Controllers:

AuthController.cs:

using JWT\_Microservice.Models;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace JWT\_Microservice.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

private readonly IConfiguration \_config;

public AuthController(IConfiguration config)

{

\_config = config;

}

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

if (model.Username == "admin" && model.Password == "password")

{

var token = GenerateJwtToken(model.Username);

return Ok(new { Token = token });

}

return Unauthorized();

}

private string GenerateJwtToken(string username)

{

var securityKey = new SymmetricSecurityKey(

Encoding.UTF8.GetBytes(\_config["Jwt:Key"]));

var credentials = new SigningCredentials(

securityKey, SecurityAlgorithms.HmacSha256);

var claims = new[]

{

new Claim(ClaimTypes.Name, username),

new Claim(ClaimTypes.Role, "User")

};

var token = new JwtSecurityToken(

issuer: \_config["Jwt:Issuer"],

audience: \_config["Jwt:Audience"],

claims: claims,

expires: DateTime.Now.AddMinutes(

Convert.ToDouble(\_config["Jwt:DurationInMinutes"])),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

SecureController.cs:

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace JWT\_Microservice.Controllers

{

[ApiController]

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

[Authorize]

public class SecureController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

return Ok(new { Message = "This is a secured endpoint!" });

}

[HttpGet("admin")]

[Authorize(Roles = "Admin")]

public IActionResult AdminEndpoint()

{

return Ok(new { Message = "This is an admin-only endpoint!" });

}

}

}

Models:

LoginModel.cs:

namespace JWT\_Microservice.Models

{

public class LoginModel

{

public string Username { get; set; }

public string Password { get; set; }

}

}

User.cs:

namespace JWT\_Microservice.Models

{

public class User

{

public string Username { get; set; }

public string Password { get; set; }

}

}

Properties:

Program.cs:

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

var jwtSettings = builder.Configuration.GetSection("Jwt");

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "JWT Auth API", Version = "v1" });

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Description = "JWT Authorization header using the Bearer scheme. Example: \"Authorization: Bearer {token}\"",

Name = "Authorization",

In = ParameterLocation.Header,

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

builder.Services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

options.SaveToken = true;

options.RequireHttpsMetadata = false;

options.TokenValidationParameters = new TokenValidationParameters()

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = jwtSettings["Issuer"],

ValidAudience = jwtSettings["Audience"],

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings["Key"])),

ClockSkew = TimeSpan.Zero

};

});

builder.Services.AddAuthorization();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "JWT Auth API v1");

c.RoutePrefix = string.Empty;

});

}

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

Appsettings.json:

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

}

}

Appsettings.json:

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"Jwt": {

"Key": "ThisIsASecretKeyForJwtToken12345",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60

}

}

JWT\_Micro.sln:

using System.Diagnostics;

using static System.Runtime.InteropServices.JavaScript.JSType;

Microsoft Visual Studio Solution File, Format Version 12.00

# Visual Studio Version 17

VisualStudioVersion = 17.14.36221.1 d17.14

MinimumVisualStudioVersion = 10.0.40219.1

Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "JWT\_Microservice", "JWT\_Microservice.csproj", "{041A4C4F-5B0D-4600-9578-D6D13B800AF7}"

EndProject

Global

GlobalSection(SolutionConfigurationPlatforms) = preSolution

Debug|Any CPU = Debug|Any CPU

Release|Any CPU = Release|Any CPU

EndGlobalSection

GlobalSection(ProjectConfigurationPlatforms) = postSolution

{041A4C4F-5B0D-4600-9578-D6D13B800AF7}.Debug | Any CPU.ActiveCfg = Debug | Any CPU

{041A4C4F-5B0D-4600-9578-D6D13B800AF7}.Debug | Any CPU.Build.0 = Debug | Any CPU

{041A4C4F-5B0D-4600-9578-D6D13B800AF7}.Release | Any CPU.ActiveCfg = Release | Any CPU

{041A4C4F-5B0D-4600-9578-D6D13B800AF7}.Release | Any CPU.Build.0 = Release | Any CPU

EndGlobalSection

GlobalSection(SolutionProperties) = preSolution

HideSolutionNode = FALSE

EndGlobalSection

GlobalSection(ExtensibilityGlobals) = postSolution

SolutionGuid = {90BDC3FD-4F33-4EB7-AFC8-D3A8EEF2EC00}

EndGlobalSection

EndGlobal

JWT\_Micro.http:

using Microsoft.AspNetCore.Http.HttpResults;

using WebApplication4;

using static System.Net.Mime.MediaTypeNames;

@JWT\_Microservice\_HostAddress = http://localhost:5096

GET {{JWT\_Microservice\_HostAddress}}/ weatherforecast /

Accept: application / json

###

JWT\_Micro.csproj:

< Project Sdk = "Microsoft.NET.Sdk.Web" >

< PropertyGroup >

< TargetFramework > net8.0 </ TargetFramework >

< Nullable > enable </ Nullable >

< ImplicitUsings > enable </ ImplicitUsings >

</ PropertyGroup >

< ItemGroup >

< PackageReference Include = "Microsoft.AspNetCore.Authentication.JwtBearer" Version = "8.0.8" />

< PackageReference Include = "Microsoft.AspNetCore.OpenApi" Version = "8.0.17" />

< PackageReference Include = "Swashbuckle.AspNetCore" Version = "9.0.3" />

< PackageReference Include = "System.IdentityModel.Tokens.Jwt" Version = "8.12.1" />

</ ItemGroup >

</ Project >

launchsettings.json:

{

"$schema": "http://json.schemastore.org/launchsettings.json",

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:21378",

"sslPort": 44327

}

},

"profiles": {

"http": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"launchUrl": "swagger",

"applicationUrl": "http://localhost:5096",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"https": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"launchUrl": "swagger",

"applicationUrl": "https://localhost:7157;http://localhost:5096",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"launchUrl": "swagger",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}

OUTPUT:





