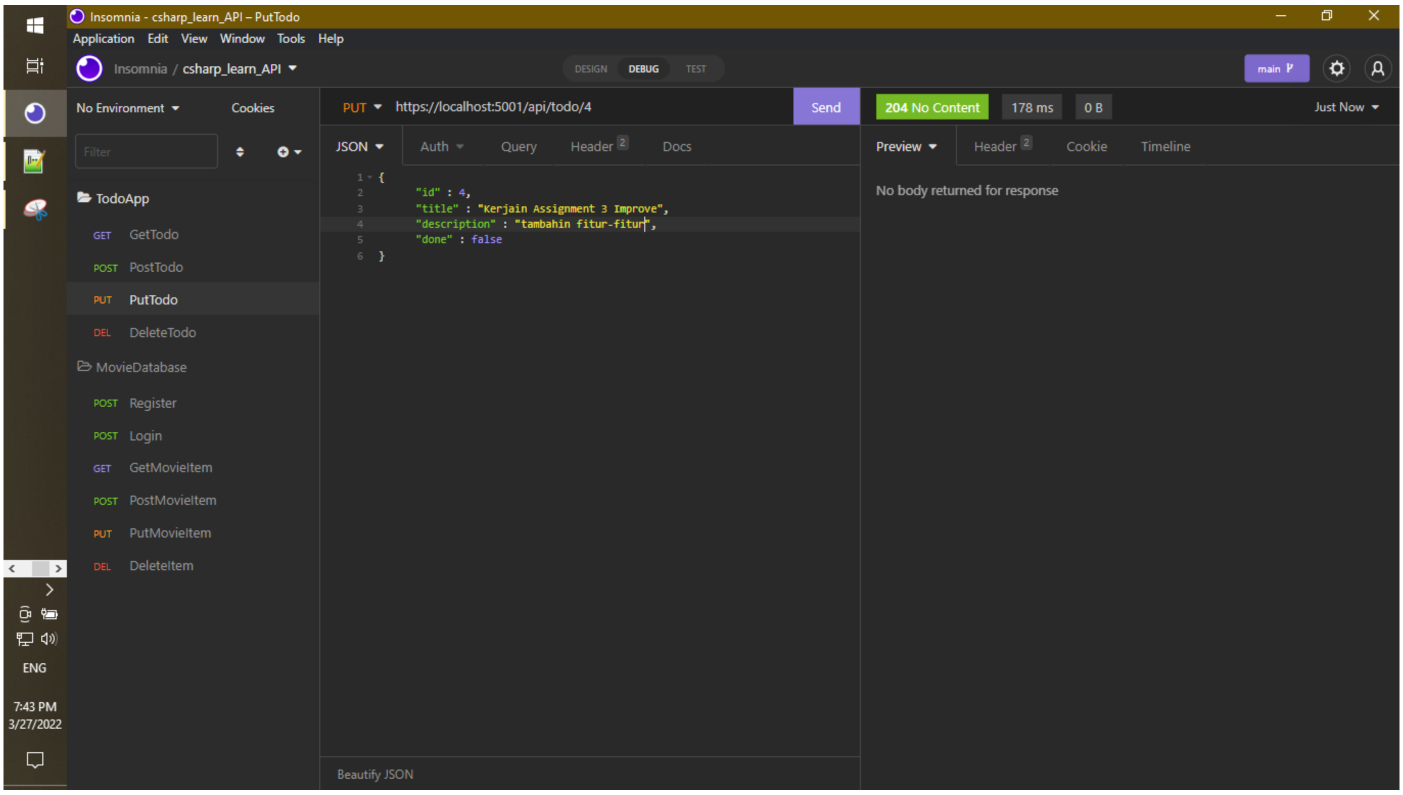
GET https://localhost:5001/api/todo

200 OK 6.97 ms 394 B

```
[
  {
    "id": 1,
    "title": "Belajar API",
    "description": "Belajar API pakai DotNet, SEMANGAT!!",
    "done": false
  },
  {
    "id": 2,
    "title": "Belajar API JWT",
    "description": "Belajar API pakai DotNet Beserta JWTnya, SEMANGAT!!",
    "done": false
  },
  {
    "id": 3,
    "title": "Kerjain Assignment 3",
    "description": "Movie Database API",
    "done": false
  },
  {
    "id": 4,
    "title": "Kerjain Assignment 3 Improve",
    "description": "tambahin fitur",
    "done": false
  }
]
```

$.store.books[*].author

## PUT



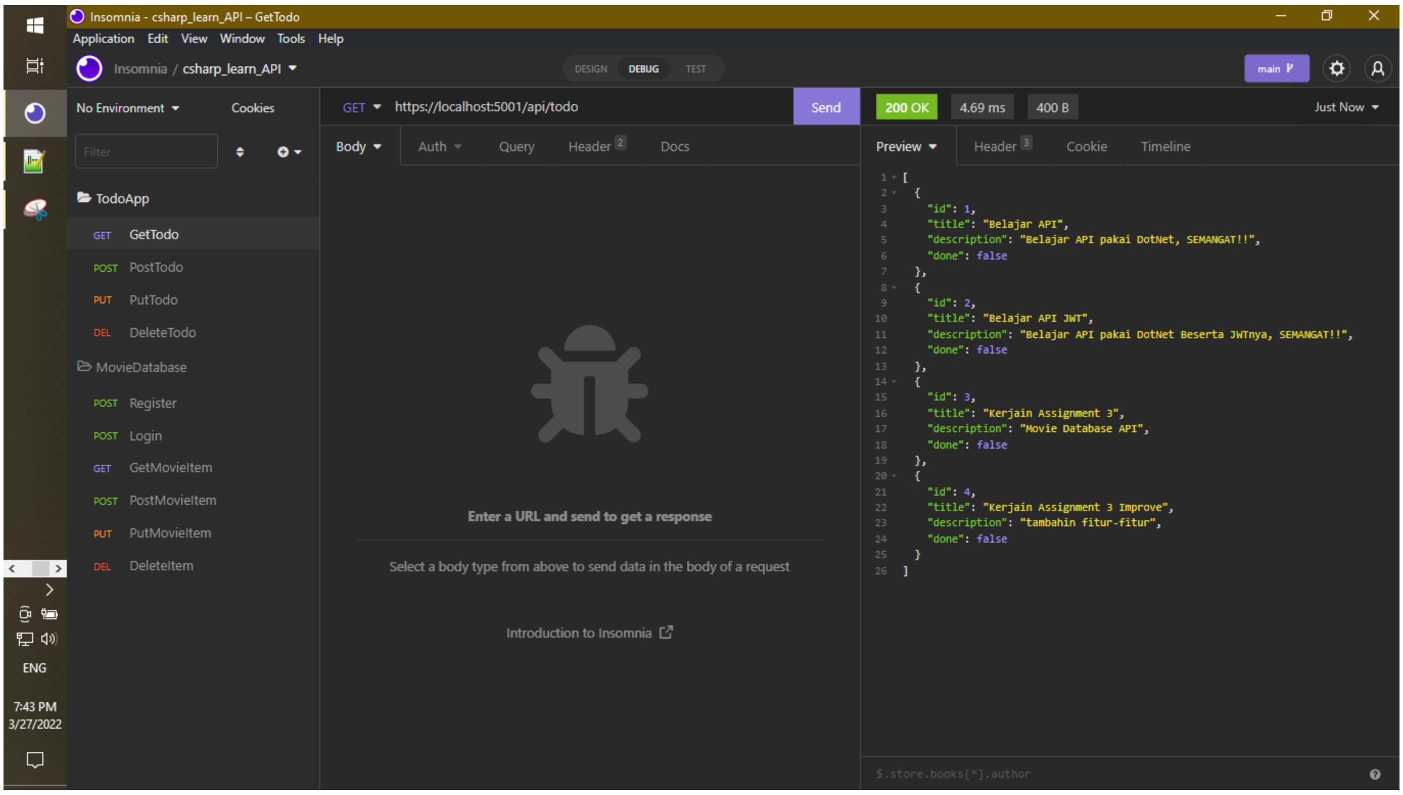
PUT https://localhost:5001/api/todo/4

```
{
    "id" : 4,
    "title" : "Kerjain Assignment 3 Improve",
    "description" : "tambahin fitur-fitur",
    "done" : false
}
```

204 No Content 178 ms 0 B

No body returned for response

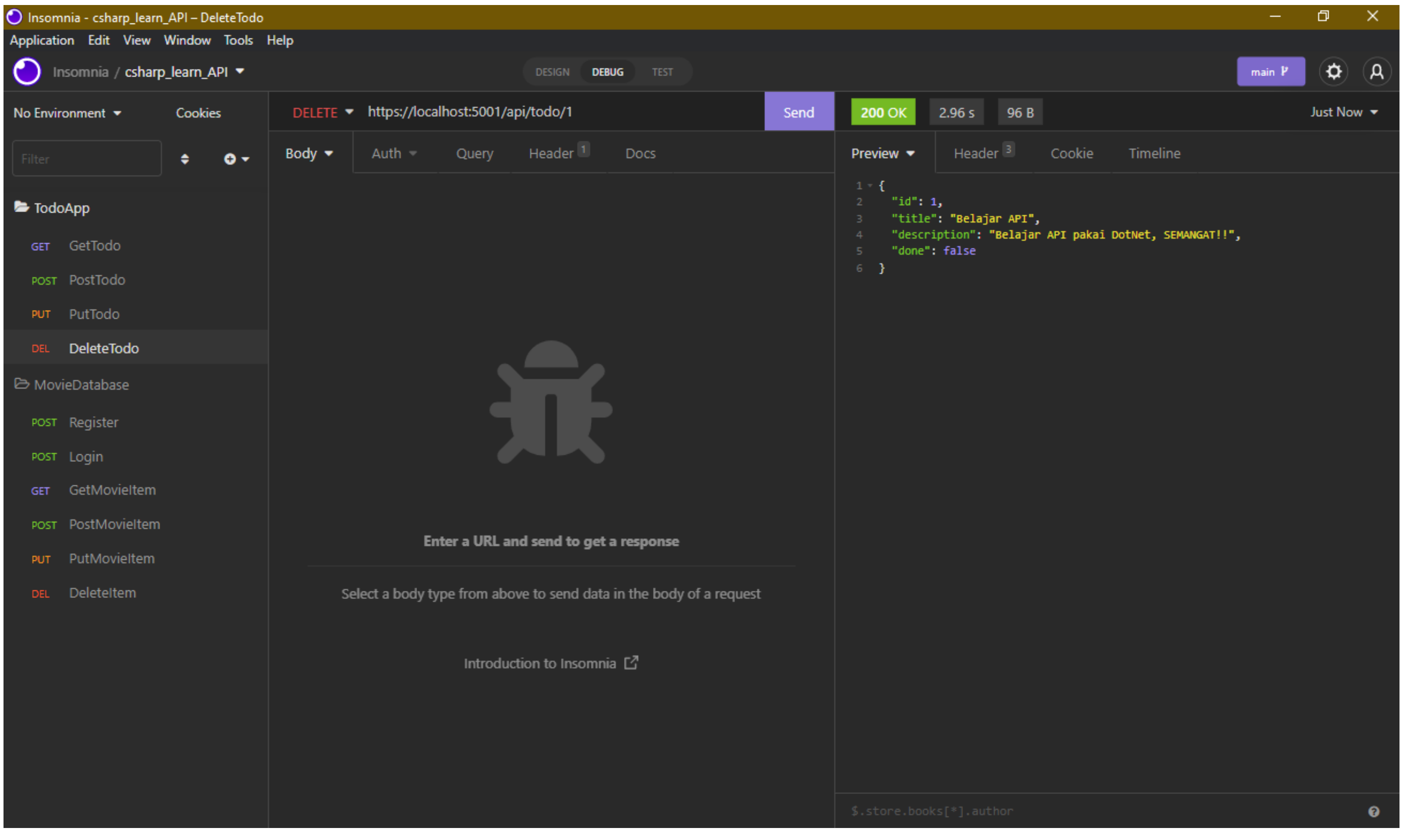
## GET setelah PUT



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

```
[
  {
    "id": 1,
    "title": "Belajar API",
    "description": "Belajar API pakai DotNet, SEMANGAT!!",
    "done": false
  },
  {
    "id": 2,
    "title": "Belajar API JWT",
    "description": "Belajar API pakai DotNet Beserta JWTnya, SEMANGAT!!",
    "done": false
  },
  {
    "id": 3,
    "title": "Kerjain Assignment 3",
    "description": "Movie Database API",
    "done": false
  },
  {
    "id": 4,
    "title": "Kerjain Assignment 3 Improve",
    "description": "tambahin fitur-fitur",
    "done": false
  }
]
```

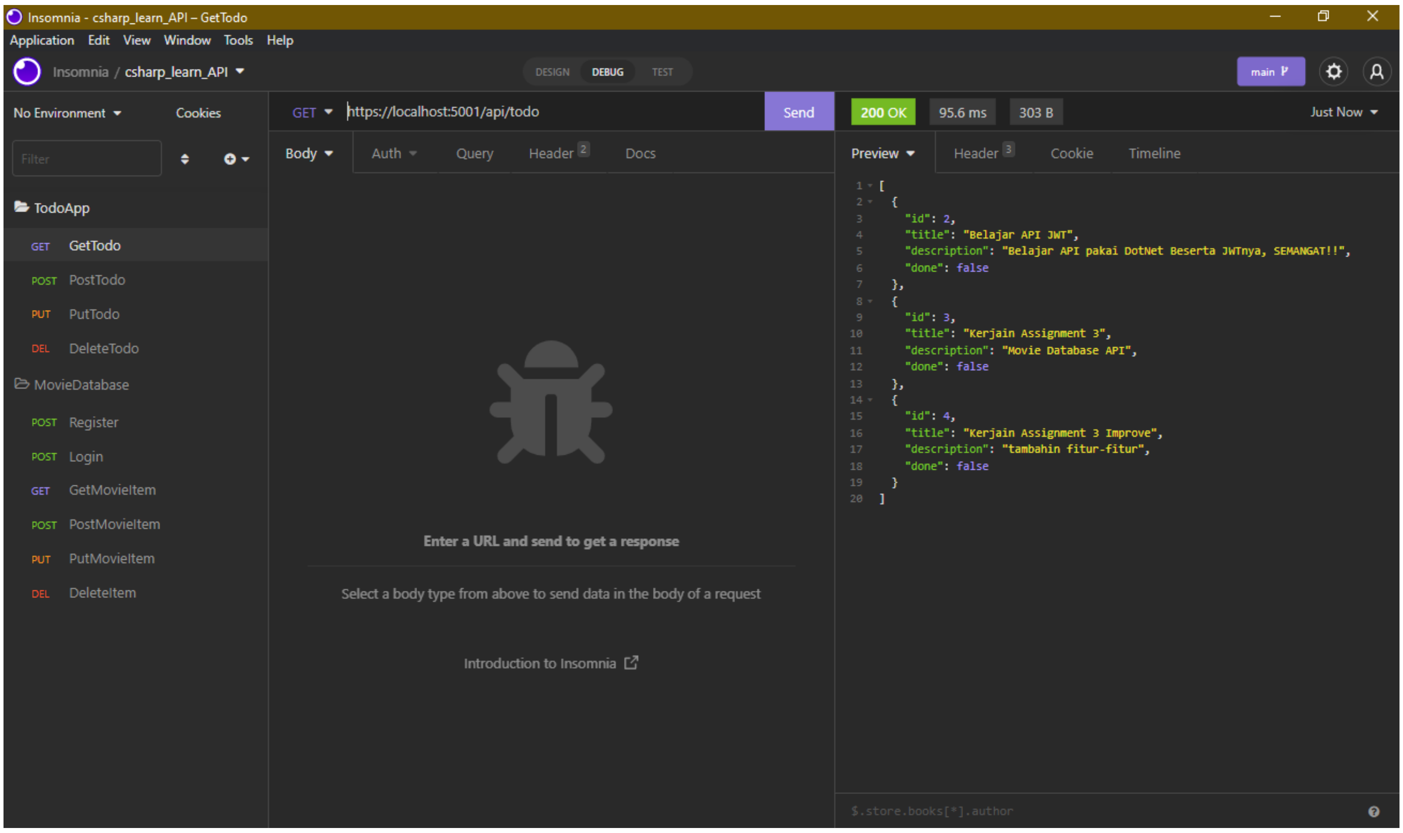
- **DELETE**



DELETE https://localhost:5001/api/todo/1 Send

```
{
  "id": 1,
  "title": "Belajar API",
  "description": "Belajar API pakai DotNet, SEMANGAT!!",
  "done": false
}
```

**GET setelah DELETE**


Insomnia - csharp_learn_API – GetTodo
Application Edit View Window Tools Help
Insomnia / csharp_learn_API
DESIGN DEBUG TEST
main
No Environment Cookies
Filter
TodoApp
GET GetTodo
POST PostTodo
PUT PutTodo
DEL DeleteTodo
MovieDatabase
POST Register
POST Login
GET GetMovieItem
POST PostMovieItem
PUT PutMovieItem
DEL DeleteItem
Body Auth Query Header 2 Docs
Enter a URL and send to get a response
Select a body type from above to send data in the body of a request
Introduction to Insomnia
Preview Header 3 Cookie Timeline
Just Now
$.store.books[*].author

```
GET https://localhost:5001/api/todo    Send
200 OK  95.6 ms  303 B
```

```
[
  {
    "id": 2,
    "title": "Belajar API JWT",
    "description": "Belajar API pakai DotNet Beserta JWTnya, SEMANGAT!!",
    "done": false
  },
  {
    "id": 3,
    "title": "Kerjain Assignment 3",
    "description": "Movie Database API",
    "done": false
  },
  {
    "id": 4,
    "title": "Kerjain Assignment 3 Improve",
    "description": "tambahin fitur-fitur",
    "done": false
  }
]
```

## V. Penambahan Authentication dengan JWT

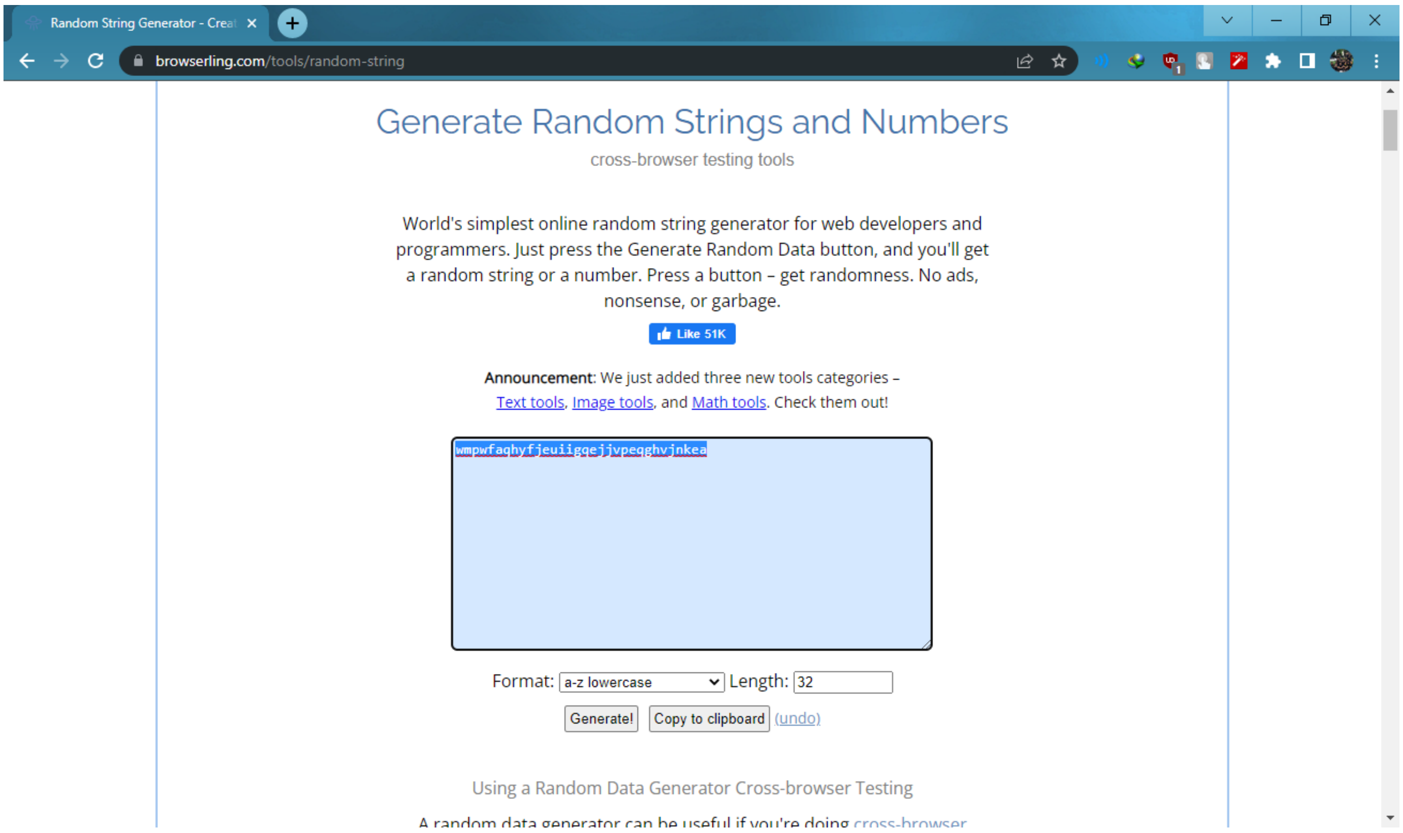
**Tambahkan package:**

- **dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer**
- **dotnet add package Microsoft.AspNetCore.Identity**
- **dotnet add package System.IdentityModel.Tokens.Jwt**
- **dotnet add package Microsoft.AspNetCore.Identity.UI**

```
services.AddDefaultIdentity<IdentityUser>(options =>
options.SignIn.RequireConfirmedAccount = true).AddEntityFrameworkStores<ApiDbContext>();
```

1. Buka https://www.browserling.com/tools/random-string, Setting Length = 32, lalu klik **‘generate’**

Hasil generate: wmpwfaqhyfjeuiigqejjvpeqghvjnkea


Generate Random Strings and Numbers

cross-browser testing tools

World's simplest online random string generator for web developers and programmers. Just press the Generate Random Data button, and you'll get a random string or a number. Press a button – get randomness. No ads, nonsense, or garbage.

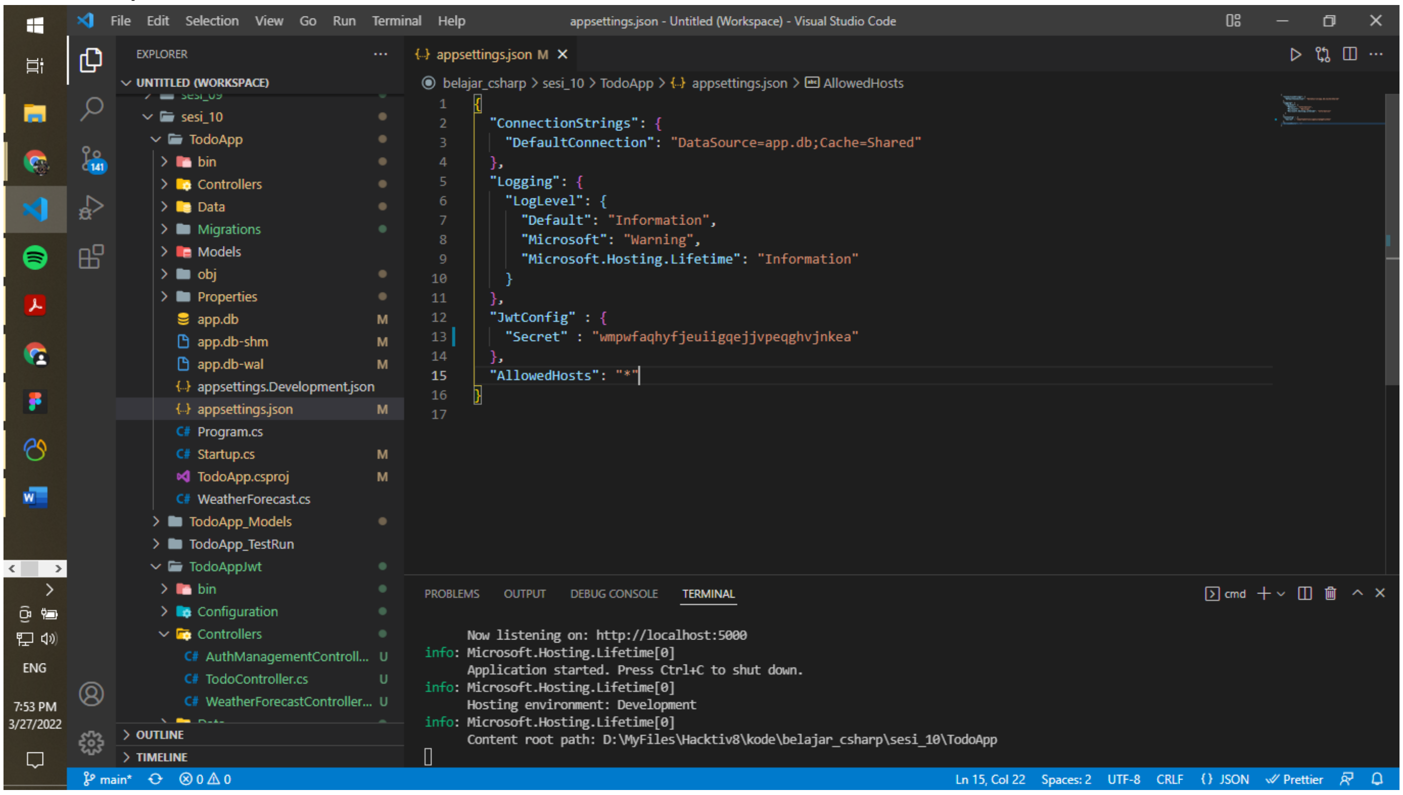
Announcement: We just added three new tools categories – Text tools, Image tools, and Math tools. Check them out!

```
wmpwfaqhyfjeuiigqejjvpeqghvjnkea
```

Format: a-z lowercase Length: 32

Using a Random Data Generator Cross-browser Testing

2. Lalu pada ‘appsetings.json’ tambahkan string hasil generate tersebut pada JwtConfig.Secret seperti dibawah ini


```
{
  "ConnectionStrings": {
    "DefaultConnection": "DataSource=app.db;Cache=Shared"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "JwtConfig" : {
    "Secret" : "wmpwfaqhyfjeuiigqejjvpeqghvjnkea"
  },
  "AllowedHosts": "*"
}
```

```
      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: D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp
```

3. Pada ‘TodoApp’ Buat folder ‘Configuration’

4. Pada ‘TodoApp>Configuration’ tambahkan file JwtConfig.cs dengan isi:

```
namespace TodoApp.Configuration
{
    public class JwtConfig
    {
        public string Secret {get; set;}
    }
}
```

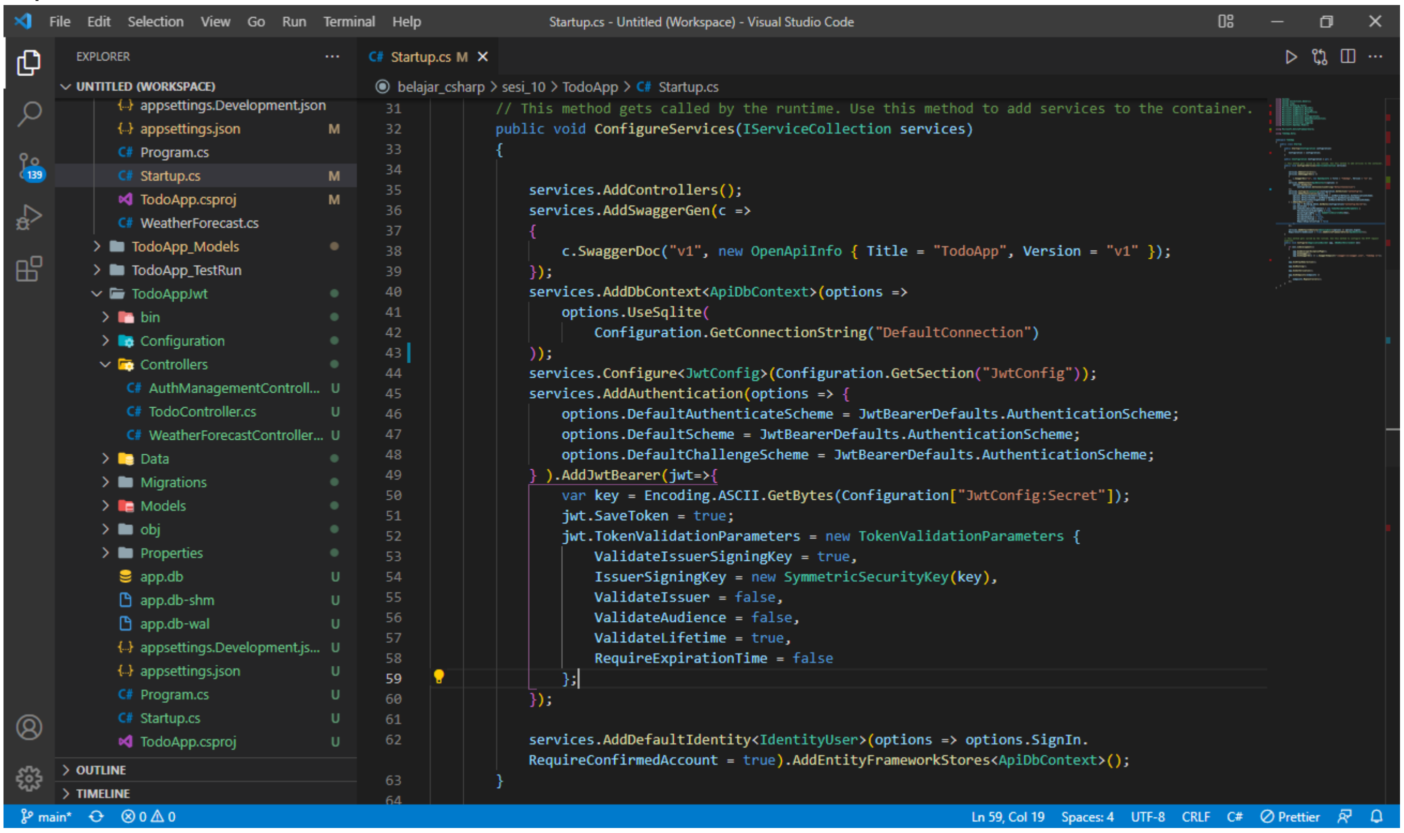
5. Lalu Pada ‘TodoApp>Startup.cs’ tambahkan di bagian atas file (include)

```
using System.Text;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using Microsoft.AspNetCore.Identity;
using TodoApp.Configuration;
```

6. Lalu Pada ‘TodoApp>Startup.cs’ tambahkan pada method ConfigurationServices:

```
services.Configure<JwtConfig>(Configuration.GetSection("JwtConfig"));
            services.AddAuthentication(options => {
                options.DefaultAuthenticateScheme = 
JwtBearerDefaults.AuthenticationScheme;
                options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
                options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
            } ).AddJwtBearer(jwt=>{
                var key = Encoding.ASCII.GetBytes(Configuration["JwtConfig:Secret"]);
                jwt.SaveToken = true;
                jwt.TokenValidationParameters = new TokenValidationParameters {
                    ValidateIssuerSigningKey = true,
                    IssuerSigningKey = new SymmetricSecurityKey(key),
                    ValidateIssuer = false,
                    ValidateAudience = false,
                    ValidateLifetime = true,
                    RequireExpirationTime = false
                };
            });

            services.AddDefaultIdentity<IdentityUser>(options => 
options.SignIn.RequireConfirmedAccount = true).AddEntityFrameworkStores<ApiDbContext>();
```

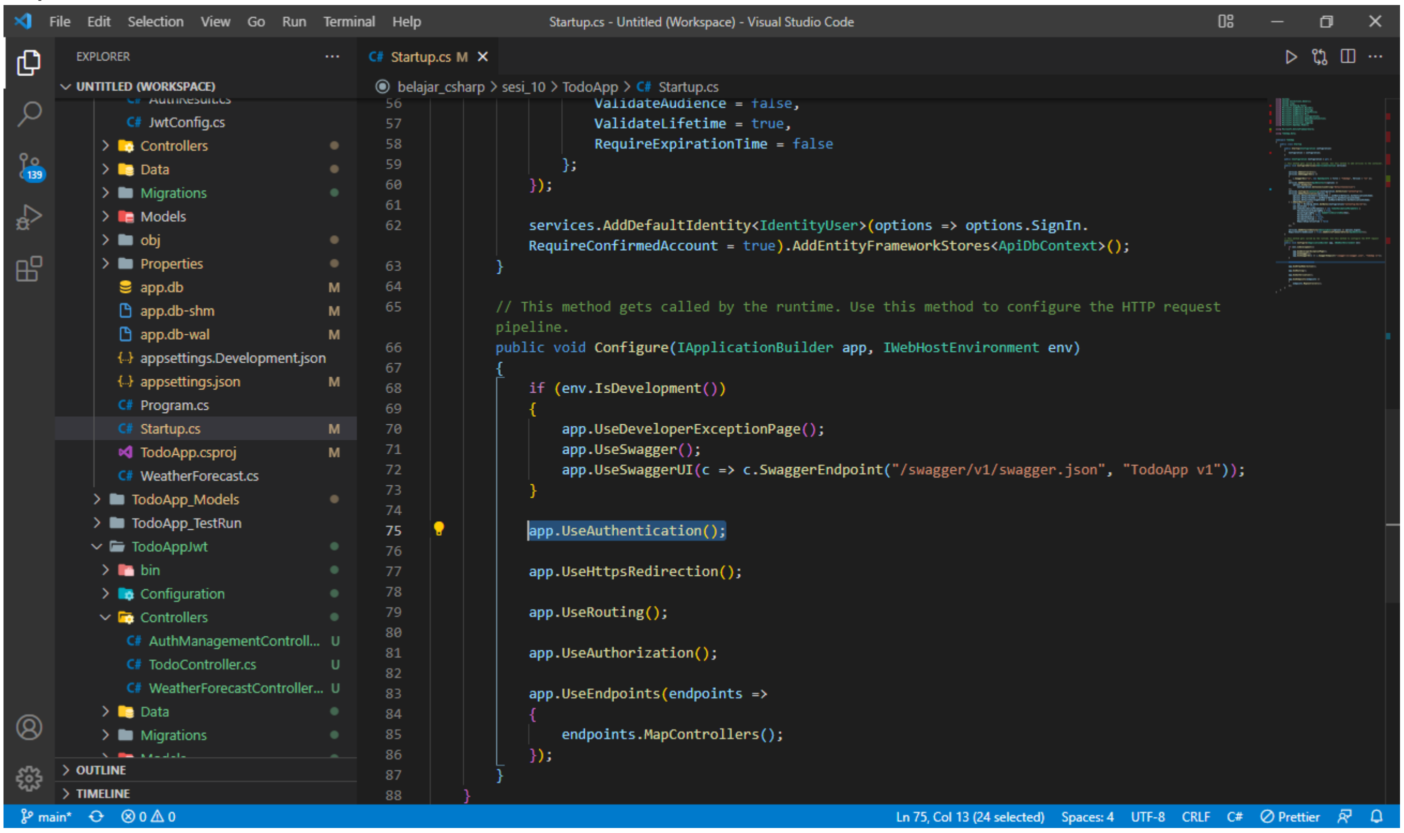
seperti ini:



```
// This method gets called by the runtime. Use this method to add services to the container.
public void ConfigureServices(IServiceCollection services)
{

    services.AddControllers();
    services.AddSwaggerGen(c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo { Title = "TodoApp", Version = "v1" });
    });
    services.AddDbContext<ApiDbContext>(options =>
        options.UseSqlite(
            Configuration.GetConnectionString("DefaultConnection")
    ));
    services.Configure<JwtConfig>(Configuration.GetSection("JwtConfig"));
    services.AddAuthentication(options => {
        options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
        options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
        options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    } ).AddJwtBearer(jwt=>{
        var key = Encoding.ASCII.GetBytes(Configuration["JwtConfig:Secret"]);
        jwt.SaveToken = true;
        jwt.TokenValidationParameters = new TokenValidationParameters {
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new SymmetricSecurityKey(key),
            ValidateIssuer = false,
            ValidateAudience = false,
            ValidateLifetime = true,
            RequireExpirationTime = false
        };
    });

    services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.
    RequireConfirmedAccount = true).AddEntityFrameworkStores<ApiDbContext>();
}
```

7. Lalu Pada ‘TodoApp>Startup.cs’ tambahkan pada method Configure:

```
app.UseAuthentication();
```
Seperti ini:

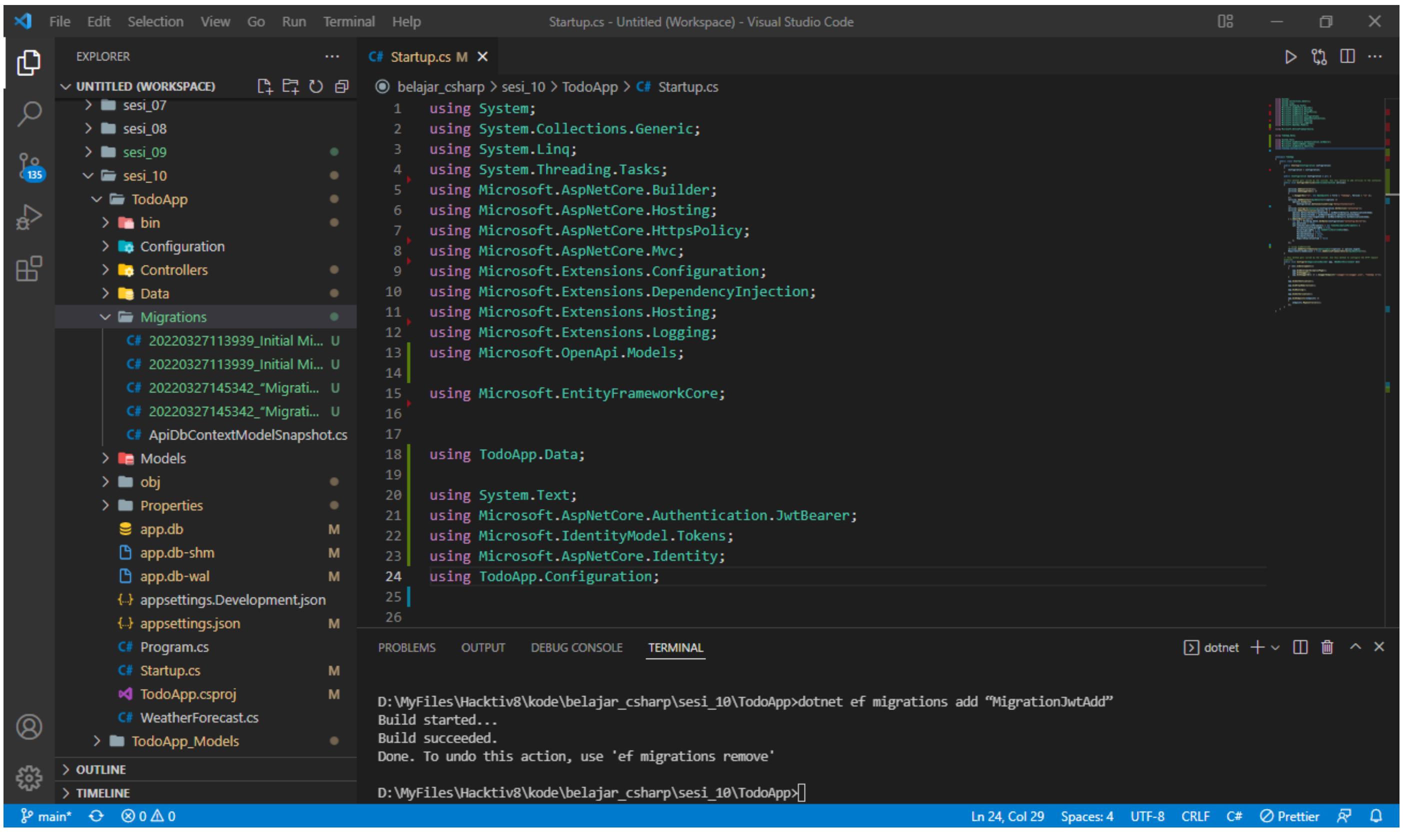


```
                ValidateAudience = false,
                ValidateLifetime = true,
                RequireExpirationTime = false
            };
        });

        services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.
        RequireConfirmedAccount = true).AddEntityFrameworkStores<ApiDbContext>();
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request
    pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
            app.UseSwagger();
            app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "TodoApp v1"));
        }

        app.UseAuthentication();

        app.UseHttpsRedirection();

        app.UseRouting();

        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}
```

8. Pada data ‘TodoApp>Data>ApiDbContext’ rubah

```
public class ApiDbContext : DbContext
```
menjadi
```
public class ApiDbContext : IdentityDbContext
```
berikut jadinya
```
using Microsoft.EntityFrameworkCore;
using TodoApp.Models;

namespace TodoApp.Data
{
    public class ApiDbContext : IdentityDbContext
    {
        public virtual DbSet<ItemData> Items {get;set;}
        public ApiDbContext(DbContextOptions<ApiDbContext> options):base(options)
        {

        }
    }
}
```
9. Menambahkan Migrasi

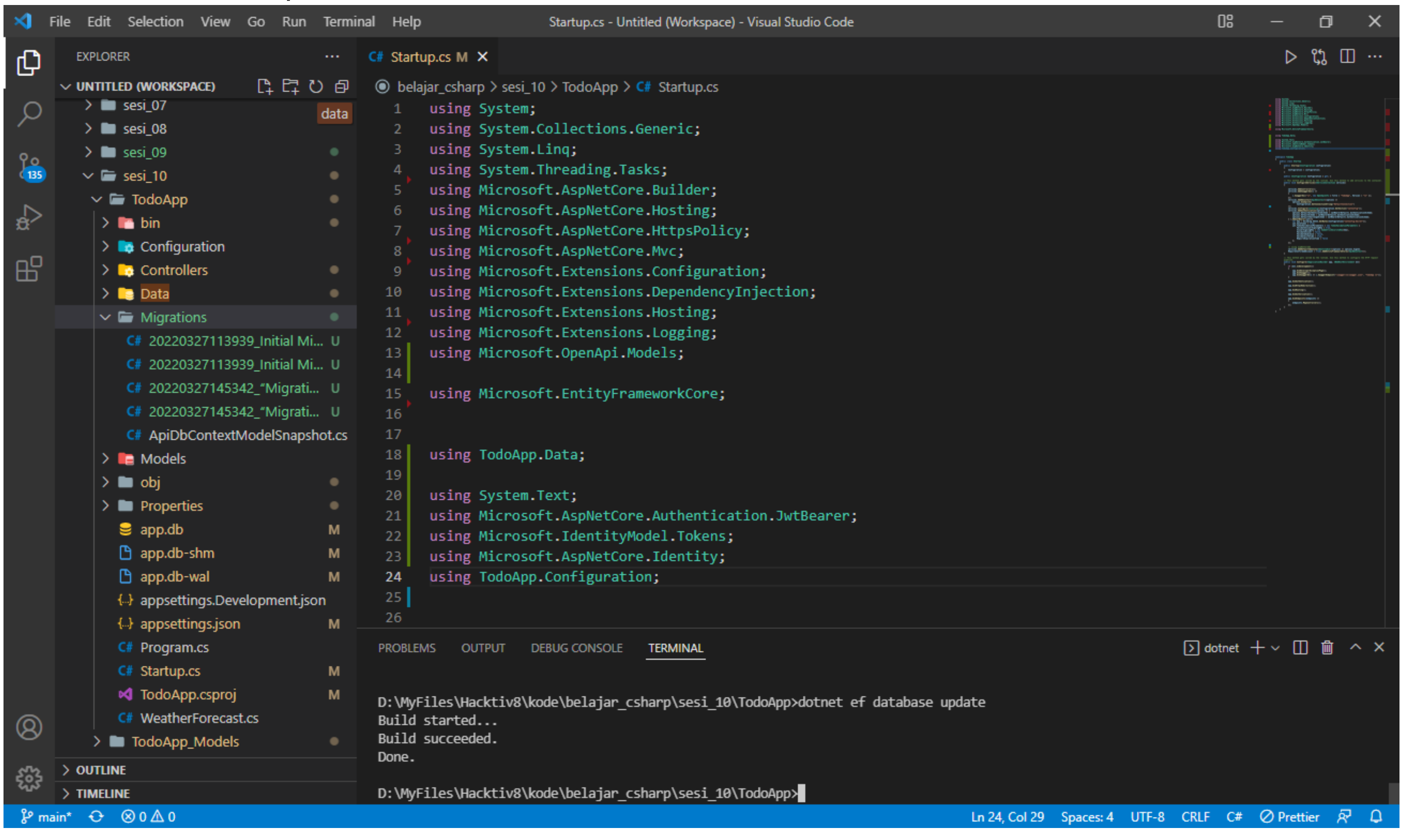
dotnet ef migrations add “MigrationJwtAdd”


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.OpenApi.Models;

using Microsoft.EntityFrameworkCore;


using TodoApp.Data;

using System.Text;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using Microsoft.AspNetCore.Identity;
using TodoApp.Configuration;
```

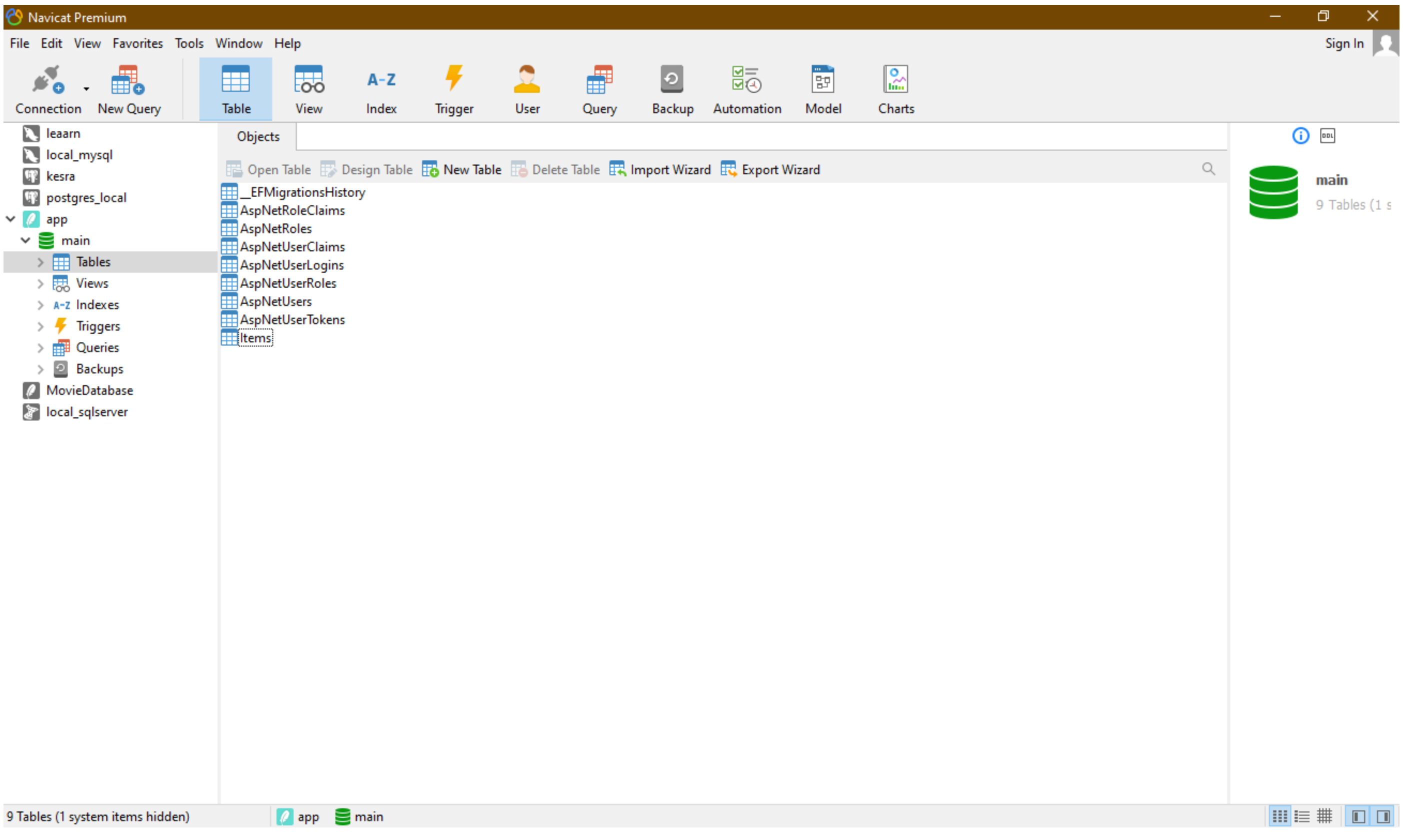
```
D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>dotnet ef migrations add "MigrationJwtAdd"
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'

D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>
```

10. Update database

dotnet ef database update


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.OpenApi.Models;

using Microsoft.EntityFrameworkCore;


using TodoApp.Data;

using System.Text;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using Microsoft.AspNetCore.Identity;
using TodoApp.Configuration;
```

```
D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>dotnet ef database update
Build started...
Build succeeded.
Done.

D:\MyFiles\Hacktiv8\kode\belajar_csharp\sesi_10\TodoApp>
```



learn
local_mysql
kesra
postgres_local
app
main
Tables
Views
Indexes
Triggers
Queries
Backups
MovieDatabase
local_sqlserver

__EFMigrationsHistory
AspNetRoleClaims
AspNetRoles
AspNetUserClaims
AspNetUserLogins
AspNetUserRoles
AspNetUsers
AspNetUserTokens
Items

9 Tables (1 system items hidden) app main

## VI. Penambahan Authentication dengan JWT II : Register

1. Pada 'TodoApp>Configuration' tambahkan file AuthResult.cs dengan isi:

```
using System.Collections.Generic;


namespace TodoApp.Configuration
{
    public class AuthResult
    {
        public string Token {get; set;}
        public bool Success {get; set;}
        public List<string> Errors {get; set;}
    }
}
```

2. Pada 'TodoApp>Models' Tambahkan Folder DTOs
3. Pada 'TodoApp>Models>DTO' Tambahkan Folder 'Requests' dan 'Responses'
4. Pada 'TodoApp>Models>DTO>Requests' Tambahkan File 'UserRegistrationDto.cs', dengan isi file:

UserRegistrationDto.cs:

```
using System.ComponentModel.DataAnnotations;

namespace TodoApp.Models.DTOs.Requests
{
    public class UserRegistationDto
    {
        [Required]
        public string Username {get; set;}
        [Required]
        [EmailAddress]
        public string Email {get;set;}
        [Required]
        public string Password {get;set;}
    }
}
```

5. Pada ‘TodoApp>Models>DTO>Responses’ tambahkan file ‘RegistrationResponse.cs’ dengan isi:

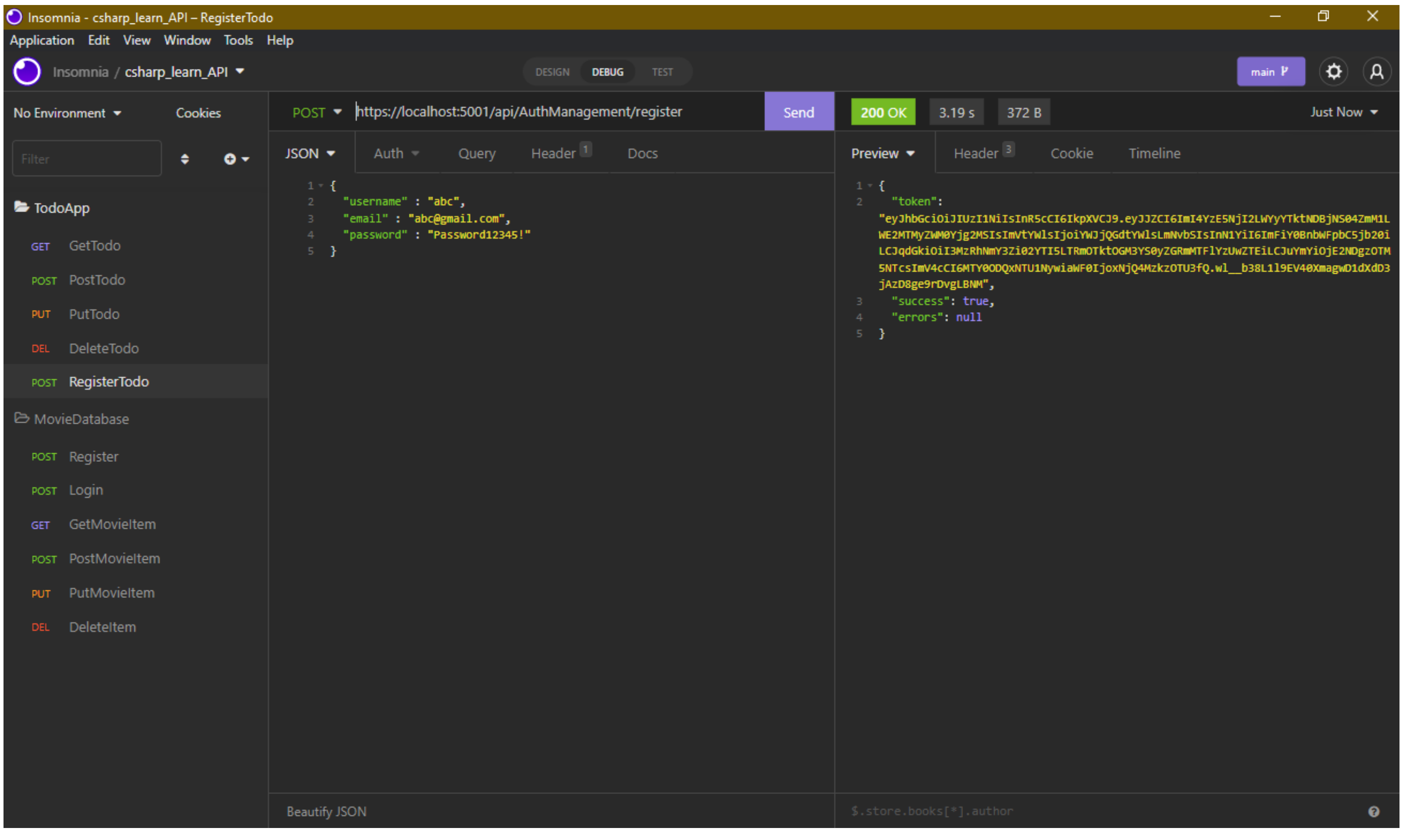
```
using TodoApp.Configuration;

namespace TodoApp.Models.DTOs.Responses
{
    public class RegistrationResponse: AuthResult
    {

    }
}
```

6. Pada ‘TodoApp>Controllers’ tambahkan file ‘AuthManagementController.cs’

```
using System;
using System.Collections.Generic;
using System.IdentityModel.Tokens.Jwt;
using System.Text;
using System.Linq;
using System.Threading.Tasks;
using System.Security.Claims;
using Microsoft.IdentityModel.Tokens;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;
using TodoApp.Models.DTOs.Requests;
using TodoApp.Models.DTOs.Responses;
using TodoApp.Configuration;


namespace TodoApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthManagementController : ControllerBase
    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly JwtConfig _jwtConfig;

        public class AuthManagementController : ControllerBase

    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly JwtConfig _jwtConfig;

        public AuthManagementController(UserManager<IdentityUser> userManager,
IOptionsMonitor<JwtConfig> optionMonitor)
```

```
        {
            _userManager = userManager;
            _jwtConfig = optionMonitor.CurrentValue;
        }

        [HttpPost]
        [Route("Register")]
        public async Task<IActionResult> Register([FromBody] UserRegistationDto user)
        {
            if(ModelState.IsValid)
            {
                var existingUser = await _userManager.FindByEmailAsync(user.Email);

                if(existingUser != null)
                {
                    return BadRequest(new RegistrationResponse(){
                        Errors = new List<string>(){
                            "Email already in use"
                        },
                        Success = false
                    });
                }

                var newUser = new IdentityUser() { Email = user.Email, UserName =
user.Username};
                var isCreated = await _userManager.CreateAsync(newUser, user.Password);

                if(isCreated.Succeeded)
                {
                    var jwtToken = GenerateJwtToken(newUser);

                    return Ok(new RegistrationResponse(){
                        Success = true,
                        Token = jwtToken
                    });
                } else
                {
                    return BadRequest(new RegistrationResponse(){
                        Errors = isCreated.Errors.Select(x=> x.Description).ToList(),
                        Success = false
                    });
                }

            }
            return BadRequest(new RegistrationResponse(){
                    Errors = new List<string>(){"Invalid Payload"},
                    Success = false
                });
        }

        [HttpPost]
        [Route("Login")]
        public async Task<IActionResult> Login([FromBody] UserLoginRequest user)
        {
            if(ModelState.IsValid)
            {
                var existingUser = await _userManager.FindByEmailAsync(user.Email);

                if(existingUser == null)
                {
                    return BadRequest(new RegistrationResponse(){
                        Errors = new List<string>(){
                            $"Invalid login request"
                        },
                        Success = false
                    });
                }

                var isCorrect = await _userManager.CheckPasswordAsync(existingUser, 
user.Password);

                if(!isCorrect)
                {
                    return BadRequest(new RegistrationResponse(){
                        Errors = new List<string>(){
                            "Invalid login request 2"
                        },
                        Success = false
                    });
                }
                var jwtToken = GenerateJwtToken(existingUser);
                return Ok(new RegistrationResponse(){
                        Success = true,
                        Token = jwtToken
                    });

            }
            return BadRequest(new RegistrationResponse(){
                    Errors = new List<string>(){"Invalid Payload"},
                    Success = false
                });
        }


}
```

7. Testing Register



```
POST https://localhost:5001/api/AuthManagement/register
```

```
{
  "username" : "abc",
  "email" : "abc@gmail.com",
  "password" : "Password12345!"
}
```

```
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJZCI6ImI4YzE5NjI2LWYyYTktNDBjNS04ZmM1LWE2MTMyZWM0Yjg2MSIsImVtYWlsIjoiYWJjQGdtYWlsLmNvbSIsInN1YiI6ImFiY0BnbWFpbC5jb20iLCJqdGkiOiI3MzRhNmY3Zi02YTI5LTRmOTktOGM3YS0yZGRmMTFlYzUwZTEiLCJuYmYiOjE2NDgzOTM5NTcsImV4cCI6MTY0ODQxNTU1NywiaWF0IjoxNjQ4MzkzOTU3fQ.wl__b38L1l9EV40XmagwD1dXdD3jAzD8ge9rDvgLBNM",
  "success": true,
  "errors": null
}
```

## VII. Penambahan Authentication dengan JWT III : Login

1. Pada 'TodoApp>Models>DTOs>Request' tambahkan file 'UserLoginRequest.cs'

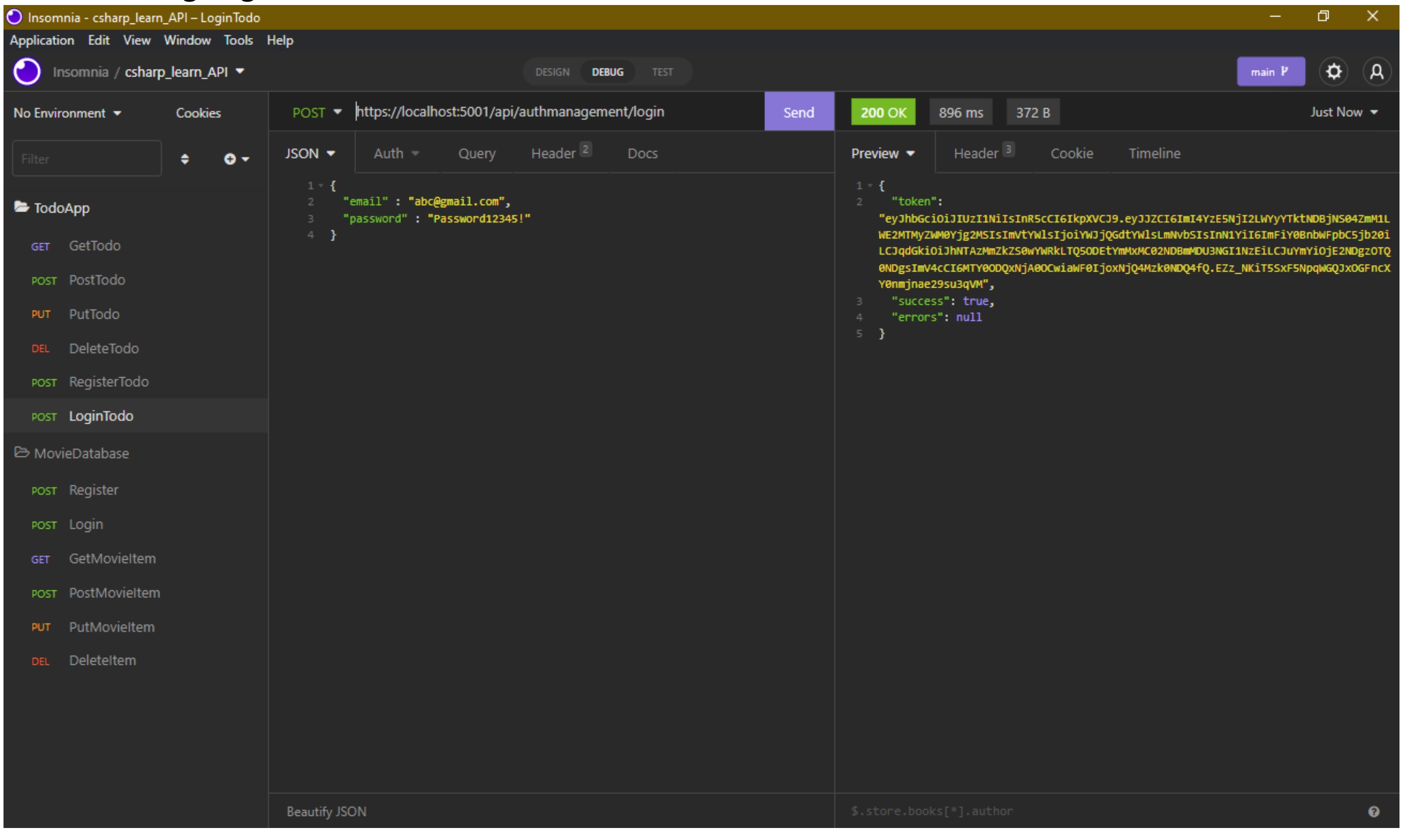
UserLogin.cs:

```
using System.ComponentModel.DataAnnotations;

namespace TodoApp.Models.DTOs.Requests
{
    public class UserLoginRequest
    {
        [Required]
        [EmailAddress]
        public string Email {get;set;}
        [Required]
        public string Password {get;set;}
    }
}
```

2. Pada 'TodoApp>Controller>AuthManagementController' Tambahkan pada bagian bawah register:

```
[HttpPost]
        [Route("Login")]
        public async Task<IActionResult> Login([FromBody] UserLoginRequest user)
        {
            if(ModelState.IsValid)
            {
                var existingUser = await _userManager.FindByEmailAsync(user.Email);
                if(existingUser == null)
                {
                    return BadRequest(new RegistrationResponse(){
                        Errors = new List<string>(){
                            $"Invalid login request"
                        },
                        Success = false
                    });
                }

                var isCorrect = await _userManager.CheckPasswordAsync(existingUser,
user.Password);

                if(!isCorrect)
                {
                    return BadRequest(new RegistrationResponse(){
                        Errors = new List<string>(){
                            "Invalid login request 2"
                        },
                        Success = false
                    });
                }
                var jwtToken = GenerateJwtToken(existingUser);
                return Ok(new RegistrationResponse(){
                        Success = true,
                        Token = jwtToken
                    });

            }
            return BadRequest(new RegistrationResponse(){
                    Errors = new List<string>(){"Invalid Payload"},
                    Success = false
                });
        }

        private string GenerateJwtToken(IdentityUser user)
        {
            var jwtTokenHandler = new JwtSecurityTokenHandler();
            var key = Encoding.ASCII.GetBytes(_jwtConfig.Secret);
            var tokenDescriptor = new SecurityTokenDescriptor
            {
                Subject = new ClaimsIdentity( new []
                {
                    new Claim("Id", user.Id),
                    new Claim(JwtRegisteredClaimNames.Email, user.Email),
                    new Claim(JwtRegisteredClaimNames.Sub, user.Email),
                    new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
                }),
                Expires = DateTime.UtcNow.AddHours(6),
                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key),
SecurityAlgorithms.HmacSha256Signature)
            };

            var token = jwtTokenHandler.CreateToken(tokenDescriptor);
            var jwtToken = jwtTokenHandler.WriteToken(token);

            return jwtToken;
        }
```

3. Testing Login



POST https://localhost:5001/api/authmanagement/login

```
{
  "email" : "abc@gmail.com",
  "password" : "Password12345!"
}
```

```
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJZCI6ImI4YzE5NjI2LWYyYTktNDBjNS04ZmM1LWE2MTMyZWM0Yjg2MSIsImVtYWlsIjoiYWJjQGdtYWlsLmNvbSIsInN1YiI6ImFiY0BnbWFpbC5jb20iLCJqdGkiOiJhNTAzMmZkZS0wYWRkLTQ5ODEtYmMxMC02NDBmMDU3NGI1NzEiLCJuYmYiOjE2NDgzOTQ0NDgsImV4cCI6MTY0ODQxNjA0OCwiaWF0IjoxNjQ4Mzk0NDQ4fQ.EZz_NKiT5SxF5NpqWGQJxOGFncXY0nmjnae29su3qVM",
  "success": true,
  "errors": null
}
```

1. Pada ‘TodoApp>Controller>TodoController.cs’ dibagian atas (include) tambahkan:

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
```

2. Pada ‘TodoApp>Controller>TodoController.cs’ di bawah

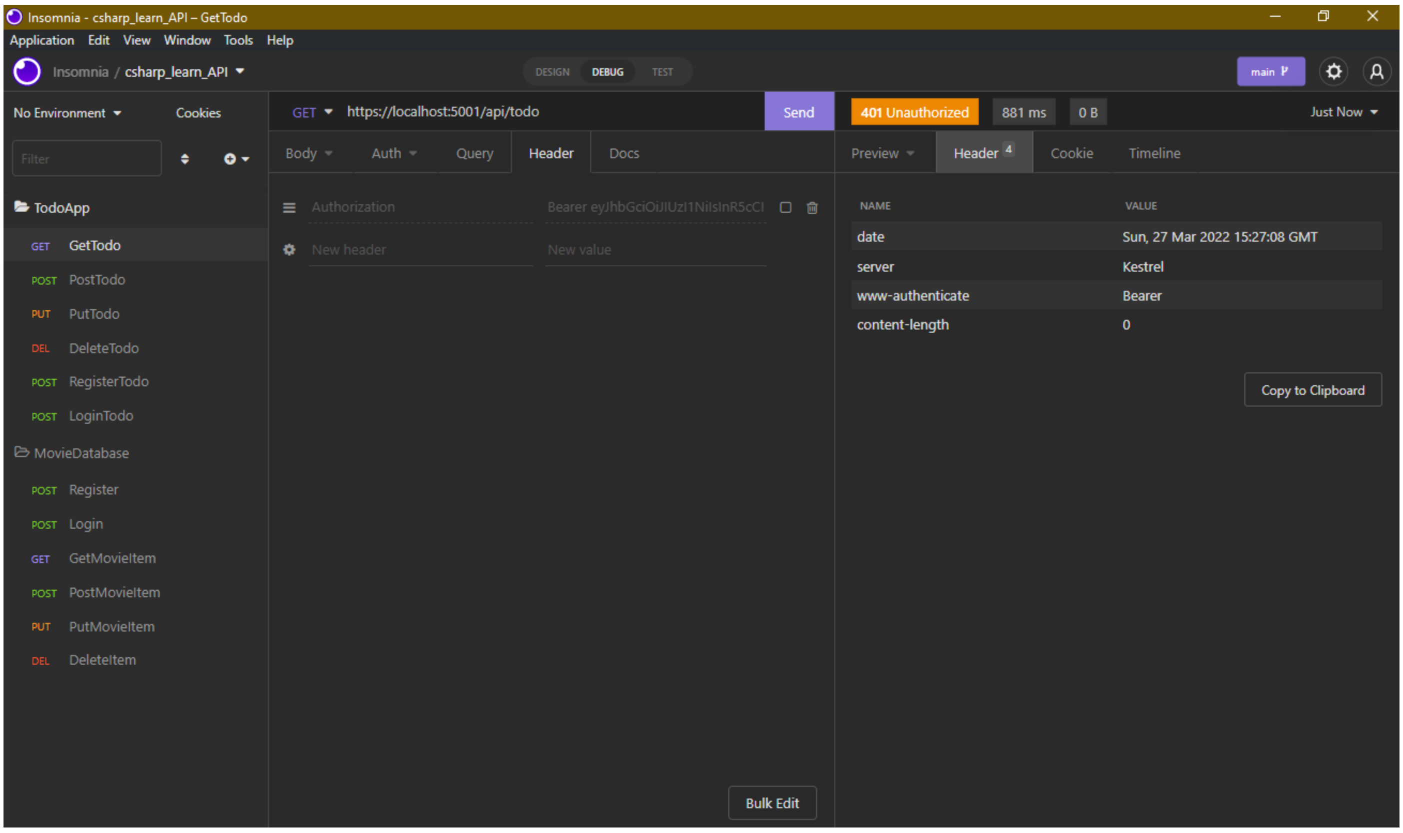
```
[ApiController]
```

tambahkan:

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

3. Hasil testing

Saat tanpa token:



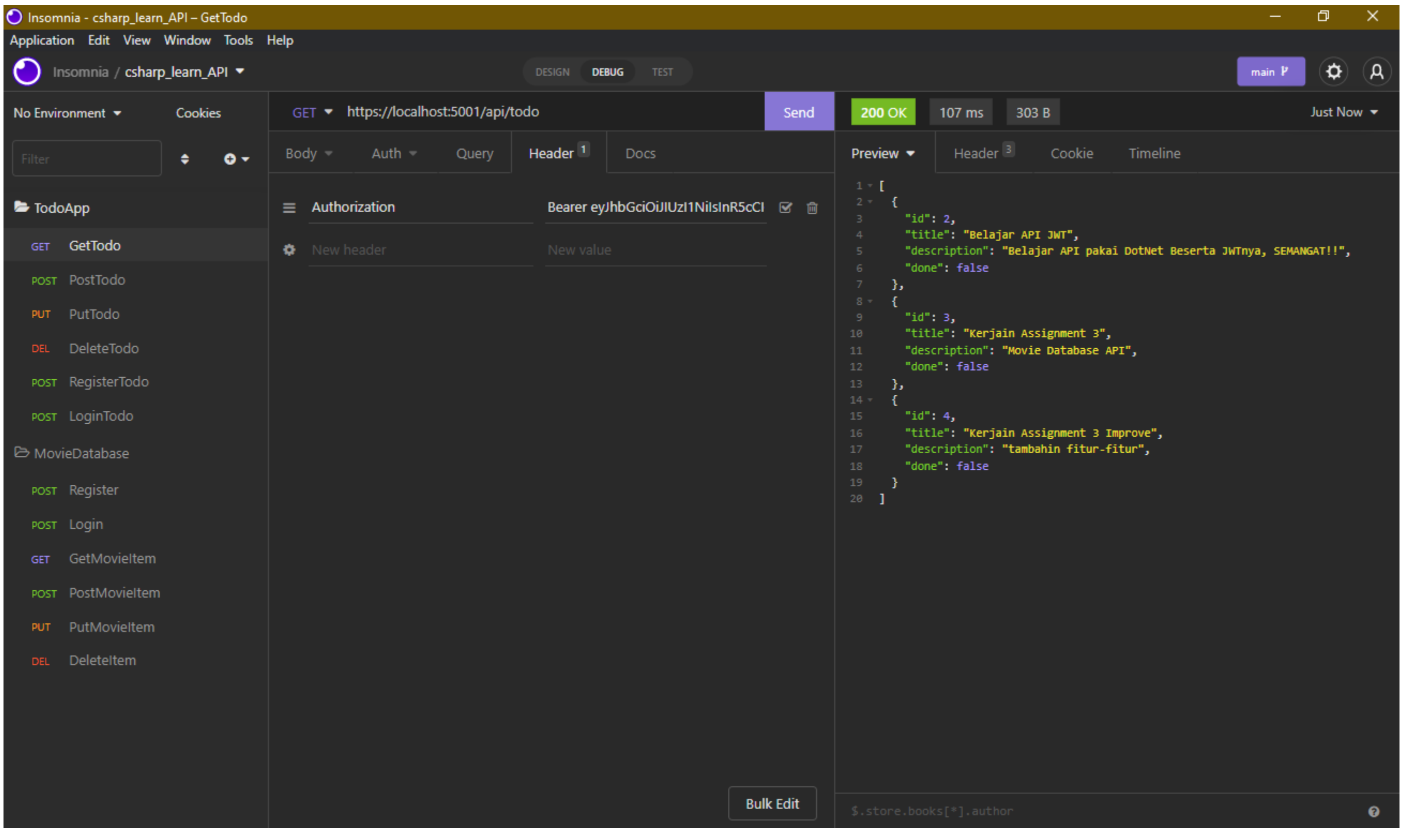
GET https://localhost:5001/api/todo

401 Unauthorized 881 ms 0 B

Authorization Bearer eyJhbGciOiJIUzI1NiIsInR5cCI

| NAME | VALUE |
|---|---|
| date | Sun, 27 Mar 2022 15:27:08 GMT |
| server | Kestrel |
| www-authenticate | Bearer |
| content-length | 0 |

Memakai token:



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

200 OK 107 ms 303 B

Authorization Bearer eyJhbGciOiJIUzI1NiIsInR5cCI

```
[
  {
    "id": 2,
    "title": "Belajar API JWT",
    "description": "Belajar API pakai DotNet Beserta JWTnya, SEMANGAT!!",
    "done": false
  },
  {
    "id": 3,
    "title": "Kerjain Assignment 3",
    "description": "Movie Database API",
    "done": false
  },
  {
    "id": 4,
    "title": "Kerjain Assignment 3 Improve",
    "description": "tambahin fitur-fitur",
    "done": false
  }
]
```

# Best Practice “Create Web API: RESTFUL API”

## ASSIGNMENT 3 – Movie Database

Membuat Web API Movie Database dengan melakukan setiap langkah yang sama dengan Langka pembuatan TodoApp diatas namun dilakukan penggantian nama:

1. Project: MovieDatabase

```
dotnet new webapi -n “MovieDatabase” -lang “C#” -au none
```

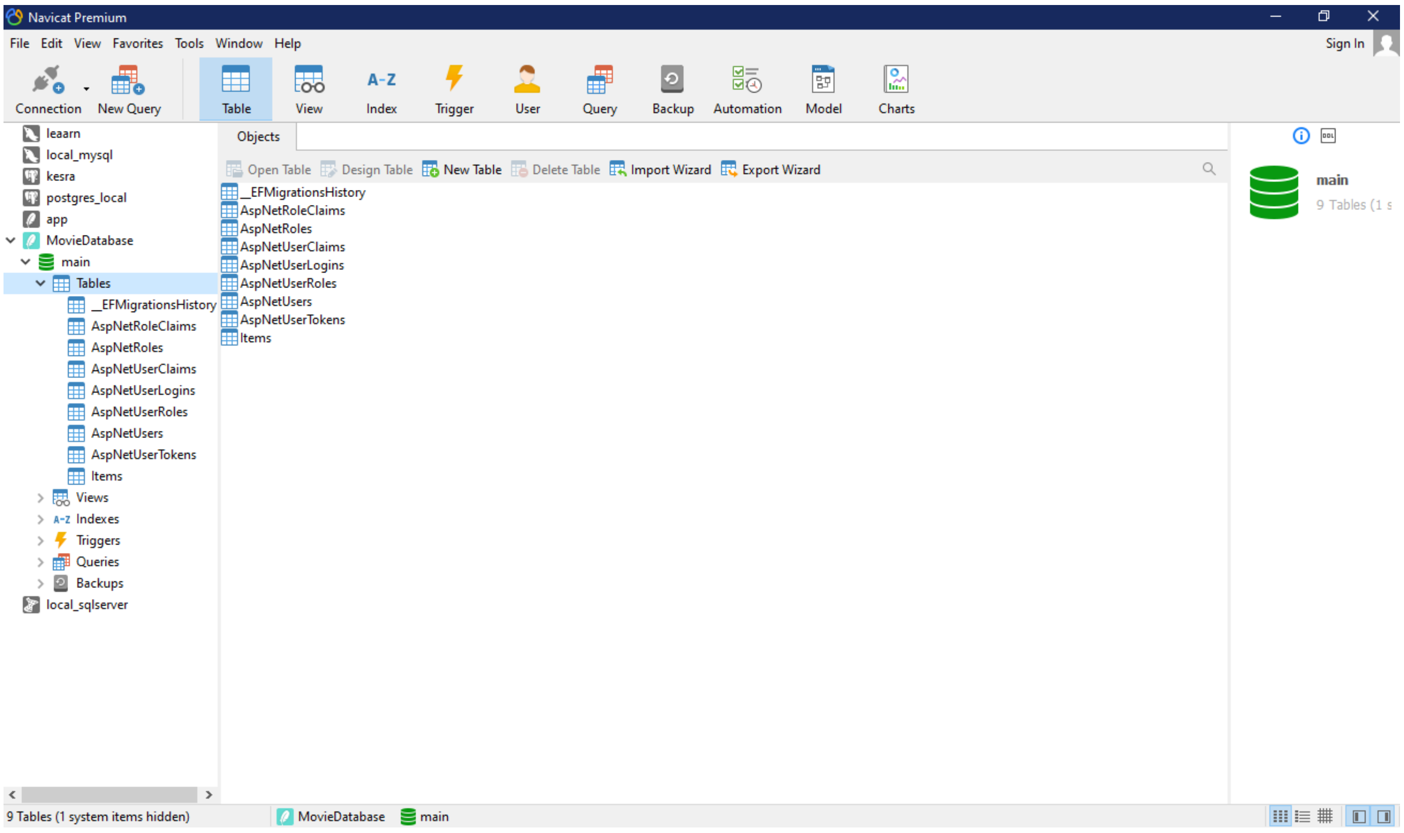
2. Models:
   - ItemData.cs

```
using System;
namespace MovieDatabaseApi.Models
{
    public class ItemData
    {
        public int Id {get; set;}
        public string Name {get; set;}
        public string Genre {get; set;}
        public string Duration {get; set;}
        public DateTime ReleaseDate {get; set;}
    }
}
```

3. Data:
   - ApiDBContext.cs

```
using Microsoft.EntityFrameworkCore;
using TodoApp.Models;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.AspNetCore.Identity;

namespace TodoApp.Data
{
    public class ApiDbContext : IdentityDbContext
    {
        public virtual DbSet<ItemData> Items {get;set;}
        public ApiDbContext(DbContextOptions<ApiDbContext> options):base(options)
        {

        }
    }
}
```

4. Drop Database
   - dotnet ef database drop
5. Remove Migration and re add migrations
   - dotnet ef migrations remove
   - dotnet ef migrations add "Initial Migrations"
6. Update database
   - dotnet ef database update



```
leaarn
local_mysql
kesra
postgres_local
app
MovieDatabase
  main
    Tables
      __EFMigrationsHistory
      AspNetRoleClaims
      AspNetRoles
      AspNetUserClaims
      AspNetUserLogins
      AspNetUserRoles
      AspNetUsers
      AspNetUserTokens
      Items
    Views
    Indexes
    Triggers
    Queries
    Backups
local_sqlserver
```

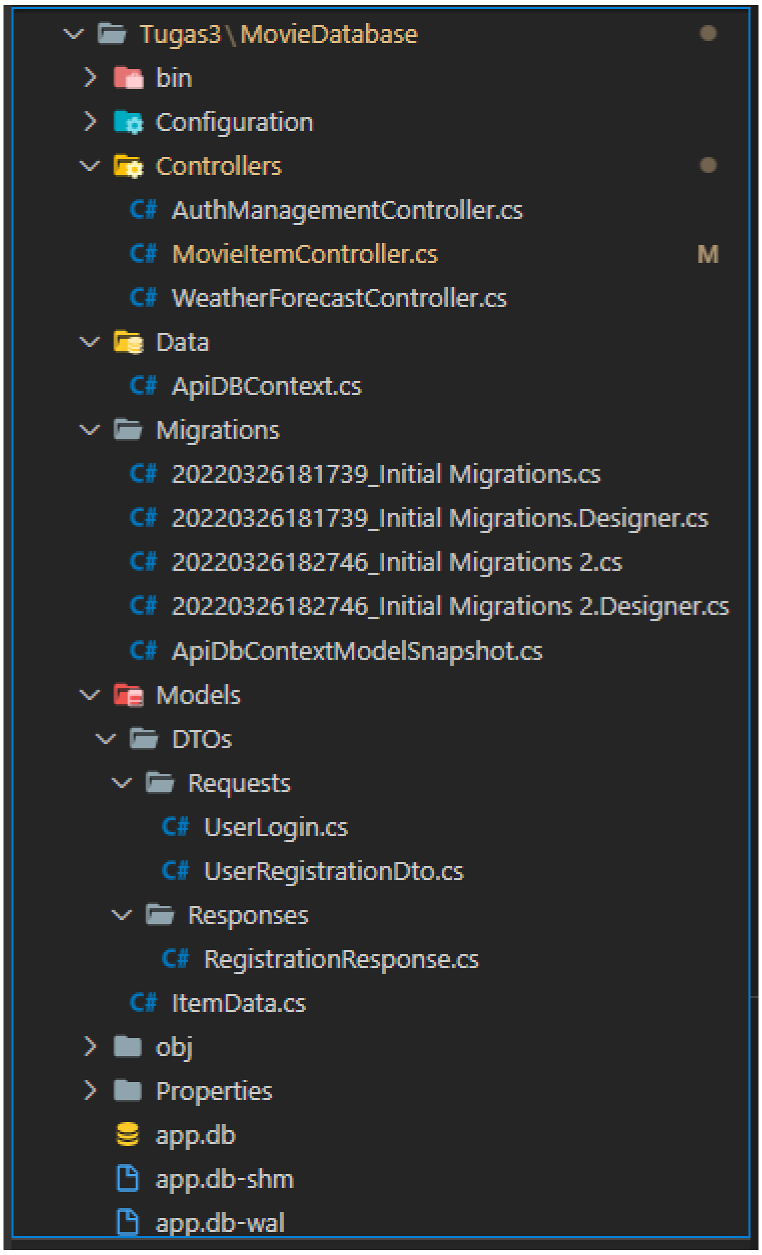
```
__EFMigrationsHistory
AspNetRoleClaims
AspNetRoles
AspNetUserClaims
AspNetUserLogins
AspNetUserRoles
AspNetUsers
AspNetUserTokens
Items
```

main
9 Tables (1 s

9 Tables (1 system items hidden)   MovieDatabase   main

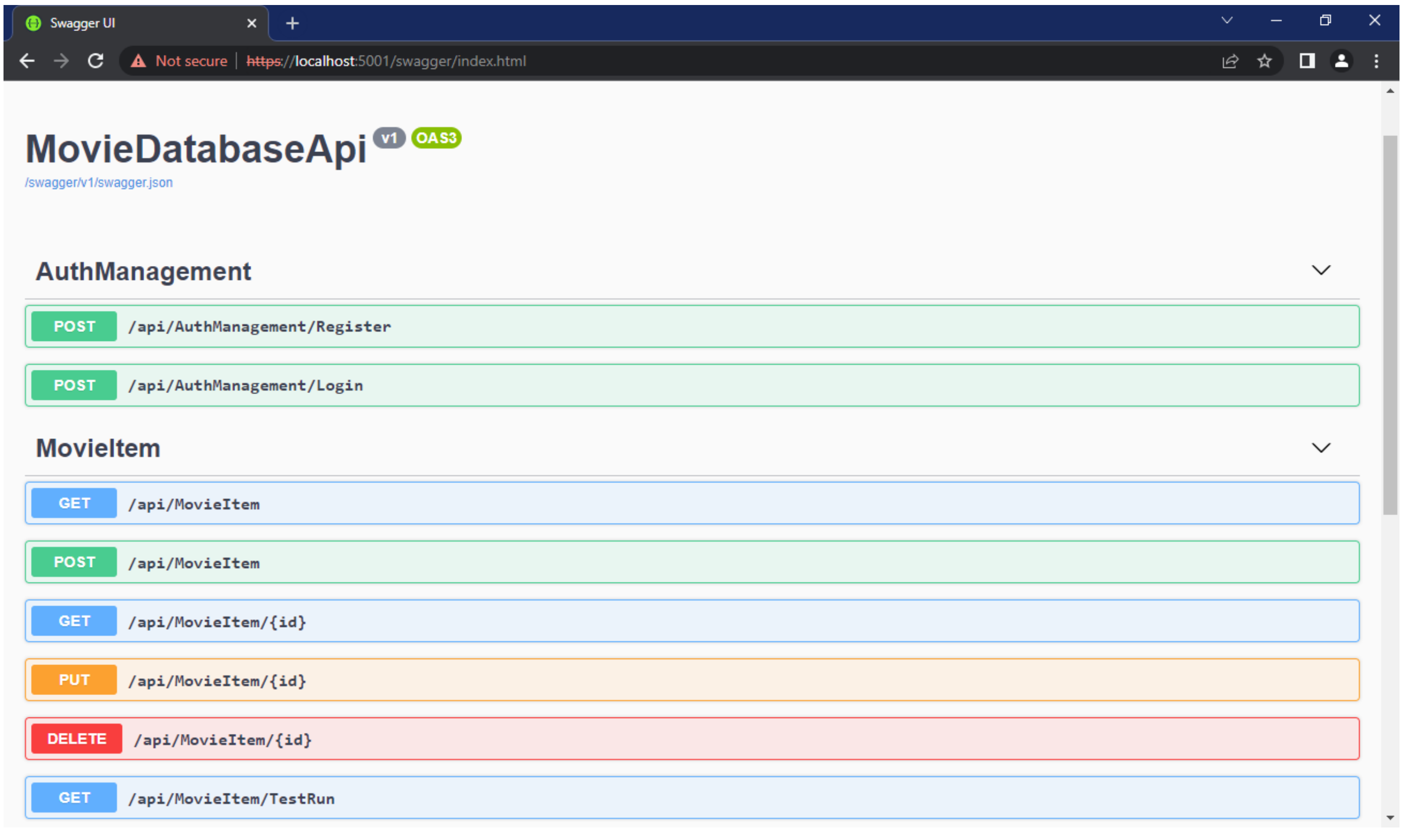
7. Controller:
   - MovieItemController.cs, dengan syntax yang telah disesuaikan pada bagian PUT, karena ada perbedaan field data:

```
        [HttpPut("{id}")]
        public async Task<IActionResult> UpdateItem(int id, ItemData item)
        {
            if(id!=item.Id){
                return BadRequest();
            }
            var existItem = await _context.Items.FirstOrDefaultAsync(x=> x.Id == id);

            if(existItem == null)
                return NotFound();

            existItem.Name = item.Name;
            existItem.Genre = item.Genre;
            existItem.Duration = item.Duration;
            existItem.ReleaseDate = item.ReleaseDate;


            await _context.SaveChangesAsync();
            // return NoContent(existItem);
            return Ok(existItem);


        }
```



```
Tugas3 \ MovieDatabase
  bin
  Configuration
  Controllers
    AuthManagementController.cs
    MovieItemController.cs                M
    WeatherForecastController.cs
  Data
    ApiDBContext.cs
  Migrations
    20220326181739_Initial Migrations.cs
    20220326181739_Initial Migrations.Designer.cs
    20220326182746_Initial Migrations 2.cs
    20220326182746_Initial Migrations 2.Designer.cs
    ApiDbContextModelSnapshot.cs
  Models
    DTOs
      Requests
        UserLogin.cs
        UserRegistrationDto.cs
      Responses
        RegistrationResponse.cs
    ItemData.cs
  obj
  Properties
  app.db
  app.db-shm
  app.db-wal
```

8. Hasil Running:
   - Running Awal memakai swagger



MovieDatabaseApi v1 OAS3

/swagger/v1/swagger.json

AuthManagement

POST /api/AuthManagement/Register

POST /api/AuthManagement/Login

MovieItem

GET /api/MovieItem

POST /api/MovieItem

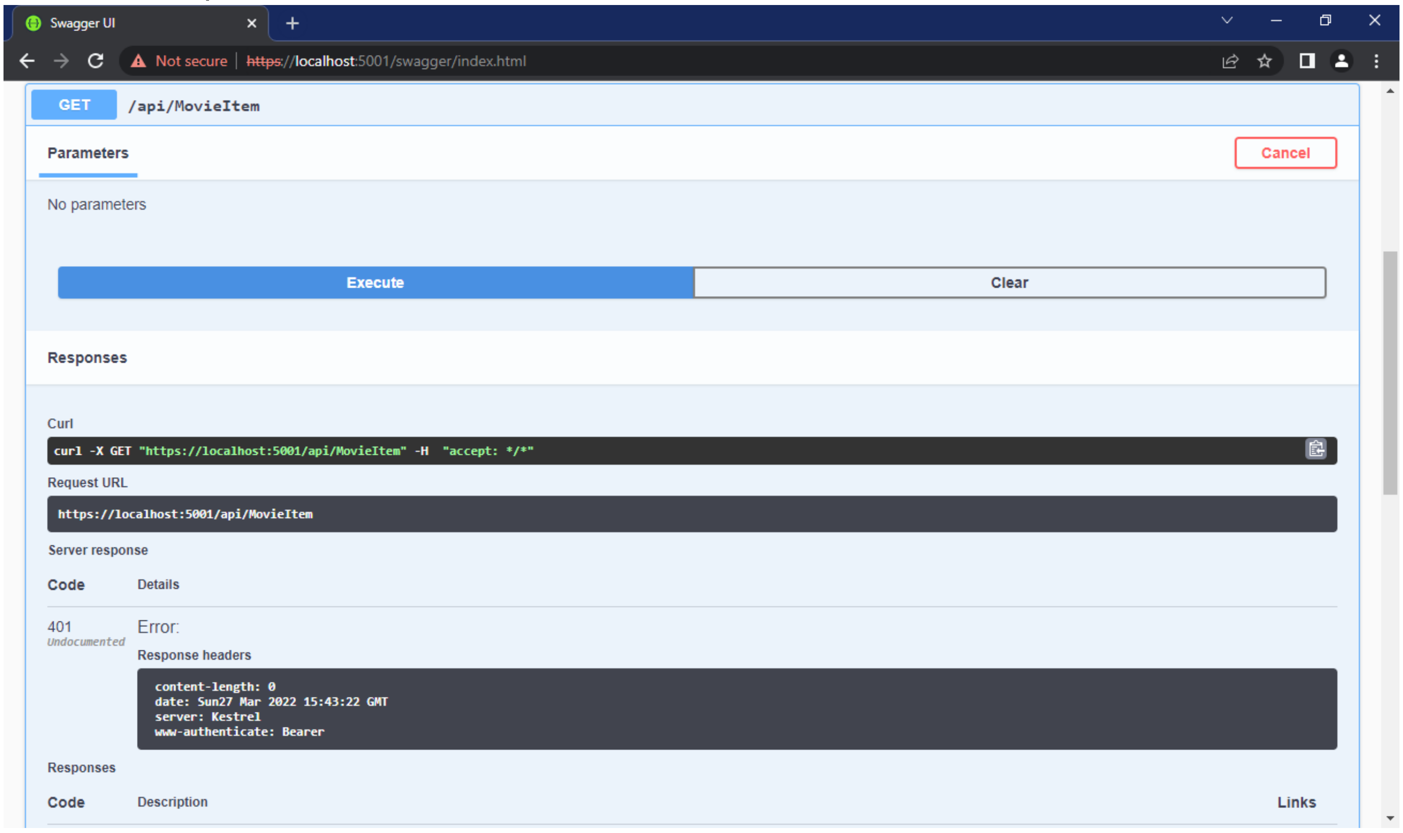
GET /api/MovieItem/{id}

PUT /api/MovieItem/{id}

DELETE /api/MovieItem/{id}

GET /api/MovieItem/TestRun

- GET tanpa token



GET /api/MovieItem

Parameters

No parameters

Responses

Curl

```
curl -X GET "https://localhost:5001/api/MovieItem" -H  "accept: */*"
```

Request URL

```
https://localhost:5001/api/MovieItem
```

Server response

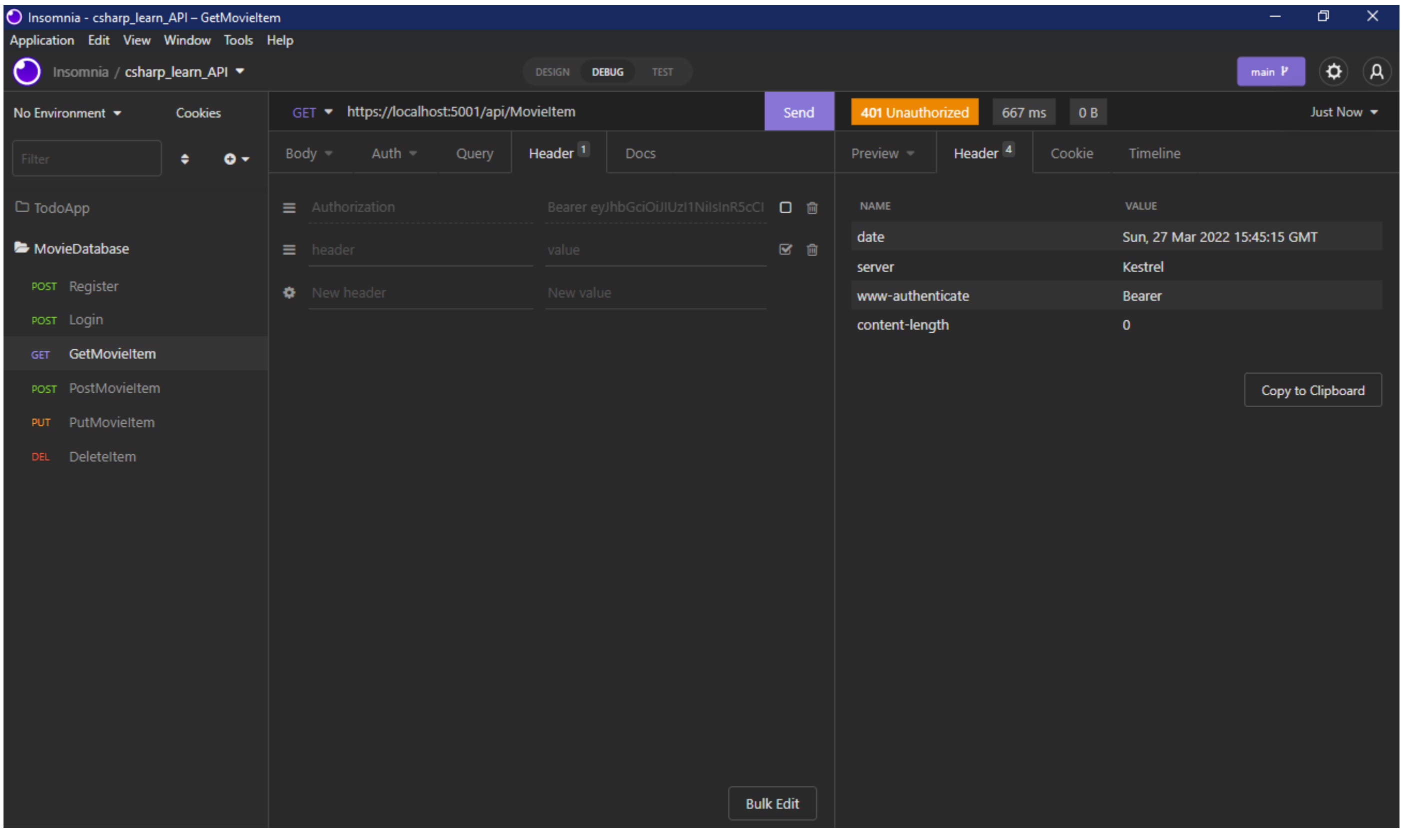
| Code | Details |
|---|---|
| 401 *Undocumented* | Error: |

Response headers

```
content-length: 0
date: Sun27 Mar 2022 15:43:22 GMT
server: Kestrel
www-authenticate: Bearer
```

Responses

| Code | Description | Links |
|---|---|---|


Insomnia - csharp_learn_API – GetMovieItem

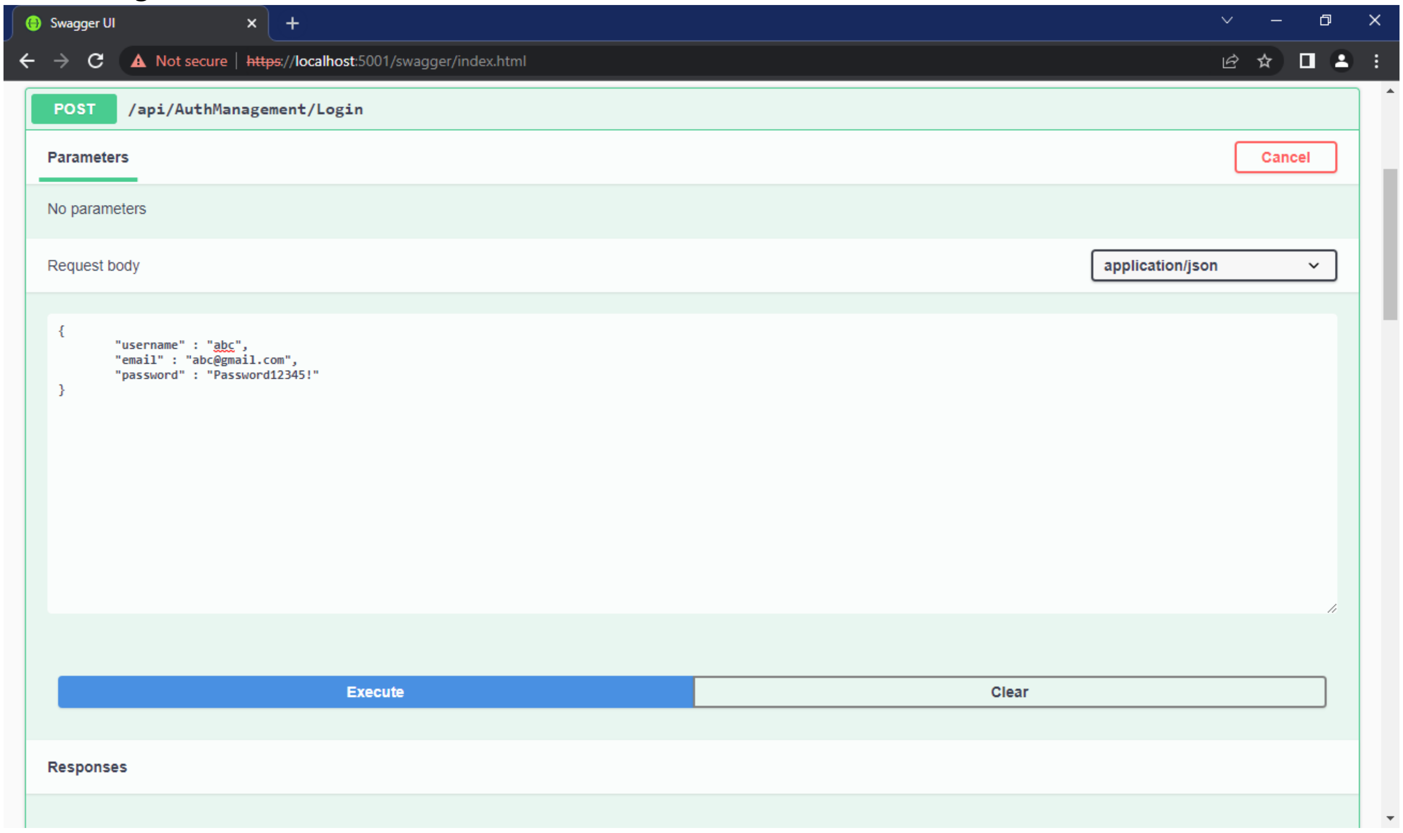
GET https://localhost:5001/api/MovieItem

| | |
|---|---|
| Authorization | Bearer eyJhbGciOiJIUzI1NiIsInR5cCI |
| header | value |
| New header | New value |

401 Unauthorized 667 ms 0 B

| NAME | VALUE |
|---|---|
| date | Sun, 27 Mar 2022 15:45:15 GMT |
| server | Kestrel |
| www-authenticate | Bearer |
| content-length | 0 |

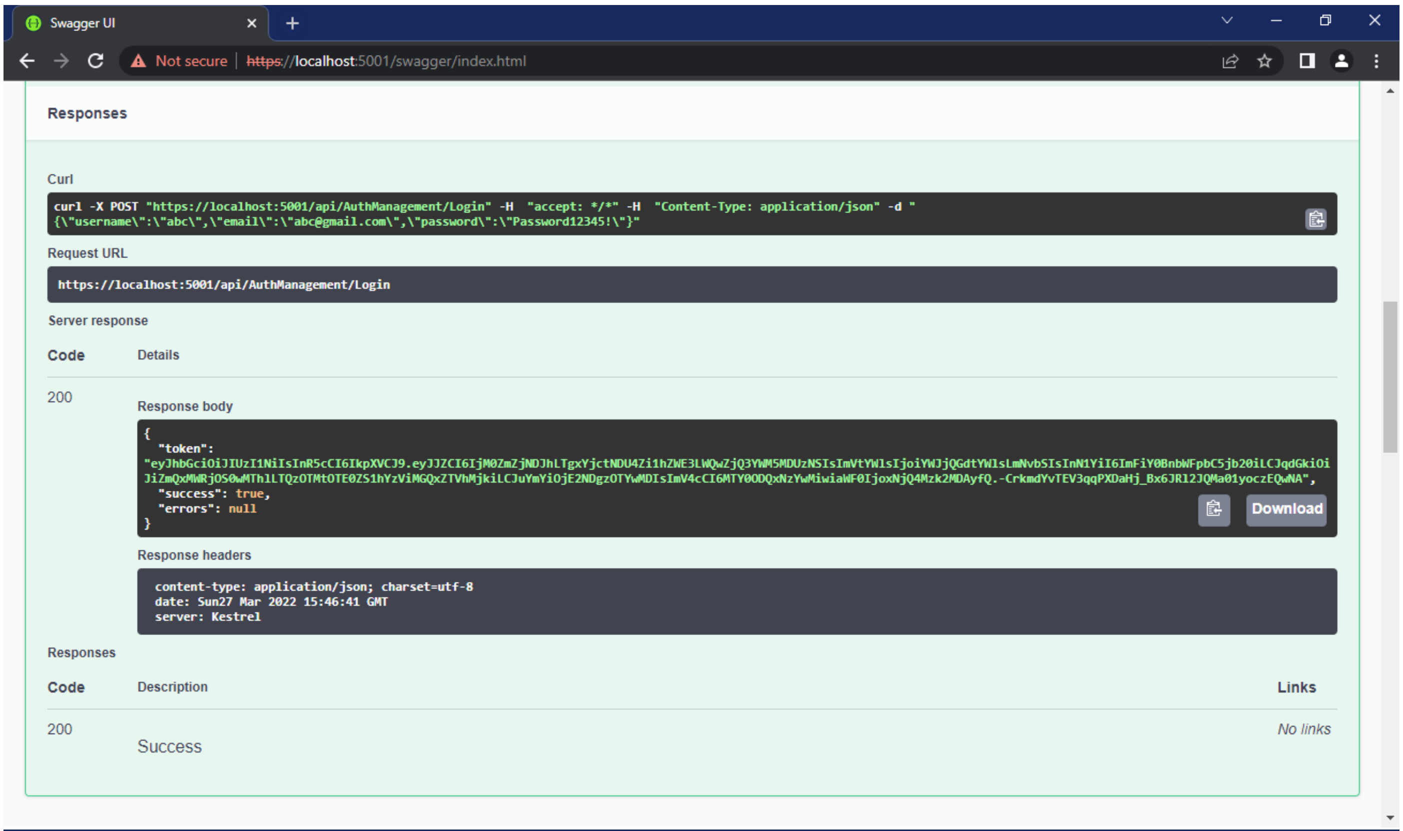
- Login


POST /api/AuthManagement/Login

No parameters

Request body application/json

```
{
        "username" : "abc",
        "email" : "abc@gmail.com",
        "password" : "Password12345!"
}
```



Responses

Curl

```
curl -X POST "https://localhost:5001/api/AuthManagement/Login" -H  "accept: */*" -H  "Content-Type: application/json" -d "
{\"username\":\"abc\",\"email\":\"abc@gmail.com\",\"password\":\"Password12345!\"}"
```

Request URL

```
https://localhost:5001/api/AuthManagement/Login
```

Server response

| Code | Details |
|---|---|
| 200 | Response body |

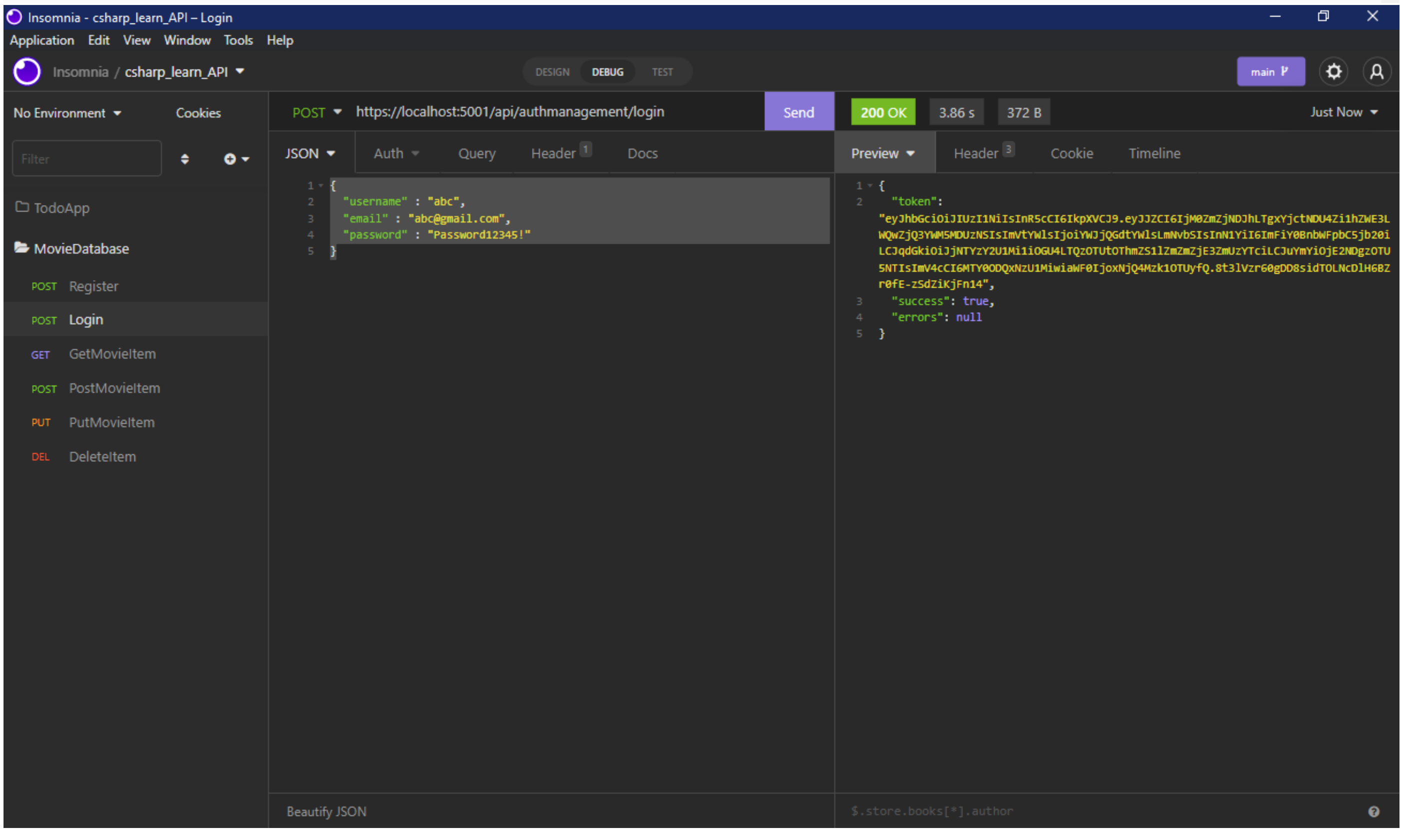
```
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJZCI6IjM0ZmZjNDJhLTgxYjctNDU4Zi1hZWE3LWQwZjQ3YWM5MDUzNSIsImVtYWlsIjoiYWJjQGdtYWlsLmNvbSIsInN1YiI6ImFiY0BnbWFpbC5jb20iLCJqdGkiOi
JiZmQxMWRjOS0wMTh1LTQzOTMtOTE0ZS1hYzViMGQxZTVhMjkiLCJuYmYiOjE2NDgzOTYwMDIsImV4cCI6MTY0ODQxNzYwMiwiaWF0IjoxNjQ4Mzk2MDAyfQ.-CrkmdYvTEV3qqPXDaHj_Bx6JR12JQMa01yoczEQwNA",
  "success": true,
  "errors": null
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Sun27 Mar 2022 15:46:41 GMT
server: Kestrel
```

Responses

| Code | Description | Links |
|---|---|---|
| 200 | Success | *No links* |

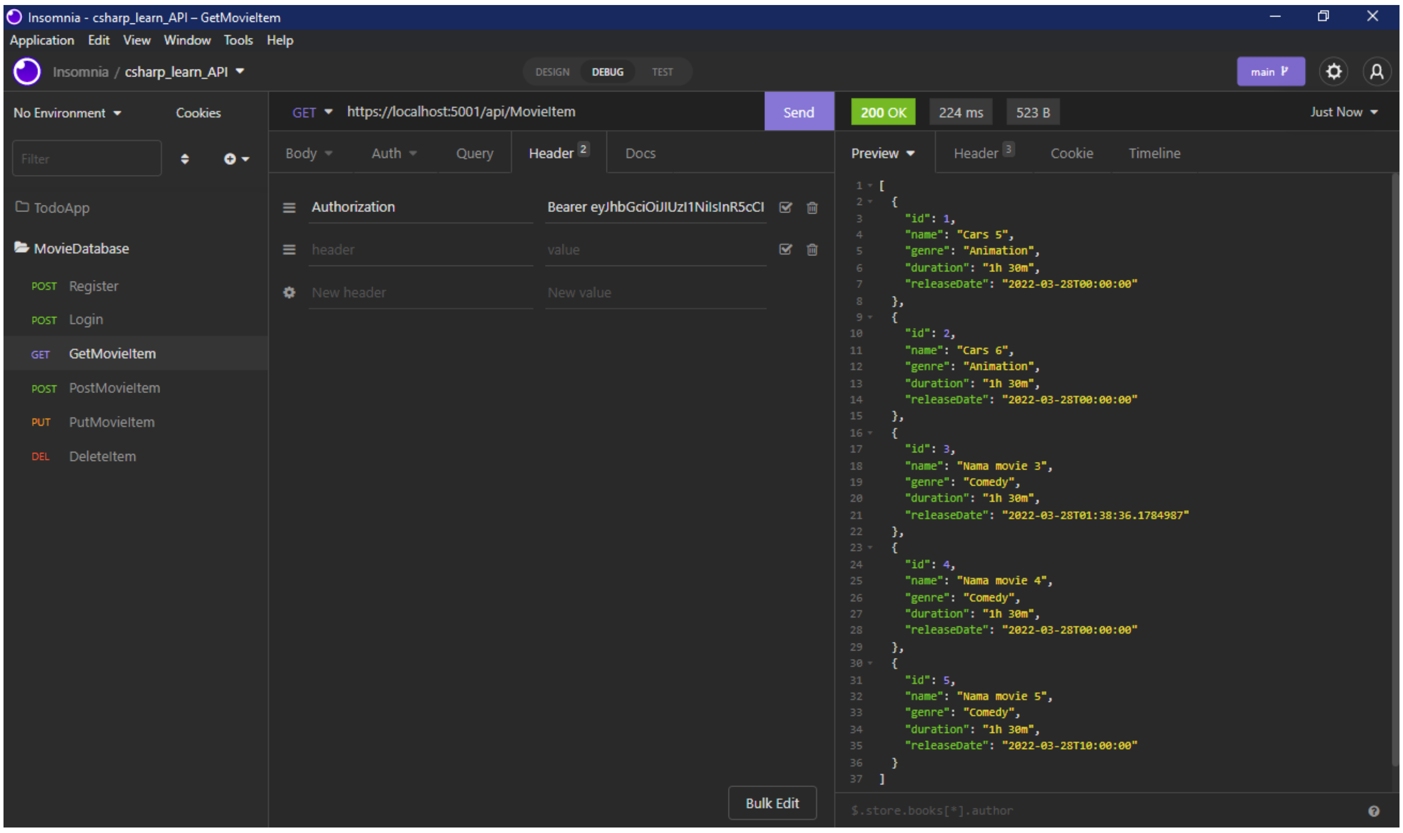


POST https://localhost:5001/api/authmanagement/login

```
{
  "username" : "abc",
  "email" : "abc@gmail.com",
  "password" : "Password12345!"
}
```

```
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJZCI6IjM0ZmZjNDJhLTgxYjctNDU4Zi1hZWE3L
WQwZjQ3YWM5MDUzNSIsImVtYWlsIjoiYWJjQGdtYWlsLmNvbSIsInN1YiI6ImFiY0BnbWFpbC5jb20i
LCJqdGkiOiJjNTYzY2U1Mi1iOGU4LTQzOTUtOThmZS1lZmZmZjE3ZmUzYTciLCJuYmYiOjE2NDgzOTU
5NTIsImV4cCI6MTY0ODQxNzU1MiwiaWF0IjoxNjQ4Mzk1OTUyfQ.8t3lVzr60gDD8sidTOLNcDlH6BZ
r0fE-zSdZiKjFn14",
  "success": true,
  "errors": null
}
```

- GET dengan token



GET https://localhost:5001/api/MovieItem

| | |
|---|---|
| Authorization | Bearer eyJhbGciOiJIUzI1NiIsInR5cCI |
| header | value |
| New header | New value |

200 OK 224 ms 523 B

```
[
  {
    "id": 1,
    "name": "Cars 5",
    "genre": "Animation",
    "duration": "1h 30m",
    "releaseDate": "2022-03-28T00:00:00"
  },
  {
    "id": 2,
    "name": "Cars 6",
    "genre": "Animation",
    "duration": "1h 30m",
    "releaseDate": "2022-03-28T00:00:00"
  },
  {
    "id": 3,
    "name": "Nama movie 3",
    "genre": "Comedy",
    "duration": "1h 30m",
    "releaseDate": "2022-03-28T01:38:36.1784987"
  },
  {
    "id": 4,
    "name": "Nama movie 4",
    "genre": "Comedy",
    "duration": "1h 30m",
    "releaseDate": "2022-03-28T00:00:00"
  },
  {
    "id": 5,
    "name": "Nama movie 5",
    "genre": "Comedy",
    "duration": "1h 30m",
    "releaseDate": "2022-03-28T10:00:00"
  }
]
```