Authentication & Authorization

Authentication זהו תהליך המזהה את המשתמש. המשתמש מספק credentials . שאותם משווים לנתונים השמורים ב DB או מקור אחר. אם הם מתאימים המשתמש מזוהה בהצלחה.

Authorization זהו תהליך האחראי לאיזה משאבים המשתמש יוכל לגשת. מה המשתמש מורשה לעשות.

ב ASP.NET Core יש באופן מובנה middleware שאחראי לכך. חלק מהפעולות שהוא עושה: מזהה את המשתמש, מחזיר תשובה מתאימה למשתמש לא מזוהה ועוד.

ישנן סוגים שונים של authentication - (schemes). לדוגמה: jwtbearer, windows, cookies

עבור כל אחד מהם יש service אותו ניתן להוסיף לIServiceCollection

Authentication

כדי להירשם ל middleware הזה נקרא ל UseAuthentication ב program.cs כמובן לפני כל middleware שתלוי במשתמשים מזוהים:

```
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
app.UseEndpoints();
```

באמצעות ה Authentication שמספק לנו את ה ClaimsPrincipal נוכל להגדיר את ההרשאות.
משתמש שעובר אימות באופן אוטומטי מאותחל אובייקט ה user ב HTTPContext

דוגמה למימוש של jwt

```
services.AddAuthentication(options =>
        {
            options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
        })
        .AddJwtBearer(cfg =>
        {
            cfg.RequireHttpsMetadata = false;
            cfg.TokenValidationParameters =
FbiTokenService.GetTokenValidationParameters();
        });
```


כל.הזכויות.שמורות

יצירת token:

```
 private static SymmetricSecurityKey key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes("SXkSqsKyNUyvGbnHs7ke2NCq8zQz
NLW7mPmHbnZZ"));
    private static string issuer = "https://fbi-demo.com";
    public static SecurityToken GetToken(List<Claim> claims) =>
      new JwtSecurityToken(
        issuer,
        issuer,
        claims,
        expires: DateTime.Now.AddDays(30.0),
        signingCredentials: new SigningCredentials(key,
SecurityAlgorithms.HmacSha256)
      );
```

ולידציה לtoken:

```
public static TokenValidationParameters
GetTokenValidationParameters() =>
      new TokenValidationParameters
      {
        ValidIssuer = issuer,
        ValidAudience = issuer,
        IssuerSigningKey = new SymmetricSecurityKey(key),
        ClockSkew = TimeSpan.Zero // remove delay of token when expire
      };
```

Authorization

בעת עלינו לממש את ההרשאות.

לצורך נגדיר policy ואיזה claim הוא דורש.

```
services.AddAuthorization(cfg =>
```

```
        {
            cfg.AddPolicy("Admin", policy => policy.RequireClaim("type", "Admin"));
            cfg.AddPolicy("Agent", policy => policy.RequireClaim("type", "Agent"));
            cfg.AddPolicy("ClearanceLevel1", policy =>
policy.RequireClaim("ClearanceLevel", "1", "2"));
            cfg.AddPolicy("ClearanceLevel2", policy =>
policy.RequireClaim("ClearanceLevel", "2"));
        });
```

לאחר שהגדרנו נוסיף תגית של [Authorize] שמקבלת את הpolicy הרצוי מעל הcontroller/action.
לדוגמה:

```
[Authorize(Policy = "Agent")]
public class AgentController : ControllerBase
```


כל.הזכויות.שמורות