Nama : Alexander Radianta Tarigan

No\_Peserta : FSDO003ONL010

Hal yang pertama yang harus dilakukan adalah membuat TokenValidationParameters pada class Startup.cs, dan buat di dalam method ConfigureServices

var key = Encoding.ASCII.GetBytes(Configuration["JwtConfig:Secret"]);

            var tokenValidationParams = new TokenValidationParameters{

                ValidateIssuerSigningKey =true,

                IssuerSigningKey = new SymmetricSecurityKey(key),

                ValidateIssuer = false,

                ValidateAudience = false,

                ValidateLifetime = true,

                RequireExpirationTime = false,

                ClockSkew = TimeSpan.Zero

            };

            services.AddSingleton(tokenValidationParams);

            services.AddAuthentication(options => {

                options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

                options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;

                options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

            })

            .AddJwtBearer(jwt => {

                jwt.SaveToken = true;

                jwt.TokenValidationParameters = tokenValidationParams;

            });

Selanjutnya, kita butuh update function GenerateJwtToken() di AuthManagementController,

private async Task<AuthResult> 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.AddSeconds(30),

                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)

            };

            var token = jwtTokenHandler.CreateToken(tokenDescriptor);

            var jwtToken = jwtTokenHandler.WriteToken(token);

            return jwtToken;

        }

Lalu buat class baru bernama AuthResult.cs dan simpan di folder configuration, dan tambahkan variable RefreshToken untuk menampung refresh token.

public class AuthResult

    {

        public string Token {get; set;}

        public string RefreshToken {get; set;}

        public bool Success {get; set;}

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

    }

Selanjutnya, tambahkan class baru bernama TokenRequest didalam folder Models/DTOs/Requests. TokenRequest berfungsi untuk mengelola refresh token.

public class TokenRequest

    {

       [Required]

       public string Token {get;set;}

       [Required]

       public string RefreshToken {get; set;}

    }

Selanjutnya buat class baru bernama RefreshToken.cs dan simpan di folder Models.

public class RefreshToken

    {

        public int Id {get; set;}

        public string UserId {get; set;}

        public string Token {get; set;}

        public string JwtId {get; set;}

        public bool IsUsed {get; set;}

        public bool IsRevorked { get; set;}

        public DateTime  AddedDate {get; set;}

        public DateTime ExpiryDate {get; set;}

        [ForeignKey(nameof(UserId))]

        public IdentityUser User {get; set;}

    }

Selanjutnya update ApiDbContext.cs,tambahkan public virtual DbSet<RefreshToken> RefreshTokens {get;set;}.

public class ApiDbContext : IdentityDbContext

    {

        public virtual DbSet<ItemData> Items {get; set;}

        public virtual DbSet<RefreshToken> RefreshTokens {get; set;}

        public ApiDbContext(DbContextOptions<ApiDbContext> options)

            :base(options)

        {

        }

    }

Lakukan migrasi dengan menjalankan perintah dotnet ef migrations add “menambahkan table refresh token”



Lakukan update database dengan menjalankan perintah dotnet ef database update diterminal



Lalu pada Controller AuthManagementController.cs tambahkan

private readonly TokenValidationParameters \_tokenValidationParams;

        private readonly ApiDbContext \_apiDbContext;

        public AuthManagementController(

            UserManager<IdentityUser> userManager,

            IOptionsMonitor<JwtConfig> optionsMonitor,

            TokenValidationParameters tokenValidationParams,

            ApiDbContext apiDbContext)

        {

            \_userManager = userManager;

            \_jwtConfig = optionsMonitor.CurrentValue;

            \_tokenValidationParams = tokenValidationParams;

            \_apiDbContext = apiDbContext;

        }

Dan update fungsi GenerateToken pada file AuthManagementController.cs

var token = jwtTokenHandler.CreateToken(tokenDescriptor);

            var jwtToken = jwtTokenHandler.WriteToken(token);

            var refreshToken = new RefreshToken()

            {

                JwtId = token.Id,

                IsUsed = false,

                IsRevorked = false,

                UserId = user.Id,

                AddedDate = DateTime.UtcNow,

                ExpiryDate = DateTime.UtcNow.AddMonths(6),

                Token = RandomString(35) + Guid.NewGuid()

            };

            //return jwtToken;

            await \_apiDbContext.RefreshTokens.AddAsync(refreshToken);

            await \_apiDbContext.SaveChangesAsync();

            return new AuthResult(){

                Token = jwtToken,

                Success = true,

                RefreshToken = refreshToken.Token

            };

Tambahkan method RandomString

private string RandomString(int length)

        {

            var random = new Random();

            var chars = "ABCDEFGHIJKLMNOPRQSTUVWXYZ01234567890";

            return new string(Enumerable.Repeat(chars, length)

                .Select(x => x[random.Next(x.Length)]).ToArray());

        }

Lalu return GenerateJwtToken di register dan login

Login

 var jwtToken = GenerateJwtToken(existingUser);

                return Ok(jwtToken);

if(isCreated.Succeeded)

                {

                    var jwtToken = GenerateJwtToken( newUser);

                    return Ok(jwtToken);

                    // return Ok(new RegistrationResponse() {

                    //     Success = true,

                    //     Token = jwtToken

                    // });

                }

Lalu tambahkan action RefreshToken di AuthManagementController.cs

[HttpPost]

        [Route("RefreshToken")]

        public async Task<IActionResult> RefreshToken([FromBody] TokenRequest tokenRequest)

        {

            if(ModelState.IsValid)

            {

                var result = await VerifyAndGenerateToken(tokenRequest);

                if(result == null)

                {

                    return BadRequest(new RegistrationResponse(){

                        Errors = new List<string>(){

                            "Invalid tokens"

                        },

                        Success = false

                    });

                }

                return Ok(result);

            }

            return BadRequest(new RegistrationResponse(){

                Errors = new List<string>(){

                    "Invalid payload"

                },

                Success = false

            });

        }

Tambahkan method VerifyAndGenerateToken

private async Task<AuthResult> VerifyAndGenerateToken(TokenRequest tokenRequest)

        {

            var jwtTokenHandler = new JwtSecurityTokenHandler();

            try

            {

                //validation 1 - validation jwt token format

                var tokenInVerification = jwtTokenHandler.ValidateToken(tokenRequest.Token, \_tokenValidationParams, out var validatedToken);

                //validation 2 - Encryption Alg

                if(validatedToken is JwtSecurityToken jwtSecurityToken)

                {

                    var result = jwtSecurityToken.Header.Alg.Equals(SecurityAlgorithms.HmacSha256, StringComparison.InvariantCultureIgnoreCase);

                    if(result == false)

                    {

                        return null;

                    }

                }

                //validation 3 - expiry date

                var utcExpiryDate = long.Parse(tokenInVerification.Claims.FirstOrDefault(x => x.Type == JwtRegisteredClaimNames.Exp).Value);

                var expiryDate =  UnixTimeStampToDateTime(utcExpiryDate);

                if(expiryDate > DateTime.UtcNow){

                    return new AuthResult(){

                        Success = false,

                        Errors = new List<string>(){

                            "Token has not yet expired"

                        }

                    };

                }

                //validation 4 - existence of the Token

                var storedToken = await \_apiDbContext.RefreshTokens.FirstOrDefaultAsync(x => x.Token == tokenRequest.RefreshToken);

                if(storedToken == null)

                {

                    return new AuthResult(){

                        Success = false,

                        Errors = new List<string>(){

                            "Token does not exist"

                        }

                    };

                }

                //validation 5 - validate if used

                if(storedToken.IsUsed)

                {

                    return new AuthResult(){

                        Success = false,

                        Errors = new List<string>(){

                            "Token has been used"

                        }

                    };

                }

                //validation 6 - if revorked

                if(storedToken.IsRevorked)

                {

                    return new AuthResult(){

                        Success = false,

                        Errors = new List<string>(){

                            "Token has been revorked"

                        }

                    };

                }

                //validation 7 - validate the id

                var jti = tokenInVerification.Claims.FirstOrDefault(x => x.Type == JwtRegisteredClaimNames.Jti).Value;

                if(storedToken.JwtId != jti)

                {

                    return new AuthResult(){

                        Success=false,

                        Errors = new List<string>(){

                            "Token doesn't match"

                        }

                    };

                }

                //update current token

                storedToken.IsUsed = true;

                \_apiDbContext.RefreshTokens.Update(storedToken);

                await  \_apiDbContext.SaveChangesAsync();

                //Generate new token

                var dbUser = await \_userManager.FindByIdAsync(storedToken.UserId);

                return await GenerateJwtToken(dbUser);

            }

            catch(Exception ex)

            {

                if(ex.Message.Contains("Lifetime validation failed. The token is expired.")){

                    return new AuthResult(){

                        Success = false,

                        Errors = new List<string>(){

                            "Token has expired please re-login"

                        }

                    };

                }else{

                    return new AuthResult(){

                        Success = false,

                        Errors = new List<string>(){

                            "Something went wrong."

                        }

                    };

                }

            }

        }

        private DateTime UnixTimeStampToDateTime(long unixTimeStamp)

        {

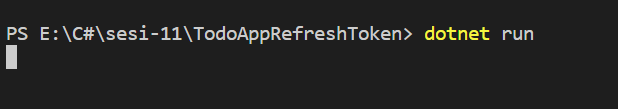
            var dateTimeVal = new DateTime(1970, 1,1,0,0,0,0, DateTimeKind.Utc);

            dateTimeVal = dateTimeVal.AddSeconds(unixTimeStamp).ToUniversalTime();

            return dateTimeVal;

        }

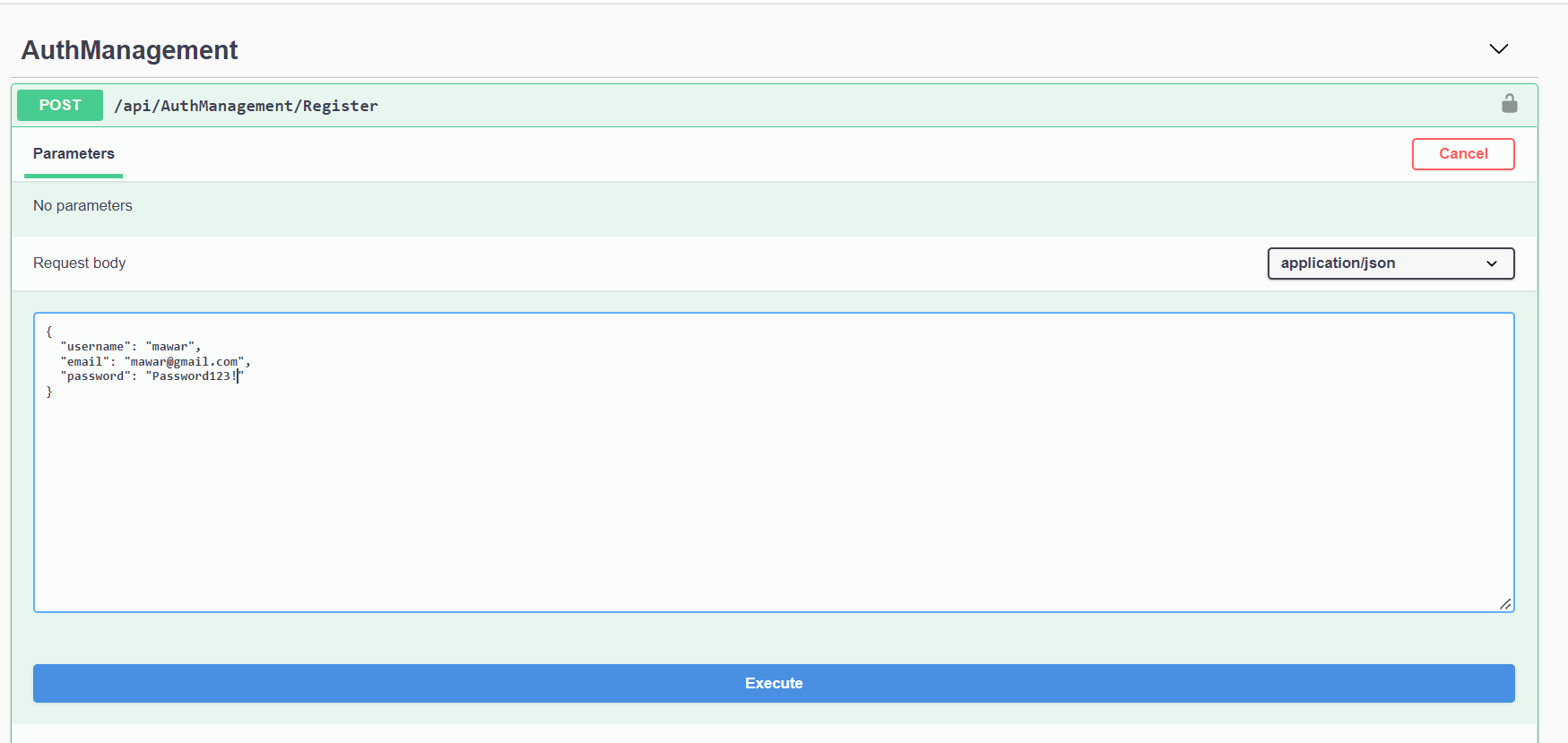
Jalankan melalui swagger



Buka Chrome dan akses

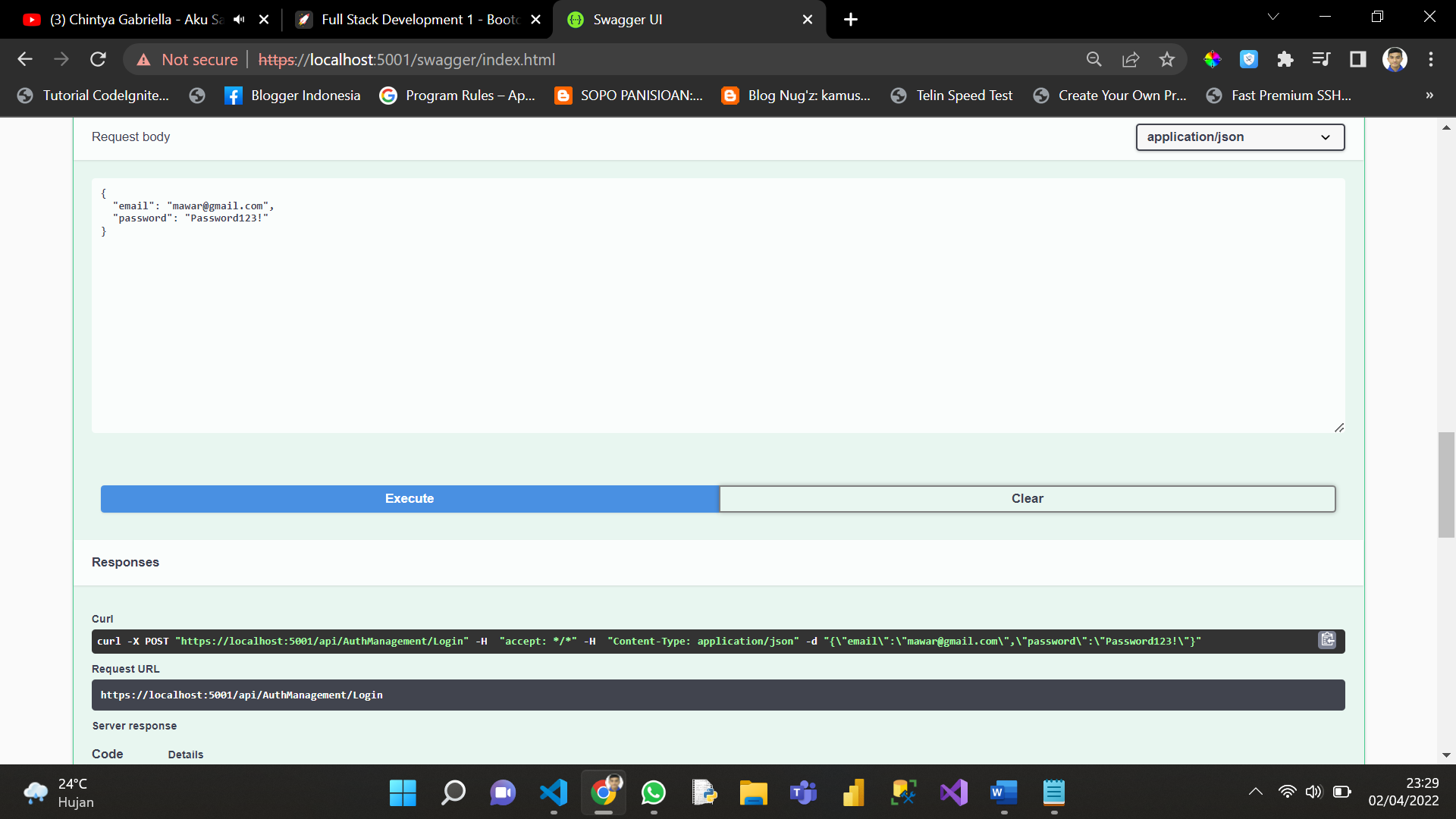
<https://localhost:5001/swagger/index.html>

Lalu coba daftarkan sebuah akun melalui controller AuthManagement

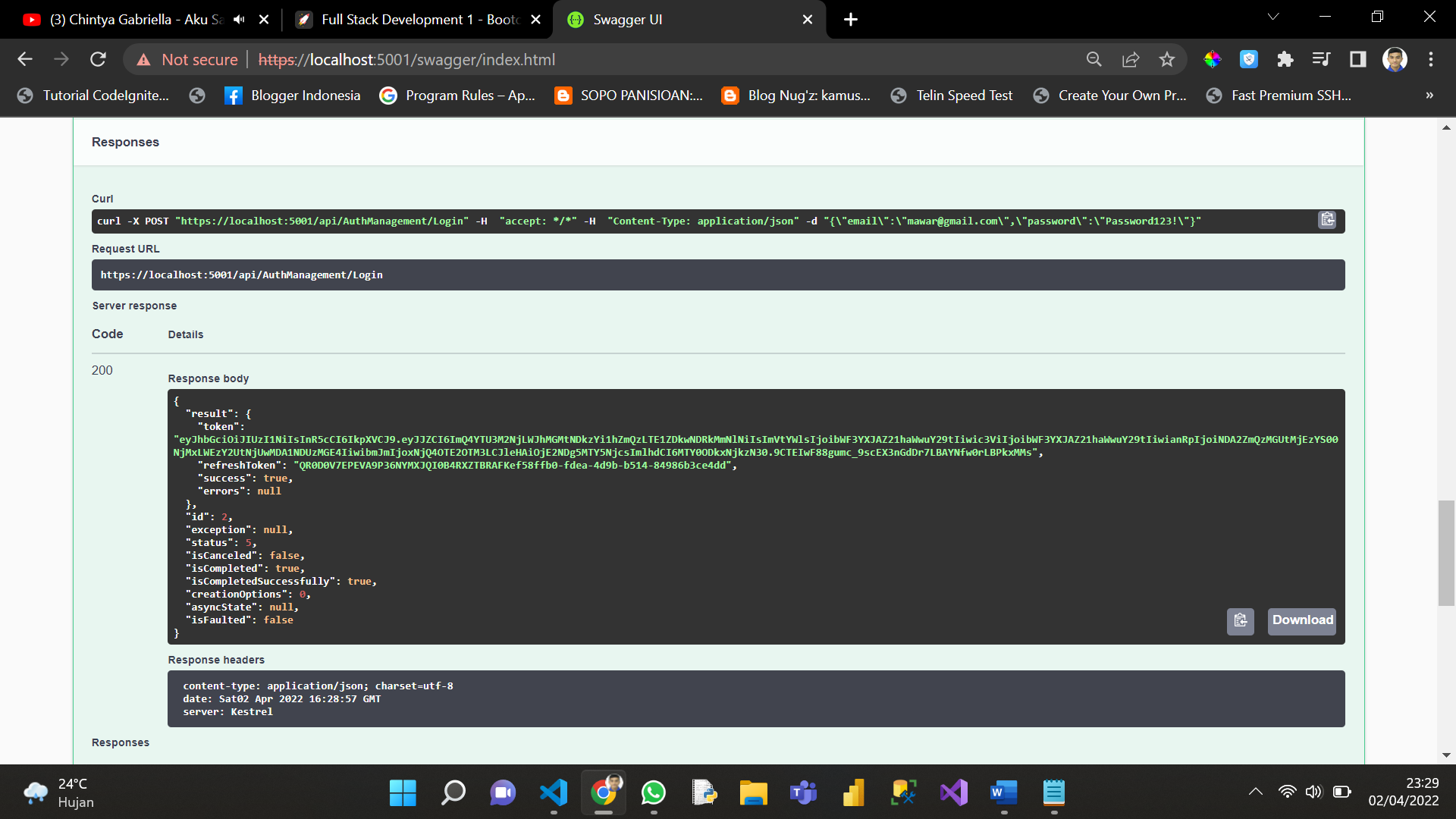


Maka swagger akan memberikan response dan mereturn token dan refreshtoken

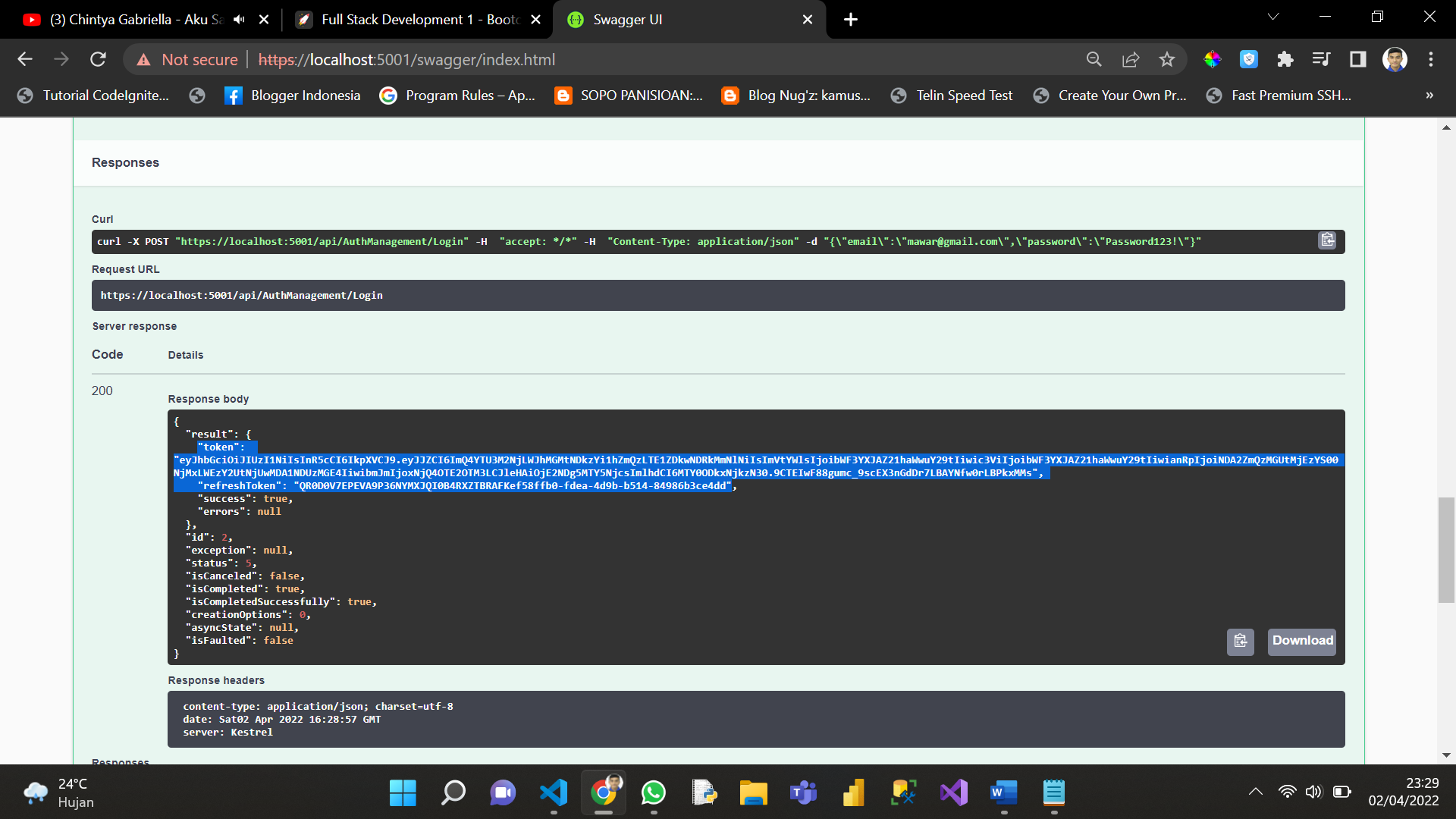
Coba login menggunakan akun yang baru didaftarkan tadi



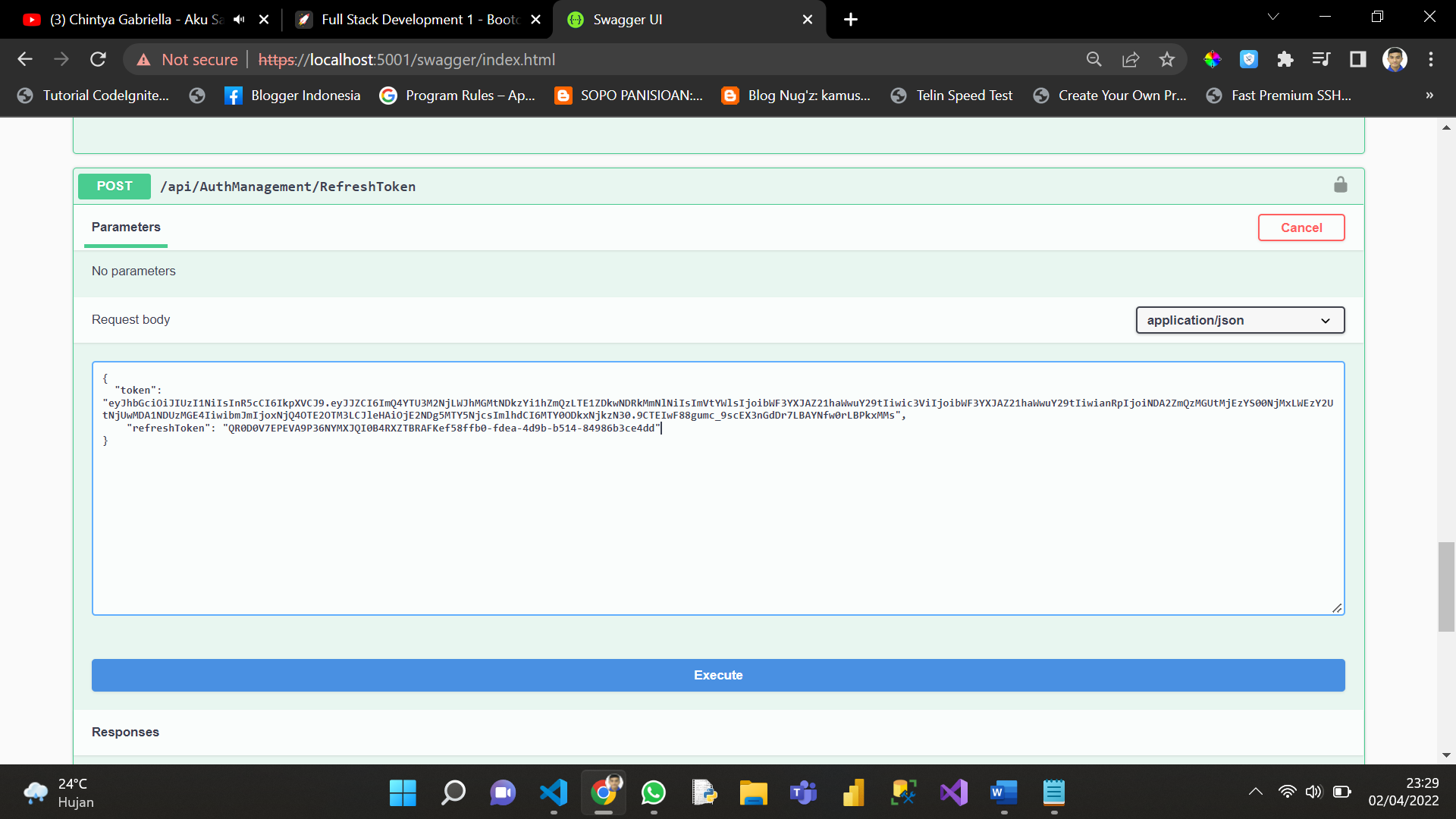
Lalu swagger akan memberikan response sukses dan akan mereturn token dan refreshtoken



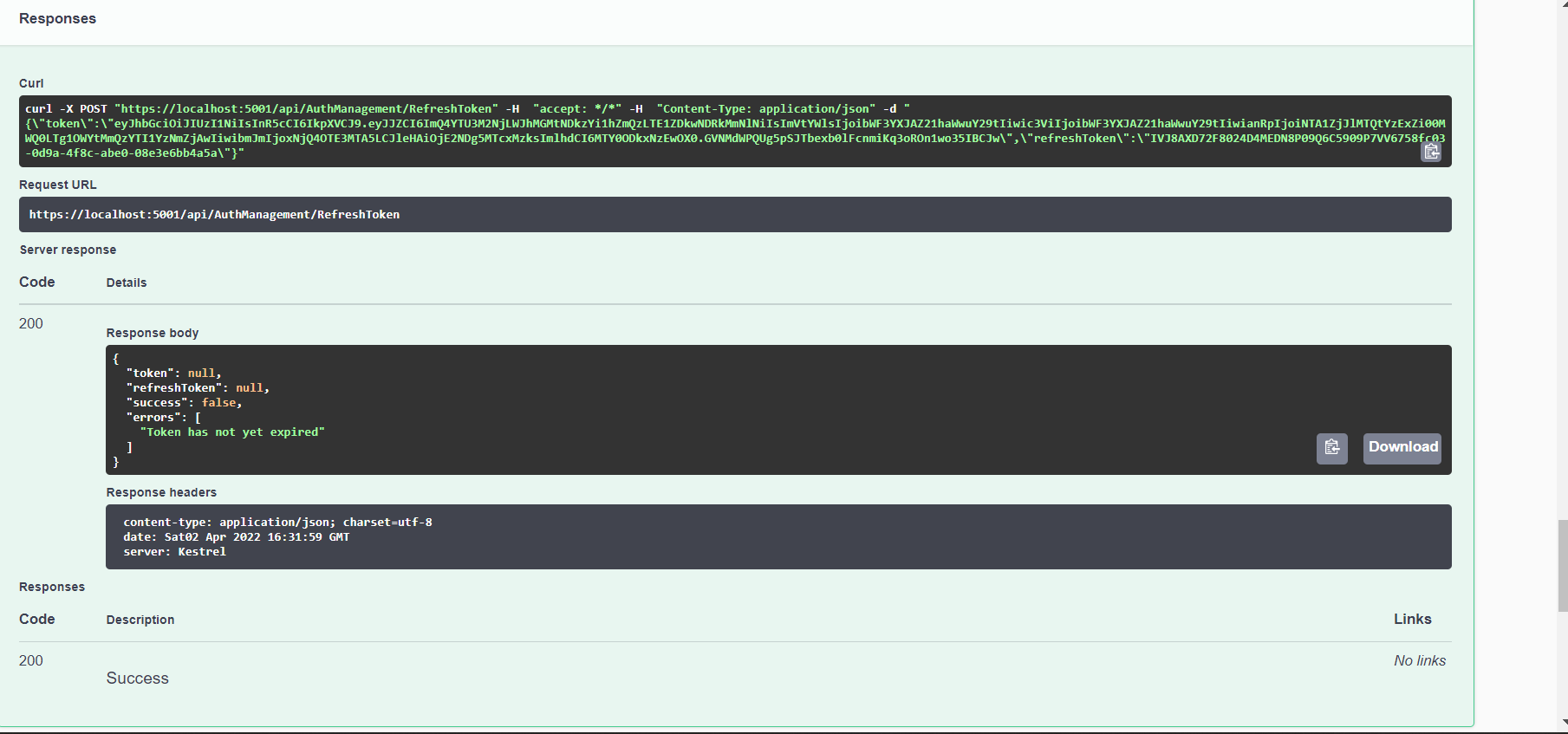
Untuk mengecek tokennya masih berlaku atau tidak, salin terlebih dahul token dan refreshtoken.



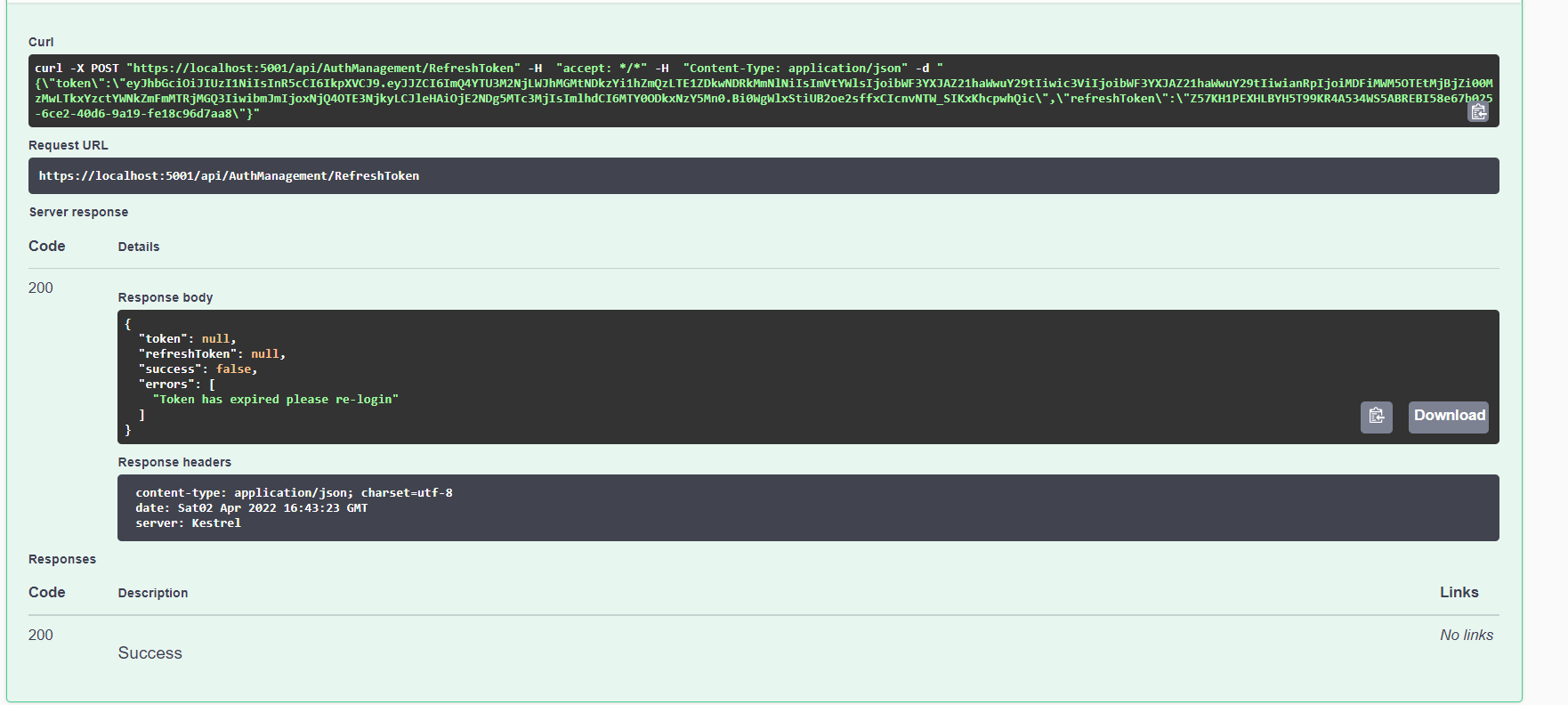
Lalu Klik RefreshToken, dan masukkan token dan refresh token



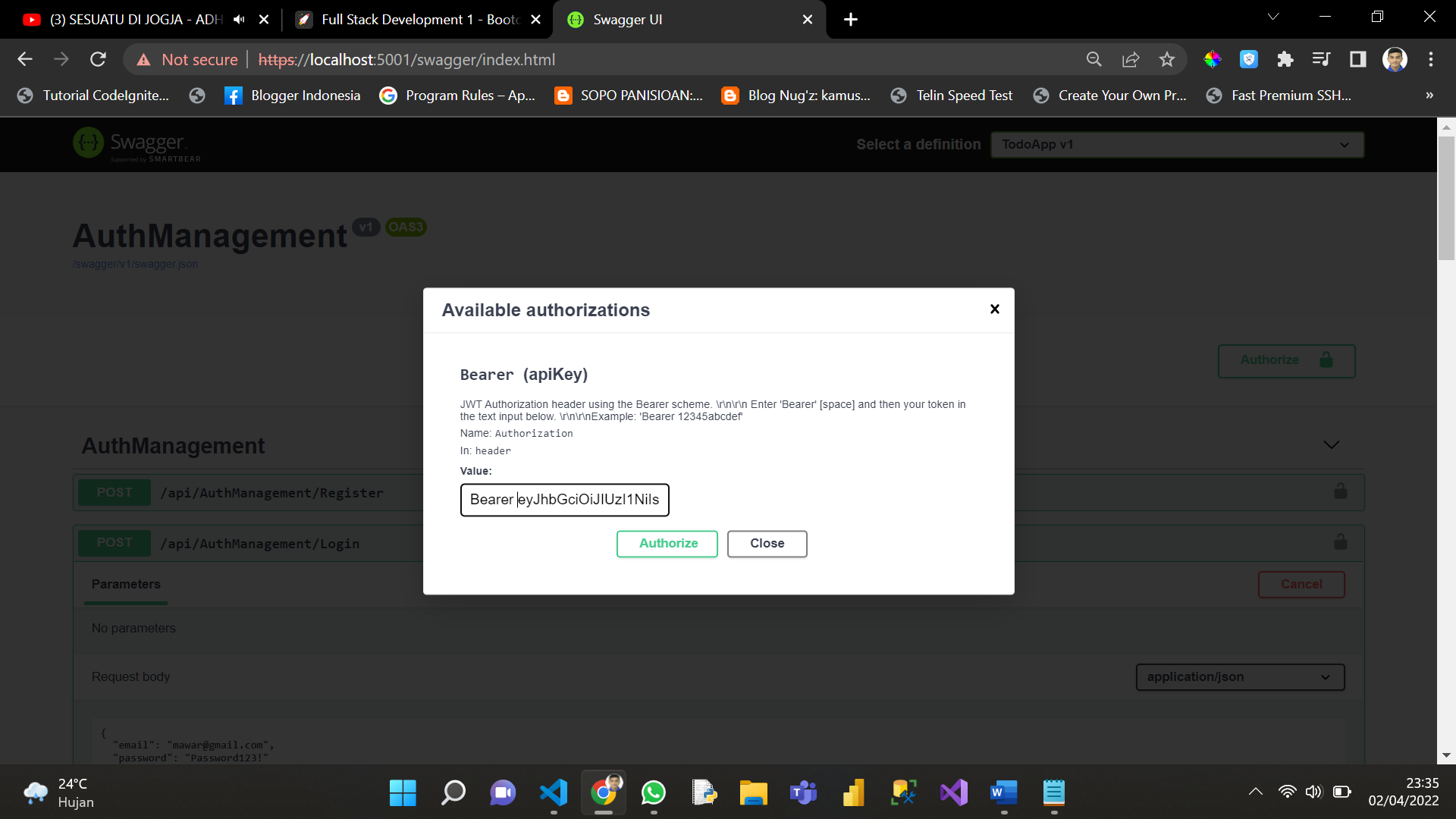
Jika belum expired, swagger akan meresponse sukses dan mereturn “token has not yet expired”



Jika sudah expired, swagger akan mereturn



Jika ingin menggunakan todocontroller, salin terlebih dahulu token yang telah kita dapatkan saat berhasil login, klik tombol authorize, lalu masukkan Bearer {token} dan klik authorize



Maka todocontroller dapat digunakan.

