JWT AUTHENTICATION

WITH

MINIMAL APIS

@romain-od

www.code-review.tech

INIT PROJET

- Create the minmal api project
- Add package JwtBearer
- Open VS code

```
dotnet new web -o MinimalApi.JWT

cd MinimalApi.JWT

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer

code .
```

@romain-od

www.code-review.tech

CREATE ENDPOINT

Protect your routes using authorization policies and forces you to provide authentication information when calling this endpoint by use of the RequireAuthorization extension method

```
var builder = WebApplication.CreateBuilder(args);

var app = builder.Build();

app.MapGet("/security/getMessage",

() => "Hello World!").RequireAuthorization();

app.Run();
```

@romain-od

www.code-review.tech

SPECIFY A SECRET KEY IN THE APPSETTINGS.JSON FILE

```
"Jwt": {
  "Issuer": "https://code-review.tech/",
  "Audience": "https://code-review.tech/",
  "Key": "This is a sample secret key"
}
```

@romain-od
www.code-review.tech

CONFIGURE JWT AUTHENTICATION

```
builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(o =>
{
    o.TokenValidationParameters = new TokenValidationParameters
    {
        ValidIssuer = builder.Configuration["Jwt:Issuer"],
        ValidAudience = builder.Configuration["Jwt:Audience"],
        IssuerSigningKey = new SymmetricSecurityKey
        (Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"])),
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = false,
        ValidateIssuerSigningKey = true
    };
});
builder.Services.AddAuthorization();

var app = builder.Build();
app.UseAuthentication();
app.UseAuthorization();
```

@romain-od

www.code-review.tech

CREATE A USER MODEL

Needed to store the login credentials of the user

```
public class User
{
    public string UserName { get; set; }
    public string Password { get; set; }
}
```

@romain-od

www.code-review.tech

CREATE AN ENDPOINT TO GENERATE JWT 1/2

```
app.MapPost("/security/createToken",
[AllowAnonymous] (User user) =>
{
    if (user.UserName == "romain_od" && user.Password == "codereview1234")
    {
        var issuer = builder.Configuration["Jwt:Issuer"];
        var audience = builder.Configuration["Jwt:Audience"];
        var key = Encoding.ASCII.GetBytes
        (builder.Configuration["Jwt:Key"]);
        var tokenDescriptor = new SecurityTokenDescriptor
        {
            Subject = new ClaimsIdentity(new[]
            {
                new Claim("Id", Guid.NewGuid().ToString()),
                new Claim(JwtRegisteredClaimNames.Sub, user.UserName),
                new Claim(JwtRegisteredClaimNames.Email, user.UserName),
                new Claim(JwtRegisteredClaimNames.Jti,
                Guid.NewGuid().ToString())
            }),
```

@romain-od

www.code-review.tech

CREATE AN ENDPOINT TO GENERATE JWT 2/2

```
            Expires = DateTime.UtcNow.AddMinutes(5),
            Issuer = issuer,
            Audience = audience,
            SigningCredentials = new SigningCredentials
            (new SymmetricSecurityKey(key),
            SecurityAlgorithms.HmacSha512Signature)
        };
        var tokenHandler = new JwtSecurityTokenHandler();
        var token = tokenHandler.CreateToken(tokenDescriptor);
        var jwtToken = tokenHandler.WriteToken(token);
        var stringToken = tokenHandler.WriteToken(token);
        return Results.Ok(stringToken);
    }
    return Results.Unauthorized();
});
```

@romain-od
www.code-review.tech

TEST IT WITH POSTMAN

Post the user credentials to the createToken endpoint

Retreive the generated token

Call the HTTP Get endpoint getMessage and pass the generated token as a bearer token in the request header

@romain-od

www.code-review.tech

THANKS FOR READING

SHARE YOUR THOUGHTS

FOLLOW ME FOR MORE SHARING

@romain-od

www.code-review.tech