Autentifikacija i autorizacija

Predrag Glavaš- pedja.glavas@uns.ac.rs
Tanja Čubrić- radojcic.tanja@uns.ac.rs

Autentifikacija

Potrebni NuGet paketi

- BCrypt.Net-Next, najnovija verzija
- Microsoft.AspNetCore.Authentication.JwtBearer, 5.xx verzija

Autentifikacija

- Kako bismo uveli mehanizme kontrole pristupa aplikaciji moramo uvesti I mehanizam korisničke autentifikacije.
- Podatke o korisnicima možemo čuvati u bazi podataka, a po unosu validnih kredencijala korisniku izdajemo Json web Token na osnovu kog se dalje autentifikuje pri pistupu našim servisima.
- U token stavljamo sve podatke koji su nam bitni da znamo o korisniku, a koji nisu poverljivi jer token sam po sebi nije zaštićen!
- Izdati token se mora potpisati digitalnim potpisom sa našim privatnim ključem ili lozinkom kako bi se sprečio napadač da podmetne neki svoj veštački token.

Verifikacija passworda

- Kako bismo se uverili da je korisnik uneo adekvatan password moramo uporediti heš unetog pasvorda i heš vrednost pasvorda iz baze podataka za tog korisnika.
- Biblioteka Bcrypt radi taj posao
- *BCrypt.Net.BCrypt.Verify(unetiPassword, hesVrednostPasswordaUBazi)*

Izdavanje tokena

- Kada smo se uverili u identitet korisnika možemo mu izdati token.
- U token možemo staviti Claimove (prava) koja korisnik ima.

```
List<Claim> claims = new List<Claim>();
//Mozemo dodati Claimove u token, oni ce biti vidljivi u tokenu i mozemo ih koristiti za autorizaciju
if(dto.Username == "pedja")
    claims.Add(new Claim(ClaimTypes.Role, "saradnik")); //Add user type to claim
if (dto.Username == "tanja")
    claims.Add(new Claim(ClaimTypes.Role, "asistent")); //Add user type to claim
if (dto.Username == "pera")
    claims.Add(new Claim(ClaimTypes.Role, "profesor")); //Add user type to claim
//mozemo izmisliti i mi neki nas claim
claims.Add(new Claim("Neki_moj_claim", "imam_ga"));
```

- U token možemo staviti i podatke o izdavaocu i primaocu tokena kao i njegovo trajanje

```
//Kreiramo kredencijale za potpisivanje tokena. Token mora biti potpisan privatnim kljucem
//kako bi se sprecile njegove neovlascene izmene
SymmetricSecurityKey secretKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_secretKey.Value));
var signinCredentials = new SigningCredentials(secretKey, SecurityAlgorithms.HmacSha256);
var tokeOptions = new JwtSecurityToken(
    issuer: "http://localhost:44398", //url servera koji je izdao token
    claims: claims, //claimovi
    expires: DateTime.Now.AddMinutes(20), //vazenje tokena u minutama
    signingCredentials: signinCredentials //kredencijali za potpis
);
string tokenString = new JwtSecurityTokenHandler().WriteToken(tokeOptions);
```

Verifikacija tokena

- Dobijeni token korisnik stavlja u HTTP zahteve prema našem serveru i tako se autentifikuje.
- Naš server međutim, mora verifikovati tokene iz svakog zahteva kako bi se uverili da se neko nije lažno predstavio ili podmetnuo token.
- Ova podešavanja vršimo u ConfigureServices metodi Startup.cs klase

```
//Dodajemo semu autentifikacije i podesavamo da se radi o JWT beareru
services.AddAuthentication(opt => {
    opt.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    opt.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters //Podesavamo parametre za validac
    {
        ValidateIssuer = true, //Validira izdavaoca tokena
        ValidateAudience = false, //Kazemo da ne validira primaoce tokena
        ValidateLifetime = true,//Validira trajanje tokena
        ValidateIssuerSigningKey = true, //validira potpis token, ovo je jako vazno!
        ValidIssuer = "http://localhost:44398", //odredjujemo koji server je validni izdavalac
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["SecretKey"]))
    };
});
```

Verifikacija tokena

- Dodatno, kako bi validacija bila izvršena mora se dodati autentifikacija u Midlver. Jako je bitno na kom je mestu dodajemo, ne može bilo gde!

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        app.UseSwagger();
        app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "Termin_6 v1"));
    }

    app.UseHttpsRedirection();
    app.UseCors(_cors);

    app.UseRouting();


    app.UseAuthentication();
    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

Autorizacija

Autorizacija

- Ako želimo da podržimo različite uloge (vrste) korisnika i ograničimo im prava pristupa na osnovu uloge koju imaju to možemo učiti mehanizmima autorizacije
- Tačke pristupa (metode kontrolera) možemo osigurati dodavanjem Authorize atributa, uz koji možemo specificirati i ulogu koju korisnik mora da ima kako bi mogao da pozove tu metodu
- Ako ne navedemo ulogu bilo koji autentifikovani korisnik (sa validnim tokenom) će moći da joj pristupi

```
[HttpGet("nesto-asistentsko")]
[Authorize(Roles = "asistent")]
0 references
public IActionResult NestoAsistentsko()
{
    return Ok("Asistent si i mozes da vidis ovo, bravo za tebe!");
}
```

Autorizacija

- Ako želimo da validiramo da li korisnik ima neki Claim koji smo mi napravili možemo to uraditi dodavanjem novog pravila u ConfigureServices

```
services.AddAuthorization(options =>
{
    options.AddPolicy("SamoOdabrani", policy => policy.RequireClaim("Neki_moj_claim"));
});
```

- Zatim to pravilo postavimo kao obavezno u Authorize atributu

```
[HttpGet("nesto")]
[Authorize(Policy = "SamoOdabrani")]
0 references
public IActionResult Nesto()
{
    return Ok("Imas token i mozes da vidis ovo, bravo za tebe!");
}
```

Tokeni i Swagger

JWT io

- Ako želimo da vidimo sadržaj nekog tokena to možemo uraditi na stranici jwt.io, kopiramo token i sa desne strane će nam se prikazati njegov sadržaj
- Ovde čak možemo i menjati sadržaj tokena, ali da li je takav token validan?

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

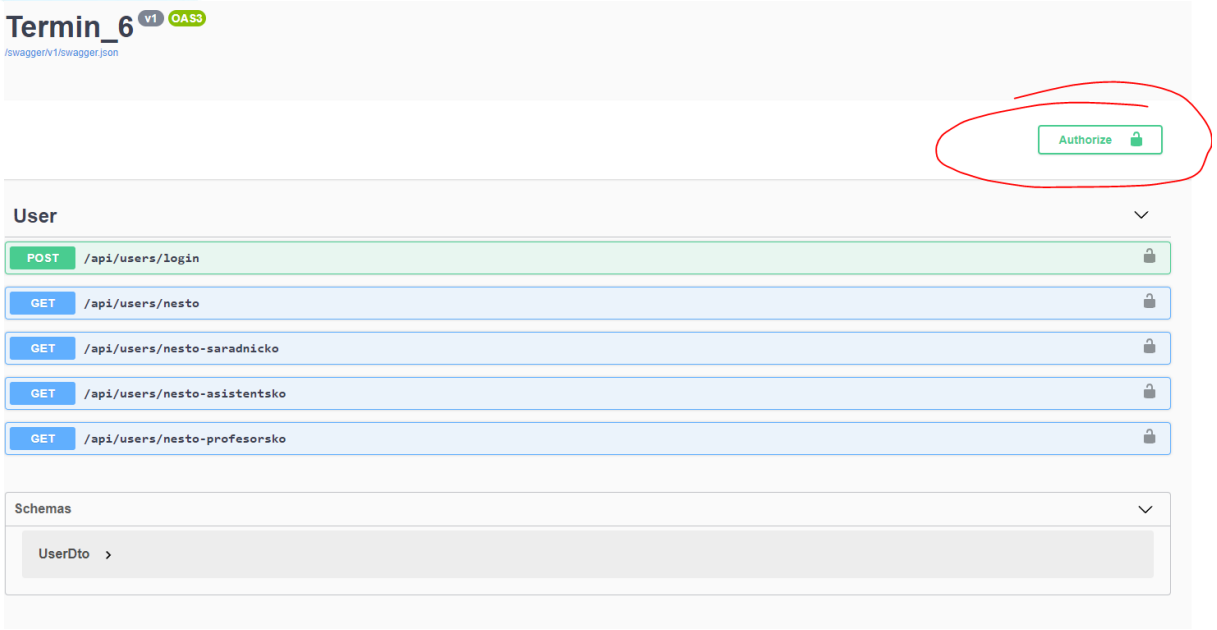
```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) secret base64 encoded
```

Swagger i token

- Token koji nam je izdala aplikacija možemo uneti u swagger i on će ga onda dodavati u svaki HTTP zahtev prilikom testiranja.



Cors

CORS

- CORS (Cross origin resource sharing) je mehanizam kojim server specificira kojim domenima dozvoljava da ga kontaktiraju.
- Ovde recimo možemo podesiti da isključivo frontend naše aplikacije može slati zahteve na zadnjoj strani.

```
services.AddCors(options =>
{
    options.AddPolicy(name: _cors, builder => {
        builder.WithOrigins("https://localhost:4200")//Ovde navodimo koje sve aplikacije smeju kontaktirati
                .AllowAnyHeader()
                .AllowAnyMethod()
                .AllowCredentials();
    });
});
```

Kors podešavanju možemo dati neki naziv i staviti ga kao polje u Startup.cs klasi

```
private readonly string _cors = "cors";
```

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        app.UseSwagger();
        app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "Termin_6 v1"));
    }

    app.UseHttpsRedirection();
    app.UseCors(_cors);

    app.UseRouting();


    app.UseAuthentication();
    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

Termin 6 zadatak

- Promeniti da se korisnici čuvaju u bazi podataka
- Dodati metode za registraciju korisnika i u servisu i u kontroleru
- Korisnički unesene passworde pri registraciji heširati i smestiti u bazu zajedno sa podacima
- Dodati da novoregistrovani korisnici nemaju „Neki_moj_claim“
- Testirati sistem

THANK YOU