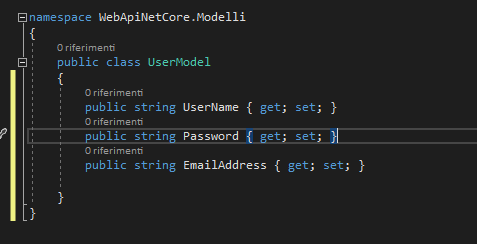
**jwt authentication con netcore.**

Video dimostrativo step by step:

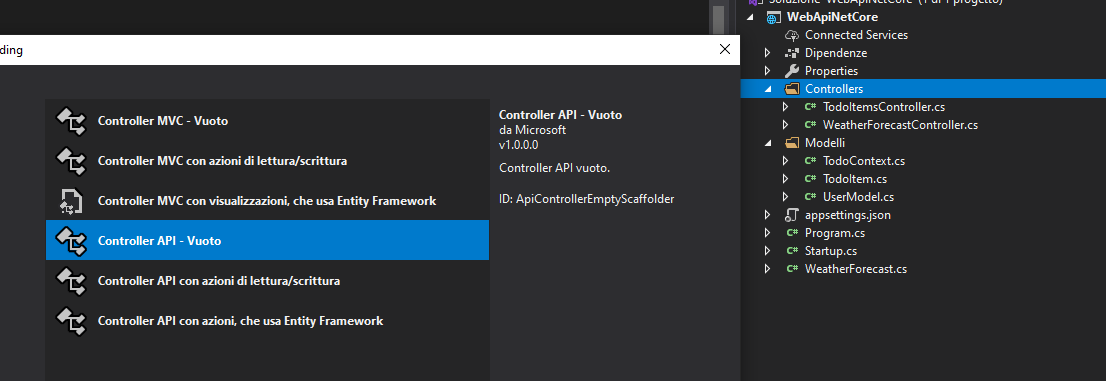
<https://www.youtube.com/watch?v=l56YLbAVAfo>

Step per implementare JWT bearer

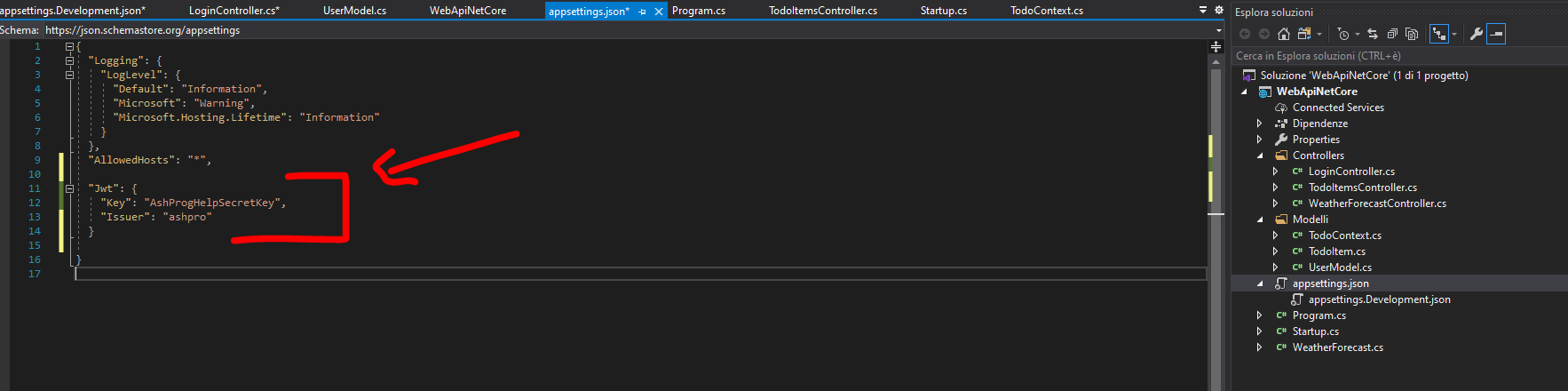
Creo un model (il model servirà per raccogliere i dati relativi all’utenza dal db quando viene recuperata dal db per poi memorizzarli nei claims:



Creo controller api vuoto di Login:



Vado aggiungere la secret key nel file di configurazione



Questo è il codice del controllo di login:

using System.Linq;

using System.Security.Claims;

using System.Text;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.IdentityModel.Tokens;

using WebApiNetCore.Modelli;

namespace WebApiNetCore.Controllers

{

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

[ApiController]

public class LoginController : ControllerBase

{

private IConfiguration \_config;

public LoginController(IConfiguration config)

{

\_config = config;

}

//prende le credenziali di login e se le credenziali sono valide ossia userid e pwd trovano riscontro nel database

//con i dati dell'utente viene generato un token

public IActionResult Login(string username, string pass)

{

UserModel login = new UserModel();

login.UserName = username;

login.Password = pass;

IActionResult reponse = Unauthorized();

var user = AuthenticateUser(login);

if (user != null)

{

//se le credenziali di accesso sono corrette genero il token

var tokenStr = GenerateJSONWebToken(user);

reponse = Ok(new {token= tokenStr });

}

return reponse;

}

private UserModel AuthenticateUser(UserModel login)

{

UserModel user = null;

//qui va a vedere nel db se esiste un username con userid e password

if (login.UserName == "ashproghelp" && login.Password == "123")

{

user = new UserModel { UserName = "ashproghelp", EmailAddress = "ashproghelp@gmail.com", Password = "123" };

}

return user;

}

private string GenerateJSONWebToken(UserModel userinfo)

{

//installare qui pacchetto nuget microsoft.identity.model.tokens

//la chiave dell'appsetting viene codificata in binario

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config["Jwt:Key"]));

//la chiave dell'appsetting viene passata a SigningCredentials per ottenere delle credenziali

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

//crea i JWT claims

var claims = new[]

{

//qui installare pacchetto nuget system.identitymodel.token.jwt

new Claim(JwtRegisteredClaimNames.Sub,userinfo.UserName),

new Claim(JwtRegisteredClaimNames.Email,userinfo.EmailAddress),

new Claim(JwtRegisteredClaimNames.Jti,Guid.NewGuid().ToString())

};

//e successivamente si crea il jwt Token

var token = new JwtSecurityToken(

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

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

claims,

expires: DateTime.Now.AddMinutes(120),

signingCredentials: credentials

);

//il token viene passato all'handler di gestione....

var encodetoken = new JwtSecurityTokenHandler().WriteToken(token);

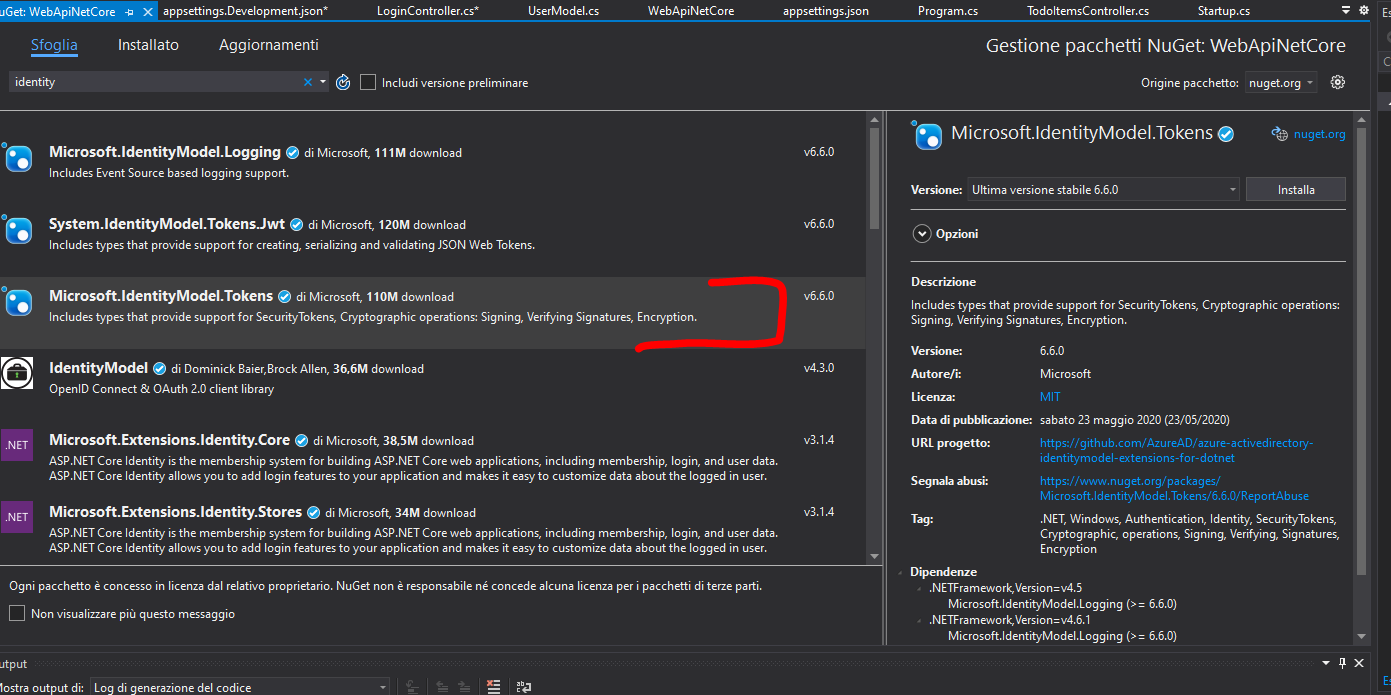
return encodetoken;

}

}

}

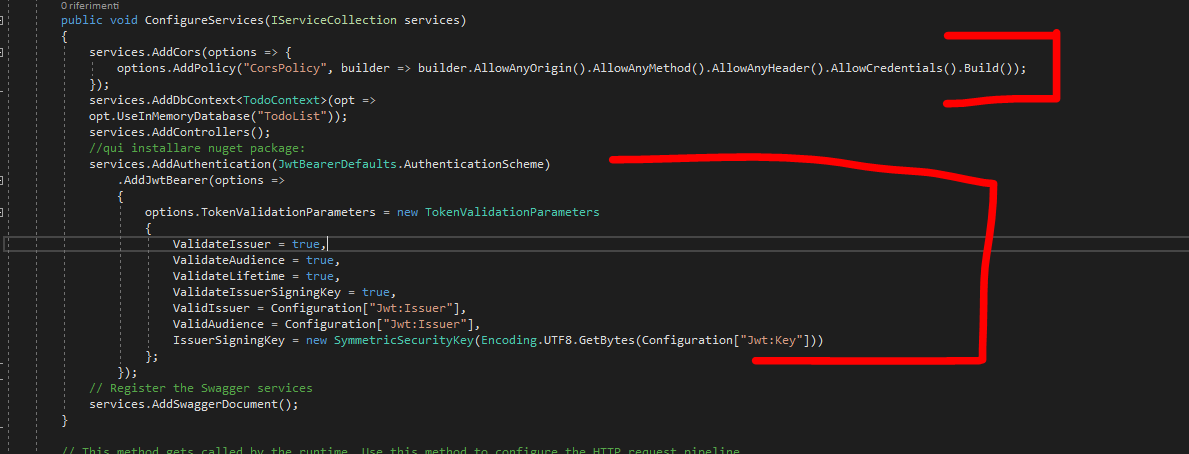
**Installare mycrosoft.Identitymodel.Tokens** e **system.identitymodel.token.jwt** con nuget



Poi occorre installare anche

**nuget system.identitymodel.token.jwt**

quindi andare a registrate nei servizi:



Quindi andare a registrare il servizio aggiungendo cors per permettere di chiamare le api da domini differenti esplicitando determinate policies, e successivamente andando ad aggiungere la autenticazione JWT esplicitando i parametri di validazione:

public void ConfigureServices(IServiceCollection services)

{

services.AddCors(options => {

options.AddPolicy("CorsPolicy", builder => builder.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader().AllowCredentials().Build());

});

services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = Configuration["Jwt:Issuer"],

ValidAudience = Configuration["Jwt:Issuer"],

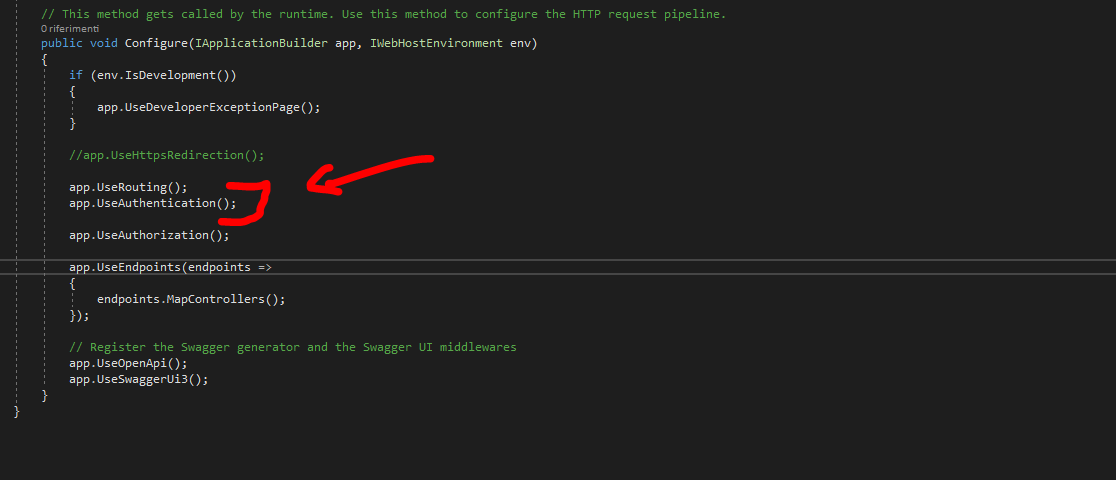
IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["Jwt:Key"]))

};

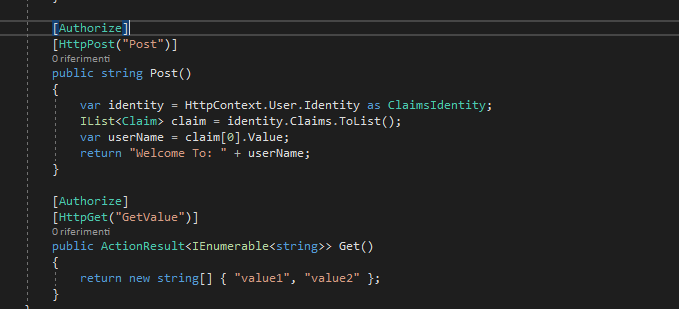
});

}



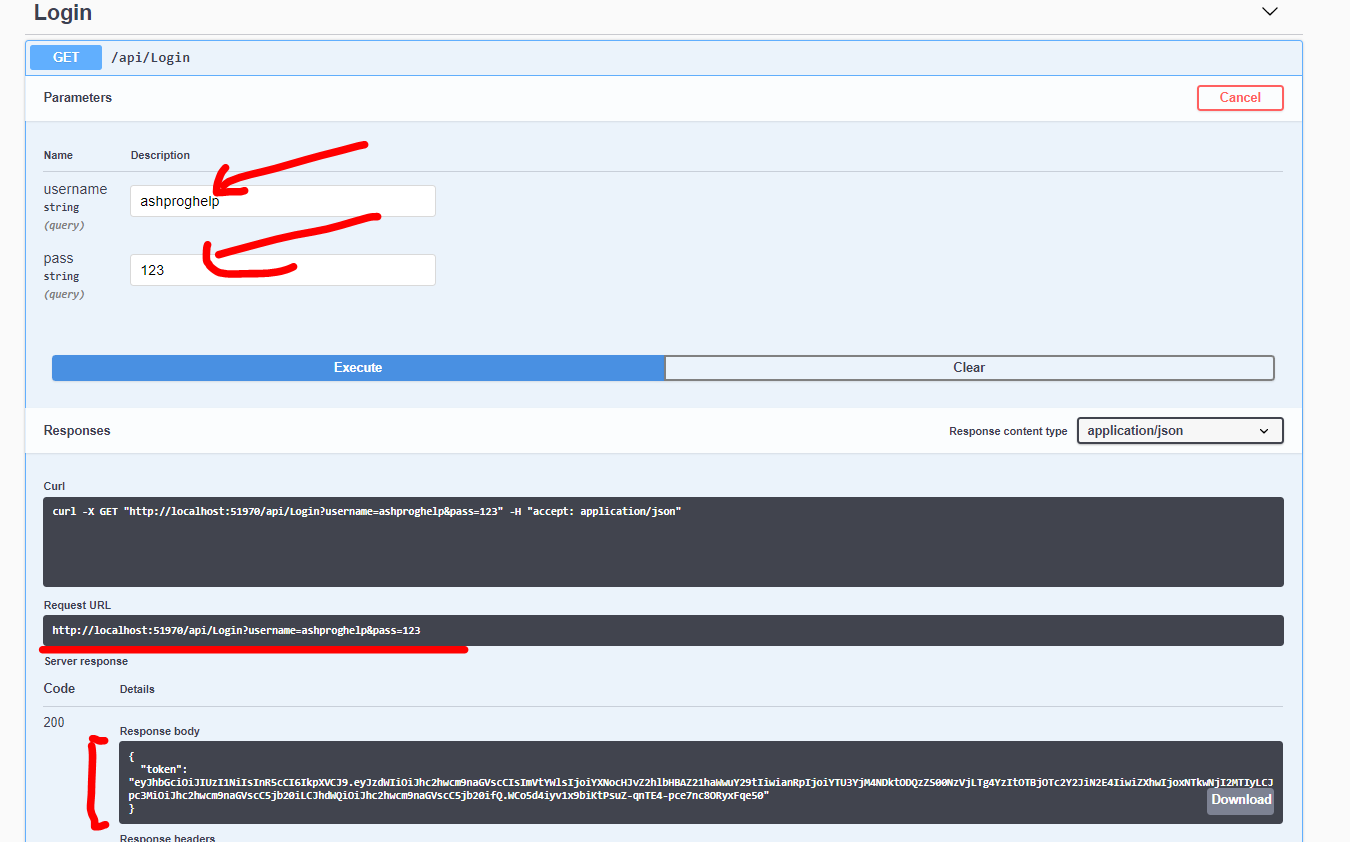


In fondo a login controller aggiungiamo anche due api che ci permettono di testare il tutto e di recuperare utenza claims e dei dati di esempio:



Per testare:

lanciare login, questa genererà un token



Il token va copiato nell’api coperta da security ad esempio post (ricordarsi di passare il token impostando il type bearertoken):

