Nama : Alexander Radianta Tarigan

No\_Peserta : FSDO003ONL010

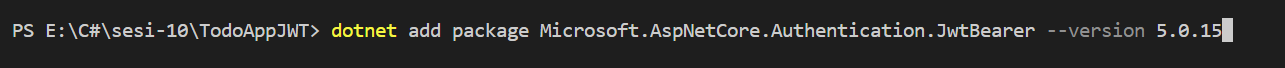
Jalankan perintah dotnet new webapi -n “TodoAppJWT” -lang “C#” -au none pada terminal untuk membuat WebAPI

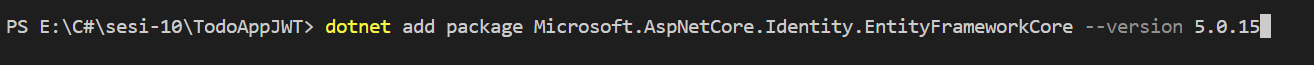


Install package yang dibutuhkan



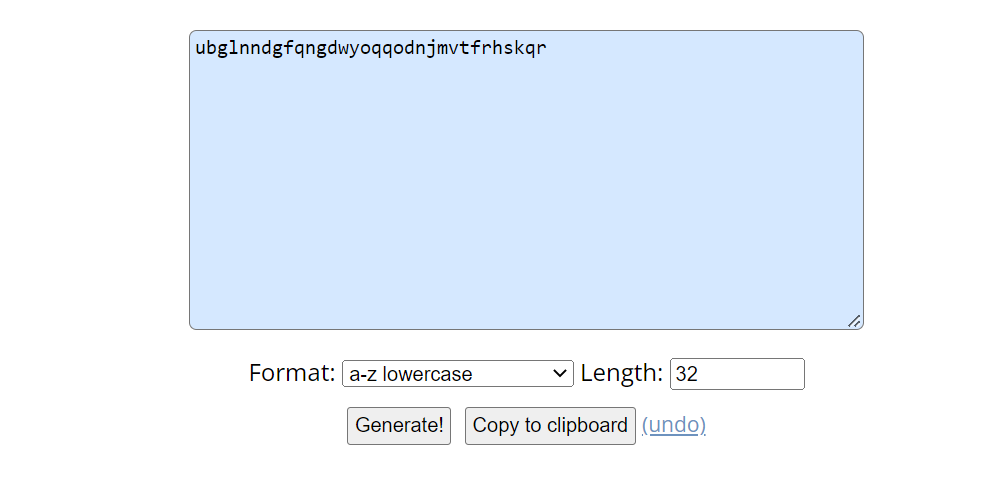








Lalu kunjungi <https://www.browserling.com/tools/random-string> untuk mendapatkan randomstring,gunakan panjang string 32, lalu klik generate



Lalu buka appsetings.json, update setting dengan menambahkan JWT setting, dimana memerlukan secret token yang sebelumnya sudah digenerete melalui randomstring.

{

  "ConnectionStrings": {

    "DefaultConnection": "DataSource=app.db;Cache=Shared"

  },

  "Logging": {

    "LogLevel": {

      "Default": "Information",

      "Microsoft": "Warning",

      "Microsoft.Hosting.Lifetime": "Information"

    }

  },

  "JwtConfig": {

    "Secret": "ubglnndgfqngdwyoqqodnjmvtfrhskqr"

  },

  "AllowedHosts": "\*"

}

Selanjutnya buat class baru pada folder Configuration dan beri nama JwtConfig.cs

public class JwtConfig

    {

        public string Secret {get; set;}

    }

Lalu pada Startup.cs tambahkan code di method ConfigureServices untuk inject JwtConfiguration nantinya.

public void ConfigureServices(IServiceCollection services)

        {

services.Configure<JwtConfig>(Configuration.GetSection("JwtConfig"));

Tambahkan konfigurasi pada class startup pada konfigurasi core ASP.NET middleware dab IOC Container.Tambahkan Method Configure dengan menambahkan Authentikasi

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>();

Tambahkan Method Configure dengan menambahkan Authentikasi

app.UseAuthentication();

Langkah selanjutnya adalah mengupdate ApiDbContext dengan memanfaatkan Identity provider yang disediakan Asp.Net , untuk dapat dinavigasi ke ApiDbContext di folder Data dan update class ApiDbContext

public class ApiDbContext : IdentityDbContext

    {

    }

Lakukan migrasi



Lalu lakukan update database



Langkah Selanjutnya adalah set-up controllers dan build proses registrasi untuk User, Buat file baru bernama : AuthResult.cs pada folder Configurations

public class AuthResult

    {

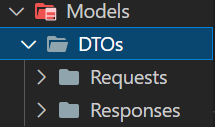
        public string Token {get; set;}

        public bool Success {get; set;}

        public List<string> Errors {get; set;}

    }

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



Lalu tambahkan class baru pada folder requests, dan beri nama UserRegistrationDto.cs

public class UserRegistrationDto

    {

        [Required]

        public string Username {get; set;}

        [Required]

        [EmailAddress]

        public string Email {get; set;}

        [Required]

        public string Password {get; set;}

    }

Lalu tambahkan class baru pada folder Response, dan beri nama RegistrationResponse.cs

namespace TodoAppJWT.Models.DTOs.Responses

{

    public class RegistrationResponse : AuthResult

    {

    }

}

Tambahkan controller user registration pada folder Controller dengan membuat file class baru bernama AuthManagementController.cs

namespace TodoAppJWT.Controllers

{

    [Route("api/[controller]")]

    [ApiController]

    public class AuthManagementController : ControllerBase

    {

        private readonly UserManager<IdentityUser> \_userManager;

        private readonly JwtConfig \_jwtConfig;

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

        {

            \_userManager = userManager;

            \_jwtConfig = optionsMonitor.CurrentValue;

        }

        [HttpPost]

        [Route("Register")]

        public async Task<IActionResult> Register([FromBody] UserRegistrationDto user)

        {

            if(ModelState.IsValid)

            {

                var existingUser = await \_userManager.FindByEmailAsync(user.Email);

                if(existingUser != null)

                {

                    return BadRequest(new RegistrationResponse(){

                        Errors = new List<string>(){

                            "Email Telah digunakan"

                        },

                        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

            });

        }

        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;

        }

    }

}

Lalu buat class baru di folder Models/DTOs/Requests dan beri nama UserLoginRequest.cs

public class UserLoginRequest

    {

        [Required]

        [EmailAddress]

        public string Email {get; set;}

        [Required]

        public string Password {get; set;}

    }

Lalu buat action login di AuthManagementController.cs

[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"

                        },

                        Success= false

                    });

                }

                var jwtToken = GenerateJwtToken(existingUser);

                return Ok(new RegistrationResponse(){

                    Success = true,

                    Token = jwtToken

                });

            }

            return BadRequest(new RegistrationResponse(){

                Errors = new List<string>(){

                    "Invalid payload"

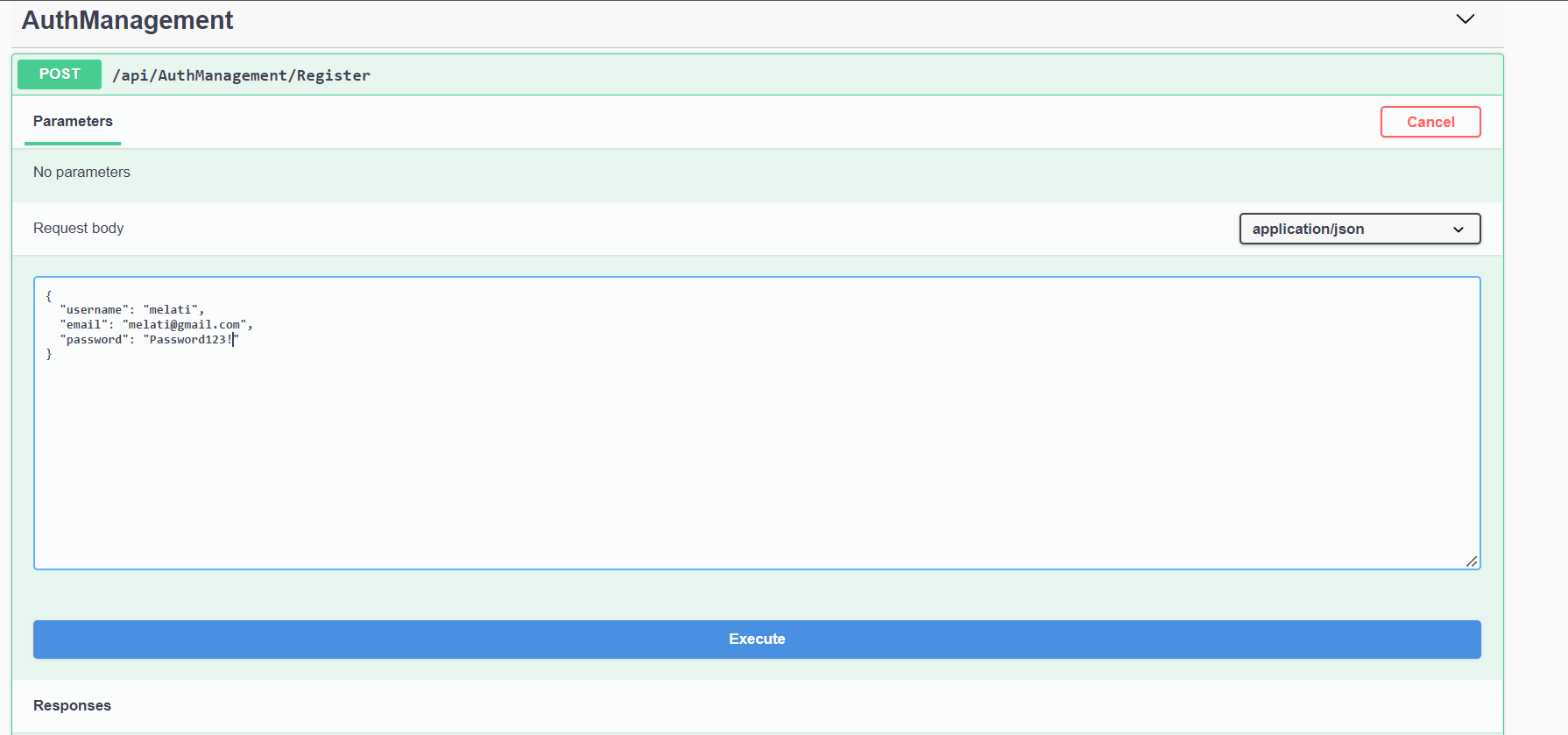
                },

                Success = false

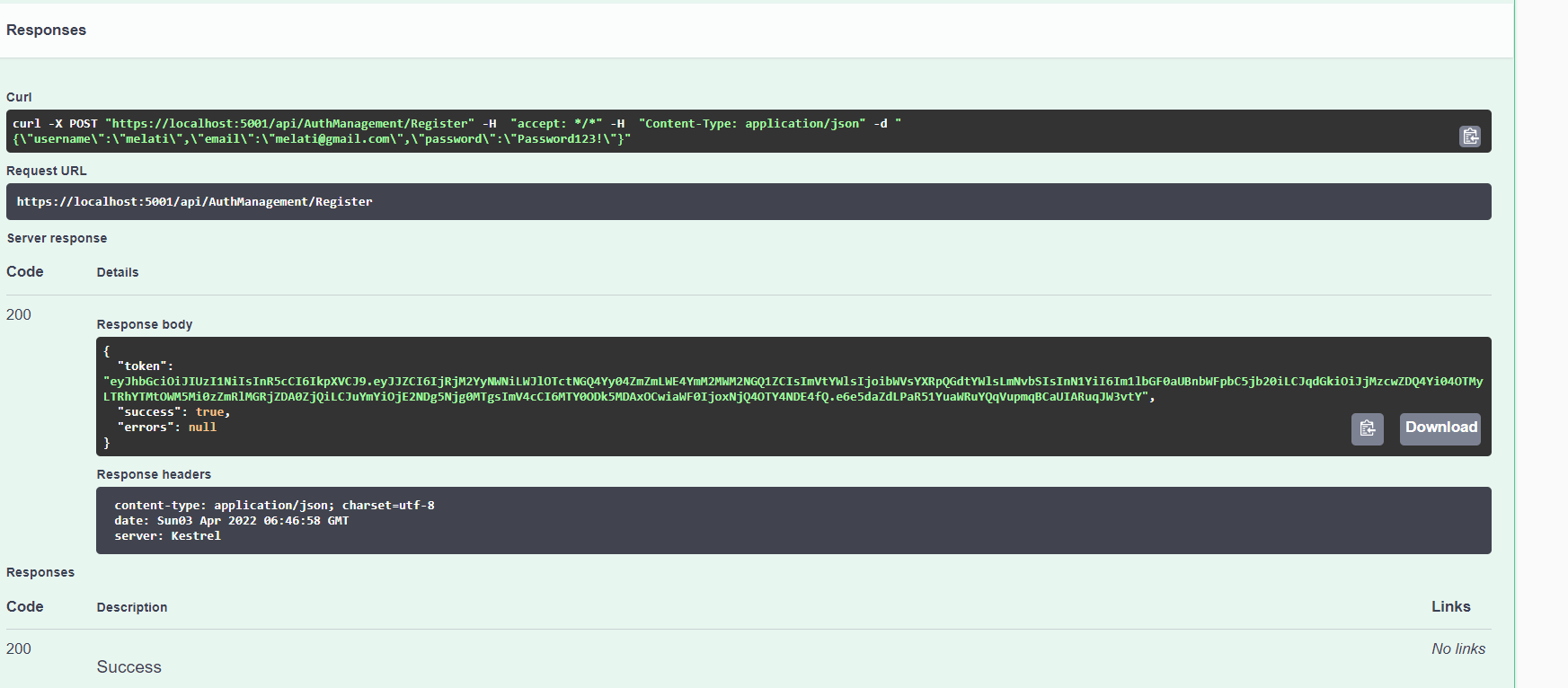
            });

        }

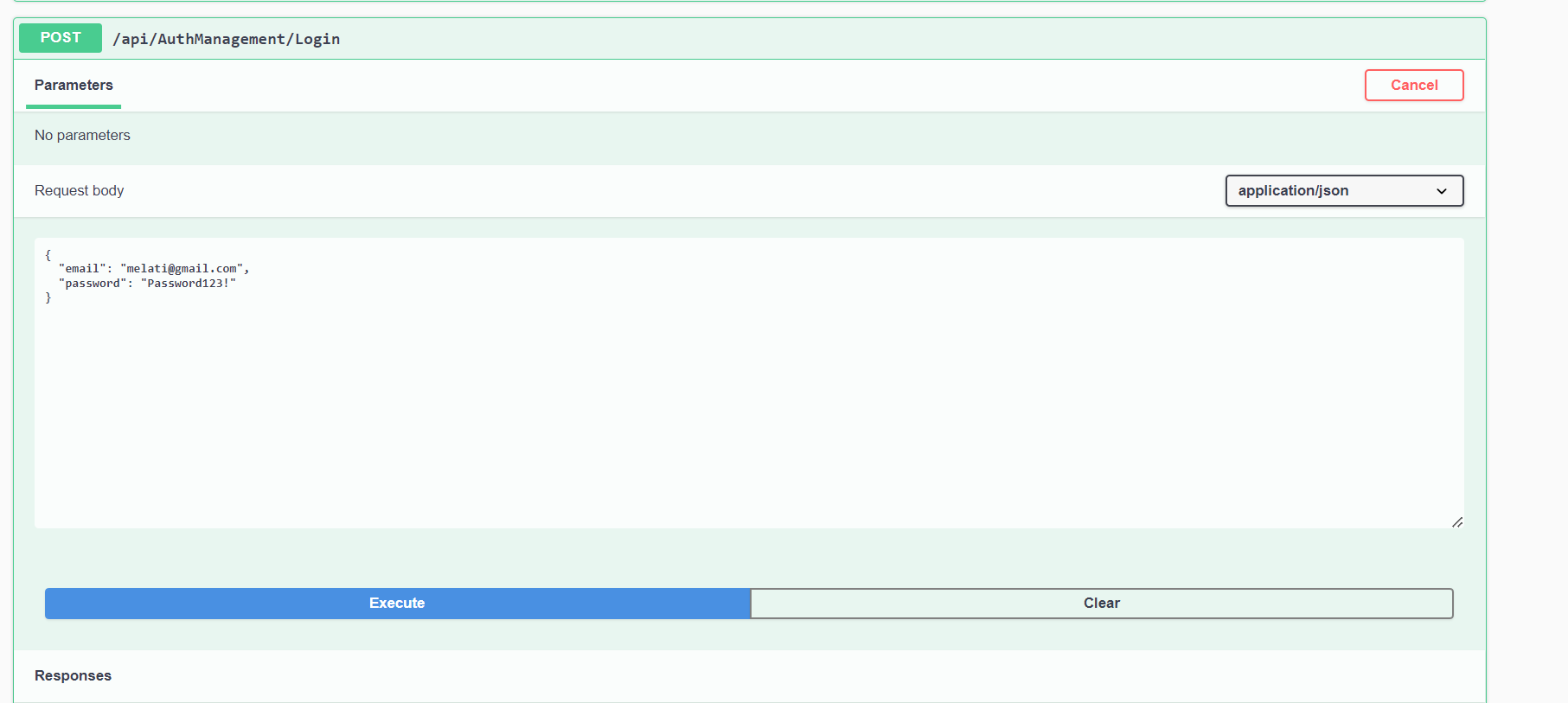
Lalu jalankan, dan buka https://localhost:5001/swagger/index.html



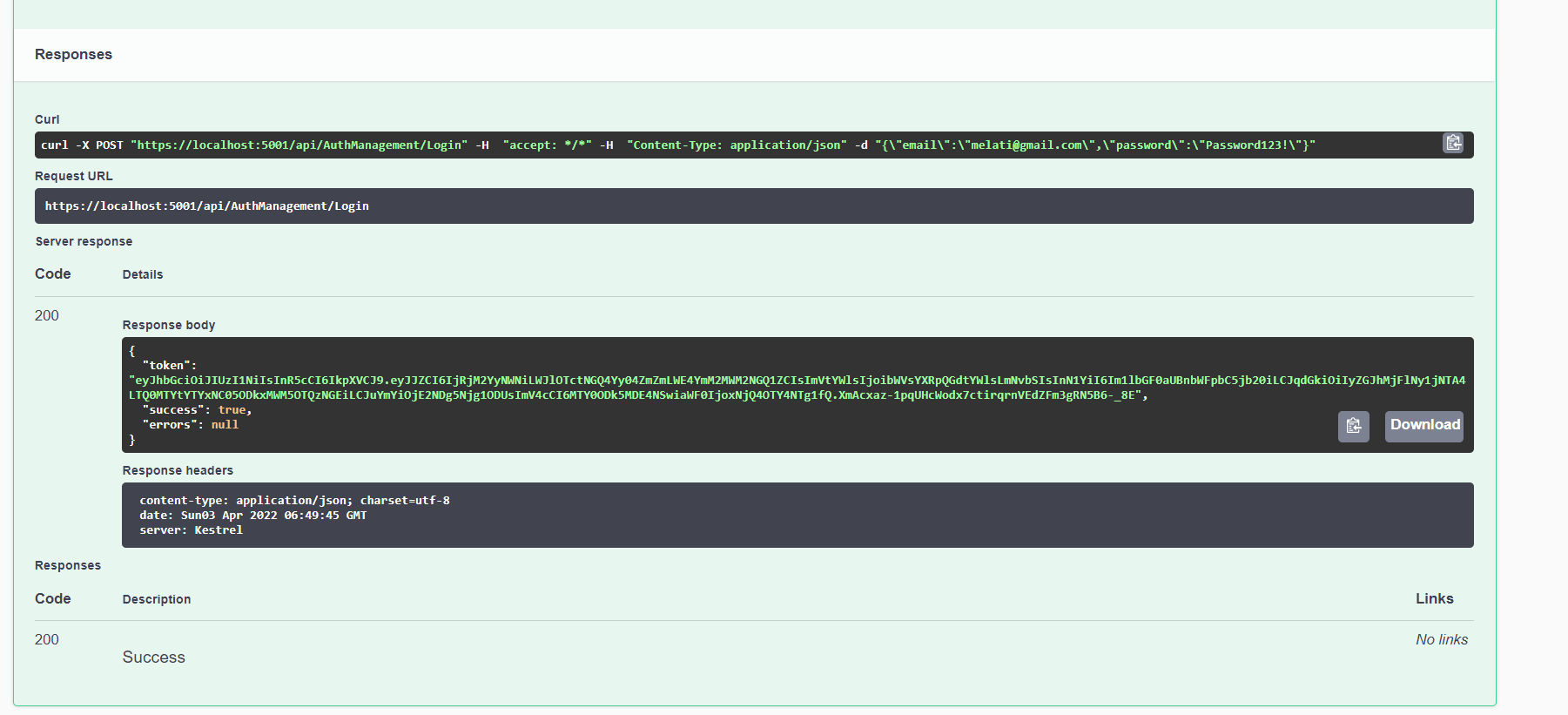
Coba daftarkan sebuah akun melalui controller AuthManagement, lalu klik execute



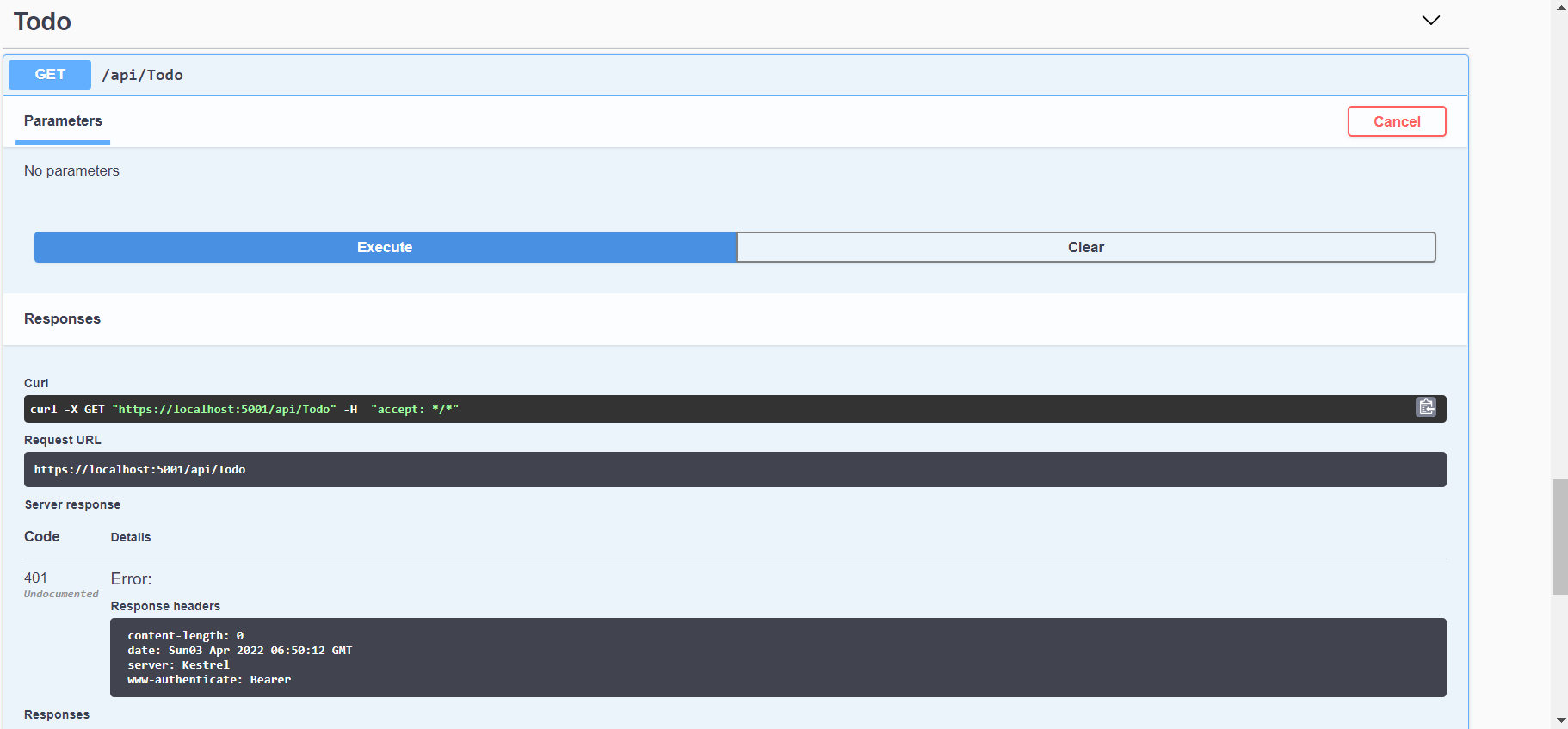
Coba Login menggunakan akun yang baru didaftarkan



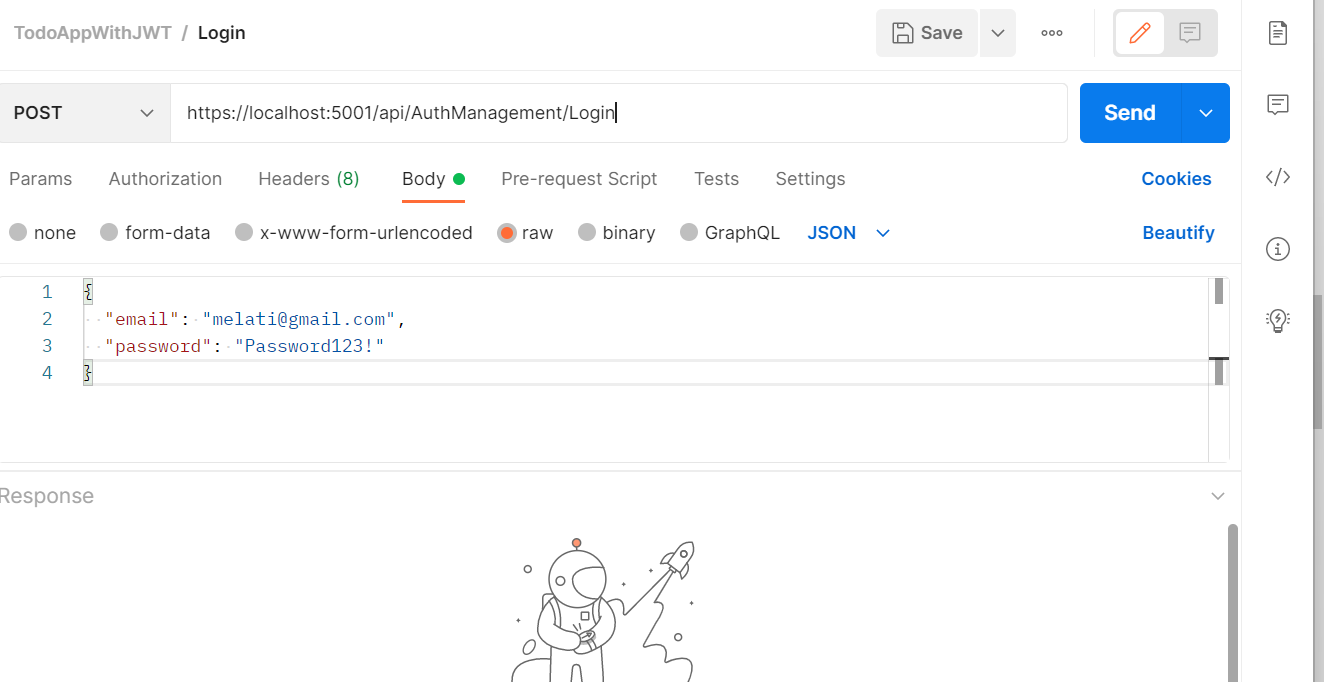
Dan swagger akan meresponse sukses dan mereturn token



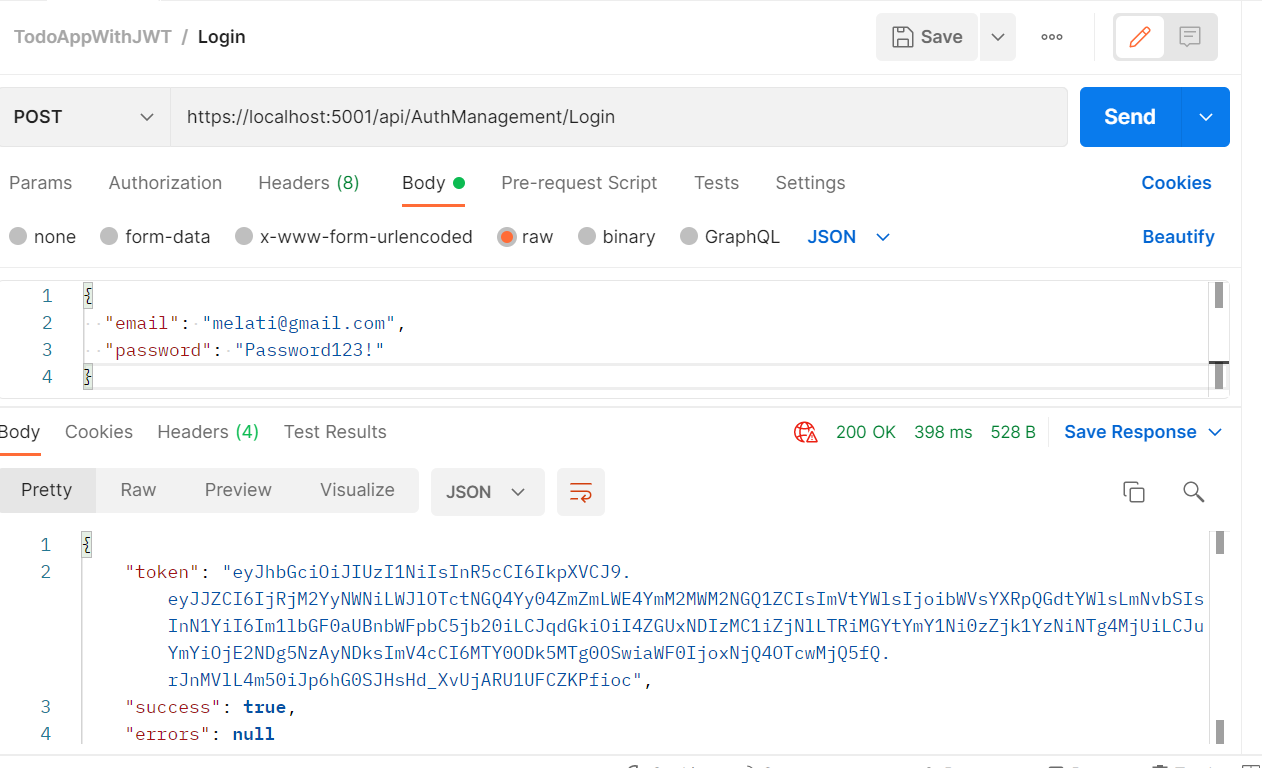
TodoController tidak dapat digunakan karena memerlukan token,



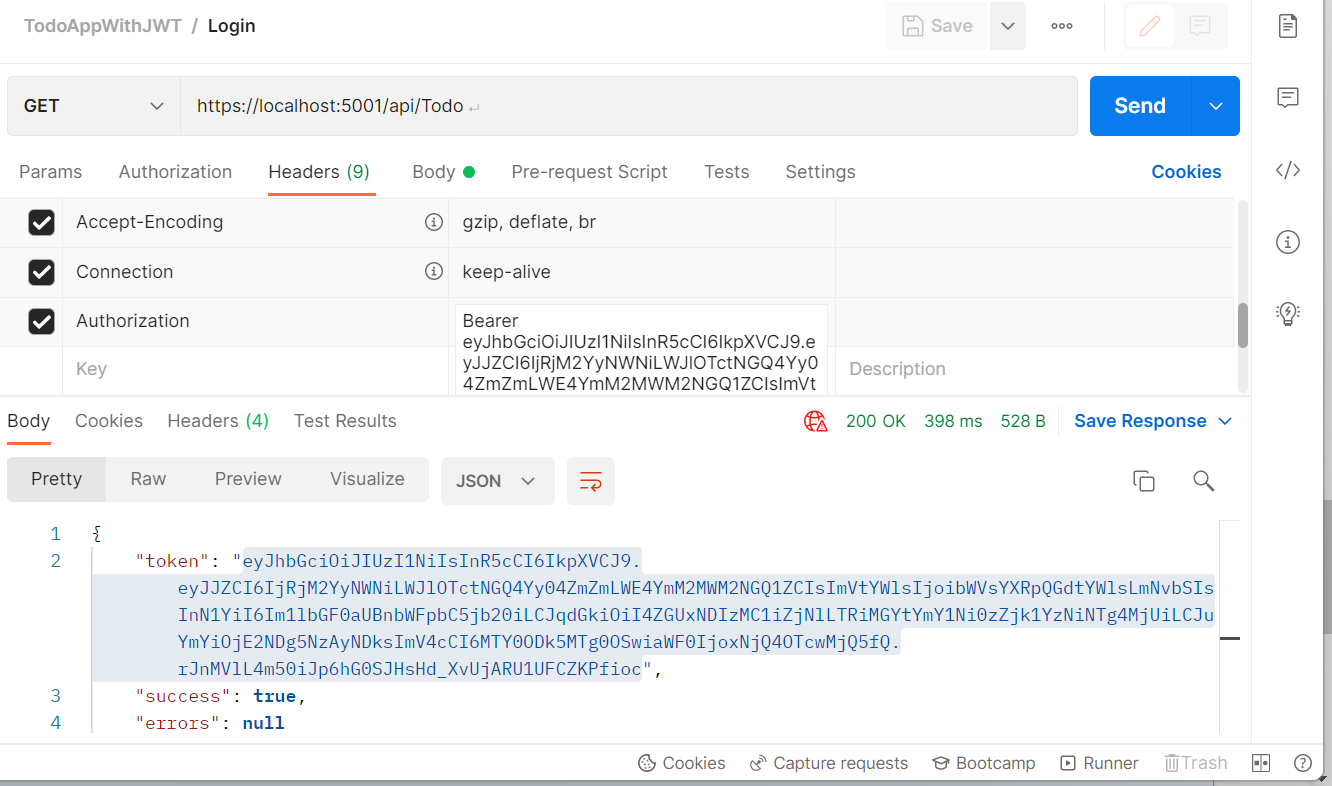
Jika ingin mencoba todocontroller, dapat melalui Postman, masukkan url nya, lalu pilih body,raw dan type json. Dan masukkan email dan password dan klik tombol send



Dann postman akan meresponse success, dan akan mereturn token



Untuk menggunakan TodoController, ganti fungsi post menjadi get, dan masukkan url [**https://localhost:5001/api/Todo**](https://localhost:5001/api/Todo)**,** pada headers tambahkan sebuah value Authorization, dan masukkan bearer tokennya



Lalu postman akan meresponse sukses, dan menampilkan data pada database

