

Examensarbete

**Systemutvecklare - Objektorienterad systemutveckling
400 YH-poäng
Vårterminen 2022**

**Titel: ASP.NET Core Identity och IdentityServer4 för användarens autentisering
Författare: Payam Shaker**

Författare: Payam Shaker
Titel: : ASP.NET Core Identity för användarens autentisering
Handledare: Alexis Flach



Sammanfattning:

Säkerställning av webbapplikationer, med flertal användare som interagera mot skyddade resurser, är en viktig del av systemutveckling. När det gäller .NET systemutveckling är det ASP.NET Core Identity som kan hjälpa till att autentisera och auktorisera av användaren. IdentityServer4, ett OpenID Connect (OIDC) och OAuth 2.0 ramverk för ASP.NET Core, kan också användas för att säkerställa applikationen genom token-baserad autentisering. Detta projektarbete undersöker grundläggande terminologi och viktiga konfigurationer av dessa ramverk, vilket resulterar i en klar bild för den som vill använda dem för första gången.

Innehållsförteckning



Inledning	3
Bakgrund	3
Syfte alt. frågeställning	3
Metod	4
Resultat	5
ASP.NET Core Identity	5
IdentityServer4	7
Diskussion alt. arbetsbeskrivning	10
Sammanfattning och slutsatser	10
Referenser	11
Bilagor	12

1. Inledning

Att säkerställa webbapplikationer är en viktig del av systemutveckling, särskilt när flertal användare ska interagera med system, då säkerhet blir en viktig fråga. När det gäller .NET systemutveckling är det ASP.NET Core Identity som spelar en viktig roll i säkerställning av applikationen. ASP.NET Core Identity är en API (Active Programming Interface) som stödjer login funktion från UI. Det kan användas också av IdentityServer4 som tillhör Duende Software nu för att hantera lösenord, profiluppgifter, s.k. roles, claims, tokens, m.m. Login uppgifter (användarnamn, lösenord och profiluppgifter) kan lagras antingen i Identity SQL tabeller eller genomförs med externa tjänster, såsom Facebook, Google, Microsoft Account och Twitter.

1.1. Bakgrund

Under min första LiA-period på Techlyceum arbetade jag med ett e-handelsprojekt som i primärt steg väckte mitt intresse av "Authentication" och "Authorization" när det gällde användarens login och autentiseringsprocess. I verklighet är det en säkerhetsfråga att användaren (inte) får tillgång till de säkra delarna av systemet/applikationen, dvs "Authorized controllers" i ett API. Data måste skyddas och det kan föras med hjälp av identifierings tjänster som redan finns i .NET Core. Min nyfikenhet att gräva mer om detta ämne ledde mig till att få mer kunskap och bättre förståelse av ASP.NET Core Identity.

1.2. Syfte alt. frågeställning

Syftet med detta arbetet är att samla grundläggande information kring Identity och ge en övergripande bild av Identity användning genom SQL tabeller i en .NET Core webbapplikation. På vägen till målet ska jag undersöka olika begrepp och terminologi och visa viktiga konfigurationer i ett .NET Core projekt. Detta ska tillsammans bilda ett resultat i form av sammanfattning av vad säkerhetsställning är och hur använder det i "real-world" applikationer och säkerställa applikationer genom ramverk/verktyg som finns i .NET Core (ASP.NET Core Identity och IdentityServer4) och undvika "security breach" och ytterligare säkerhetsutmaningar.

2. Metod

I följande avsnitt redogörs viktiga kunskaper som krävs för att komma igång med Identity i ett webbapplikation projekt. Med hänsyn till senaste uppdateringar i .NET har jag valt att visa praktiska delar/kod användning i både .NET 6 .NET 5.

Först och främst är det viktigt att ha en tydlig bild av vad autentisering betyder och hur det skiljer sig från auktorisering. Enligt Microsoft documentation¹ är autentisering definierad som en process där:

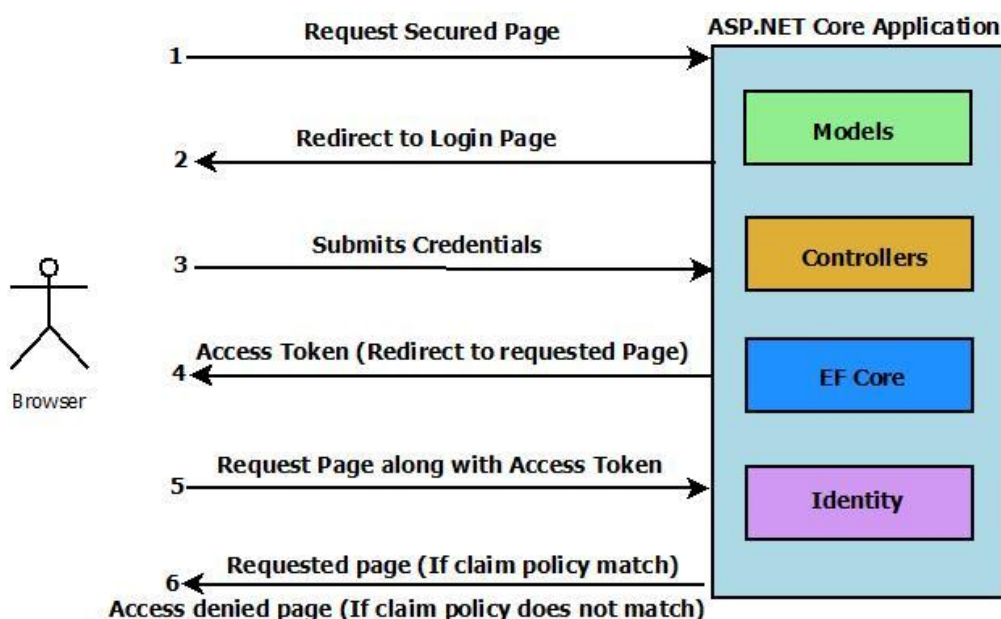
“a user provides credentials that are then compared to those stored in an operating system, database, app or resource. If they match, users authenticate successfully, and can then perform actions that they're authorized for, during an authorization process.”

Auktorisering (Authorization) är nästa steg i processen, som är en separat process i sig själv, där:

“determines what a user is allowed to do.”

I andra ord, “authentication is the process of verifying who a user is, while authorization is the process of verifying what they have access to.”²

Följande bild är ett lysande exempel på hur användaren autentiseras i sin interagerande med bakomliggande API:



Figur 1. Autentisering av användaren i interagerande med ASP.NET Core applikation ³

När man försöker få tillgång till ett säkert API måste man skicka med token som bekräftar att ens identitet har varit autentiserad. En del av denna process handlar om autentisering, dvs “vem är den som vill få tillgång till API:et?”. Dessutom är det nödvändigt att API:et dubbelkollar om alla egenskaper av denna användare är matchade mot uppgifter som antingen finns i SQL tabeller eller kan hämtas från

¹ <https://docs.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-6.0>

² <https://auth0.com/docs/get-started/identity-fundamentals/authentication-and-authorization>

³ <https://procodeguide.com/programming/aspnet-core-identity-claims/>

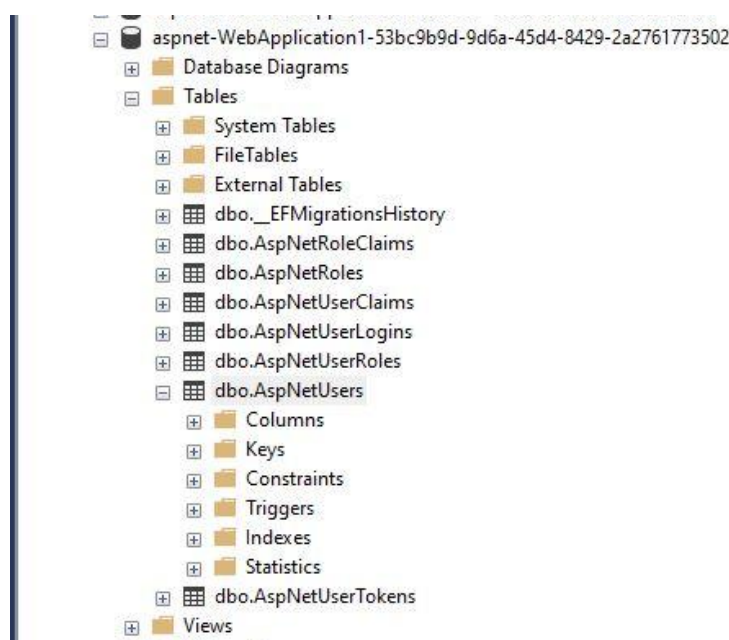
IdentityServer4. Claim är en sådan egenskap som hjälper till att auktorisera användaren. Om man tänker om autentisering som passet man identifiera sig på flygplats är det auktorisering som boarding pass som låter personen flyga, där flygnummer är ett egenskap (claim) kopplat till personen.

3. Resultat

3.1. ASP.NET Core Identity

För att få en tydligare bild av Identity ska jag visa stegvis hur lägga till ASP.NET Core Identity till en .NET Core 6 webbadapplikation projekt och beskriva varje steg i detta avsnitt.

Ett smidigt sätt att använda sig av ASP.NET Core Identity är att aktivera Individual User Accounts vid skapande av webbadapplikation projektet ⁴. Därigenom ska applikationen lägga till två nödvändiga paket (Microsoft.AspNetCore.Identity.UI och Microsoft.AspNetCore.Identity.EntityFrameworkCore)⁵ till Dependencies, vilka sker login och autentiseringsprocess. Genom databasmigrering kan man skapa Identity tabeller där användarens uppgift, roles, claims och token lagras (Figur 2).



Figur 2. Identity tabeller i ett webbadapplikation projekt (Skärmbild)

Vidare är det viktigt att konfigurera Identity middleware i Program.cs för att få tillgång till olika funktioner (User Register, Login, Confirmation, Forgotten password, bl.a.) som Identity förser applikationen med. Figur 3 visar viktiga Identity inställningar i Program.cs, som utgångspunkt i applikationen. Viktigaste är att konfigurera AddDefaultIdentity metod i Program.cs för att introducera paketet till .NET 6 projekt. Denna metod innehåller 3 viktiga metoder (AddIdentity, AddDefaultUI, AddDefaultTokenProviders) som introducerades sedan .NET Core 2.1.

Tack vare Microsoft.AspNetCore.Identity.UI paket går det att registrera en ny användare och sedan logga in med loginuppgifter som har sparats i AspNetUsers tabell in i applikationen (Figur 4 och 5) från UI.

⁴ <https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio>

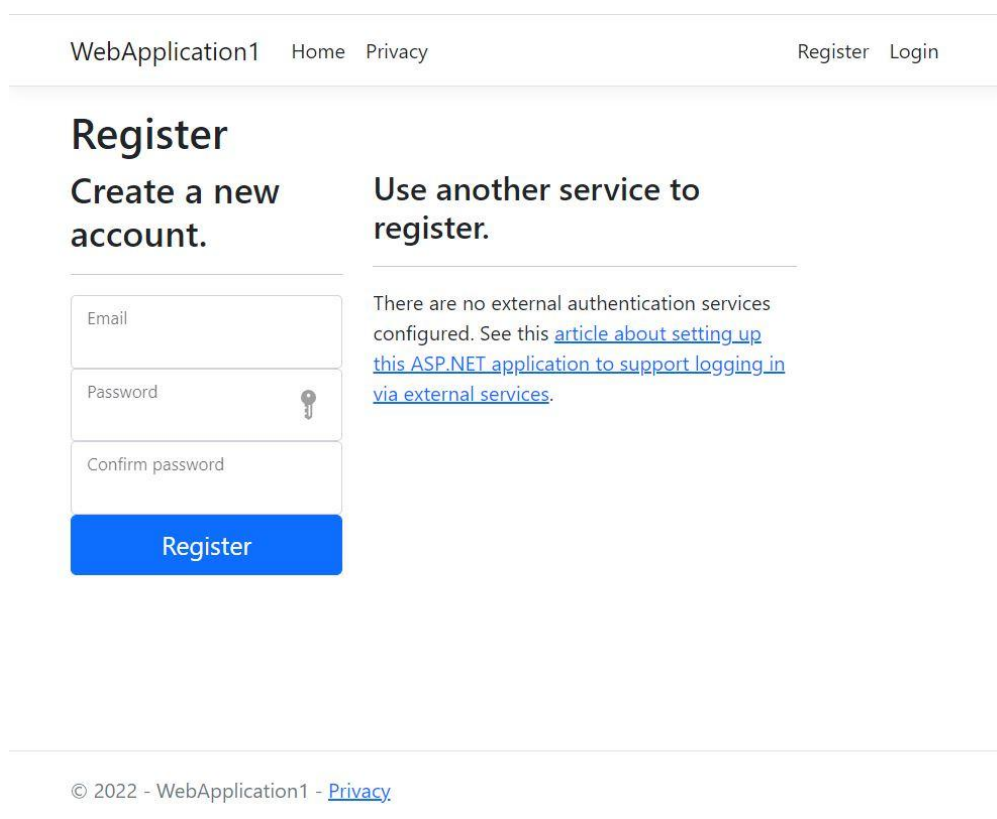
⁵ <https://www.nuget.org/packages/Microsoft.AspNetCore.Identity.UI> och <https://www.nuget.org/packages/Microsoft.AspNetCore.Identity.EntityFrameworkCore>

```
builder.Services.Configure<IdentityOptions>(options =>
{
    // Password settings.
    options.Password.RequireDigit = true;
    options.Password.RequireLowercase = true;
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequireUppercase = true;
    options.Password.RequiredLength = 6;
    options.Password.RequiredUniqueChars = 1;

    // Lockout settings.
    options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);
    options.Lockout.MaxFailedAccessAttempts = 5;
    options.Lockout.AllowedForNewUsers = true;

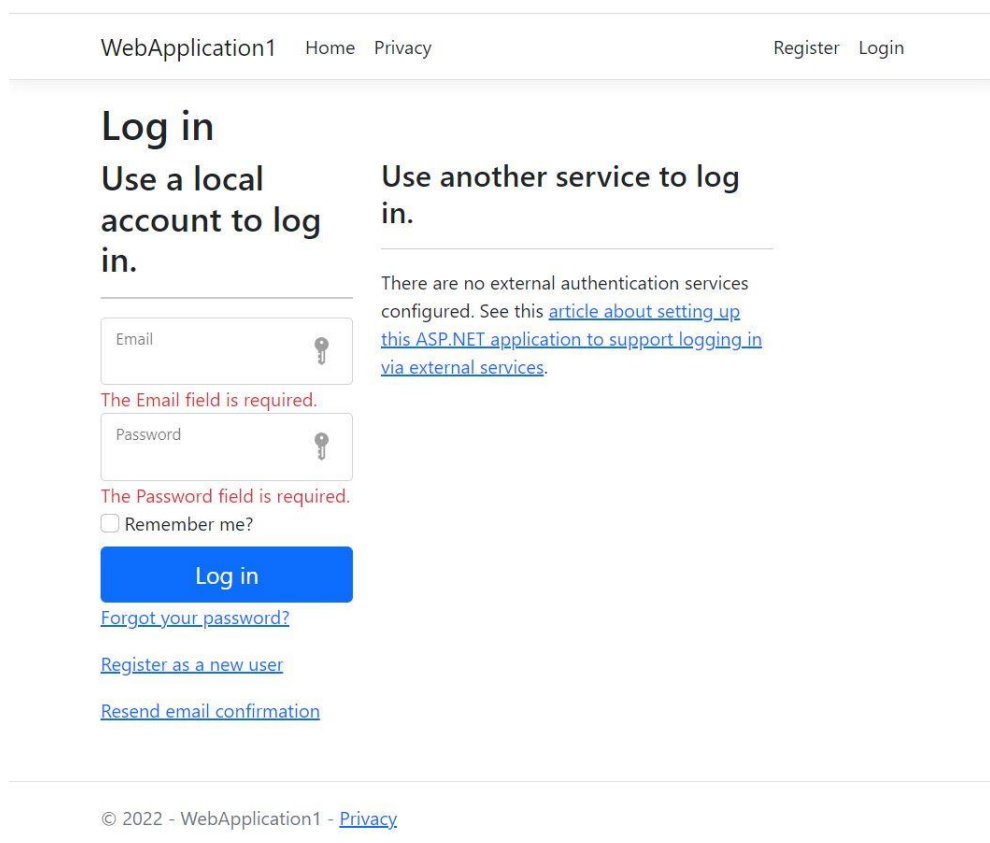
    // User settings.
    options.User.AllowedUserNameCharacters =
    "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-._@+";
    options.User.RequireUniqueEmail = false;
});
```

Figur 3. Konfigurering av Identity i Program.cs (Skärmbild)



Figur 4. Registreringssida (Skärmbild)

När användaren loggar in i applikationen skapas en cookie med användarens claims (såsom custom-claims som kan skapas genom IdentityUser objekt). Det kan också lägga till roller som är kopplade till claims. Cookie claims raderas bort när användaren loggar ut ur applikationen.



Figur 5. Login sida (Skärmbild)

Claims kan skapas utifrån användarens eller identity uppgifter som utfärdas eller skapas. Claim är ett "name value" par som förklarar vad subjektet är, inte vad det kan göra. I första delen av detta avsnitt pratade vi om pass kontra boarding pass exempel för att förklara skillnaden mellan autentisering och auktorisering. Om flygnummer är namn på en claim är det själva "flygnummret" claim värde.

Att skapa claims går att göra med egenskaper som kan skapas under IdentityUser objekt i ASP.NET Core Identity paket. Ett exempel är det födelsedatum som vi vill koppla till alla användarna av applikationen. En sådan claim, alltså ett sådant egenskap, kan läggas till genom ett nytt objekt som ärver IdentityUser.

Ett enkelt test för att se om en skyddad sida (Controller) redigerar till login sida är att lägga till attributen [Authorize] på kontrollen som måste autentiseras.

Ett annat sätt att autentisera användaren, alltså UI requests som skickas till applikation/server, är token-baserad autentisering. IdentityServer4 är ett sådant ramverk som hanterar autentisering.

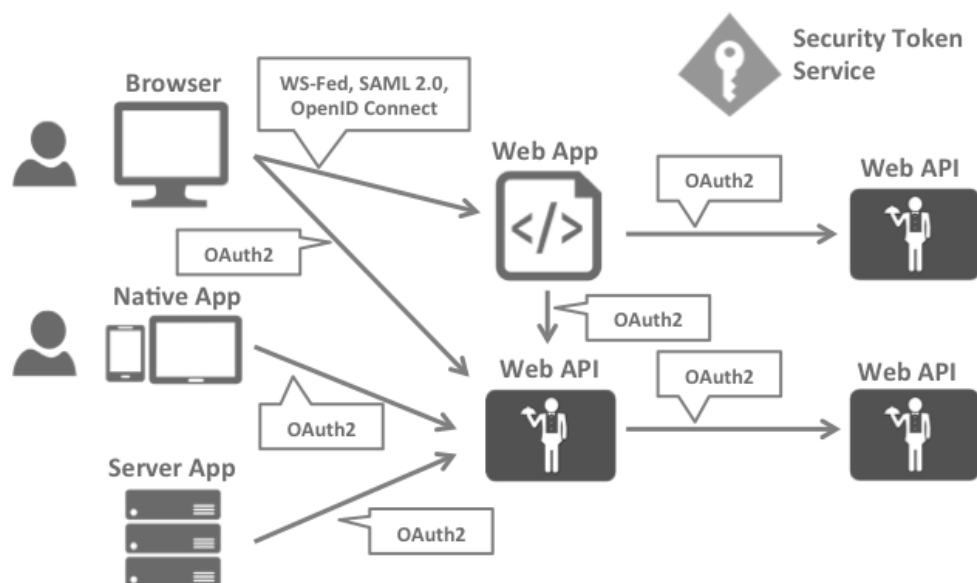
3.2. IdentityServer4

IdentityServer4 är ett OpenID Connect (OIDC) och OAuth 2.0⁶ ramverk för ASP.NET Core⁷ (Figur 6). Det förser applikationen med en autentiseringstjänst som kan användas för att bygga en login-logik. Denna

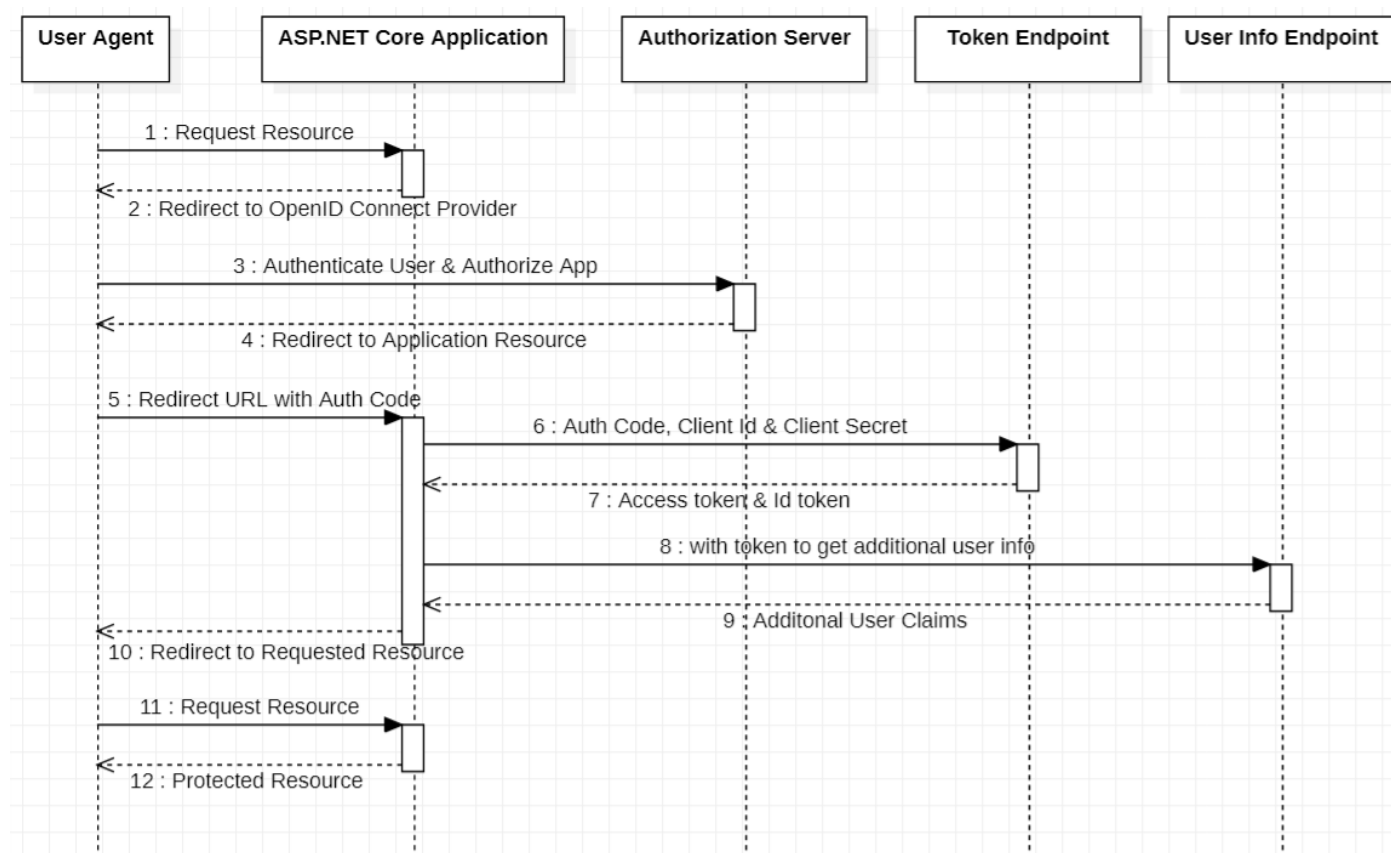
⁶ som är båda autentisering eller identifierings protokoll.

⁷ <https://docs.identityserver.io/en/latest/index.html>

logik baserar på token och claims som ska beskrivas i detta avsnitt. Figur 7 ger en större bild av hela processen⁸.



Figur 6. Applikationssäkerhets arkitektur



Figur 7. Auktoriserings flöde diagram - IdentityServer4

⁸ https://docs.identityserver.io/en/latest/intro/big_picture.html

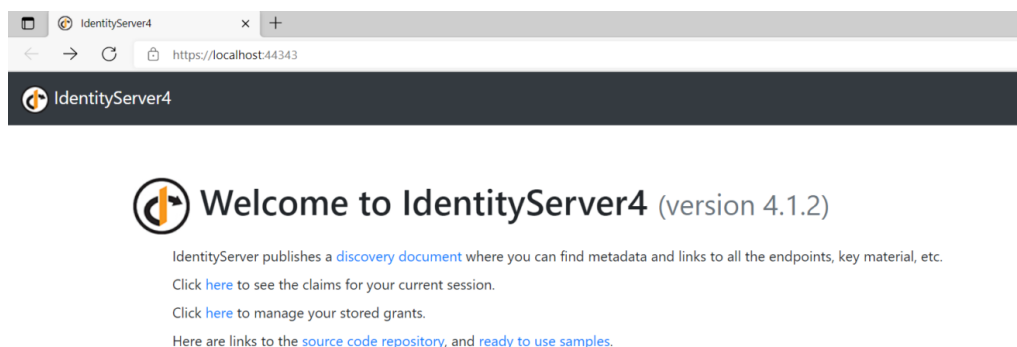
Figur 7 visar vad händer när användaren försöker nå en resurs som är skyddad. En skyddad controller i ett API kan vara ett exempel. Login process är det första steget som användaren måste genomföra. Efter att ha loggat in i systemet skickar IdentityServer4 en "Auth code" tillbaka till användaren, vilken i samband med Client id och secret skickas igen till applikationen för att få token från token endpoint. Client är egentligen namnet på API som är skyddad och är definierad hos IdentityServer4 i SQL tabeller. Client kan döpas till API namnet och har sin egen Secrets och Scopes. Scope är de åtgärder som användaren kan föra mot API:et, exempelvis om användaren får att läsa/skriva på API:et eller köra CRUD operationer. Koden i figur 8 visar ett client exempel.

Efteråt genereras access och id token från token endpoint som skickas tillbaka till applikationen. Gäller token är det dags nu att fråga på användarens claims, dvs. ytterligare egenskaper som kan läggas till till användarens profil, från User endpoint. Dessa claims lagras i cookie form och ger användaren behörighet att få tillgång till de skyddade resurserna.

```
1 reference
public class Clients
{
    1 reference
    public static IEnumerable<Client> Get()
    {
        return new List<Client>
        {
            new Client
            {
                ClientId = "weatherApi",
                ClientName = "ASP.NET Core Weather Api",
                AllowedGrantTypes = GrantTypes.ClientCredentials,
                ClientSecrets = new List<Secret> { new Secret("ProCodeGuide".Sha256()) },
                AllowedScopes = new List<string> { "weatherApi.read" }
            },
        },
    }
}
```

Figur 8. Kod som visar Client konfiguration i IdentityServer4 (Skärmbild)

Att använda IdentityServer4 är det möjligt genom att installera senaste versionen (nu 4.1.2) av IdentityServer4 paket⁹ i ett .NET Core, .NET 5 eller 6 projekt. IdentityServer4 kommer med användarvänlig UI (Figur 9) där man kan se uppgifter om client, claims och token, som hör till användaren som har loggat in i systemet. För att lägga till UI till projektet ska man bara använda sig av open-source källkoden som finns i IdentityServer4 GitHub konto¹⁰.



Figur 9. IdentityServer4 UI

⁹ <https://www.nuget.org/packages/IdentityServer4/>

¹⁰ <https://github.com/IdentityServer/IdentityServer4.Quickstart.UI>

4. Diskussion alt. arbetsbeskrivning

Först och främst är det värt att diskutera hur olika begrepp kan växlas ihop när det gäller applikationens säkerhet. Autentisering, auktorisering, claims, client, m.m. är olika begrepp som krävs vidare förklaring innan man kan börja med ramverket. Dessa begrepp är väsentliga för att förstå mekanismen bakom OIDC och OAuth2.0 protokoll. Det är varför vi började metod avsnitt med en kort genomgång av viktiga terminologi. Fördelen med ASP.NET Core Identity och IdentityServer4 är bra dokumentation som finns på Microsoft och IdentityServer webbsidor. De är ett stort hjälp att få viktig information för att komma igång med ett projekt som vill implementera säkerhet.

En av svårigheter i att säkerställa systemet är att tänka på nackdelar och fördelar av att jobba särskilt med IdentityServer4. Ramverket brukade vara open-source men nuförtiden ligger det under Duende Software license som företag måste betala för¹¹. Det är nu en budgetfråga för företag att välja mellan IdentityServer4 och andra leverantörer, eller helt enkelt att outsourca autentiseringsprocess till "third-party" tjänster som Google, Facebook, Twitter, osv.

Dessutom är det viktigt att tänka på support perioder för olika IdentityServer versioner som anpassar till .NET Core, .NET 5, eller .NET 6¹². Dessa förändringar kräver ständig uppdatering av API:et som får sin autentisering från IdentityServer. Det är nu ännu svårare att behålla applikationer uppdaterade och anpassningsbara till IdentityServer nya versioner. Det är dock en vanlig fenomen inom IT värld som tjänster och ramverk uppdateras under tiden och det är en utmaning att ändra autentiserings leverantör med en ny tjänst/ett nytt ramverk eller gå i takt med uppdateringar.

5. Sammanfattning och slutsatser

Säkerställning av system och skydda delar av resurserna som användaren interagerar med var en utmaning som jag var nyfiken av att lära mig mer om. Tack vare flertal resurser på nätet i form av tutorials, blogginlägg, och sist men inte minst, Microsoft och IdentityServer dokumentation kunde jag få en bättre bild.

Att inte ha en större bild av hela processen var en utmaning när jag förberedde detta arbete. Men tack vare min förkunskaper inom enklare login system som fungerade med lösenord och användaren uppgifter i vanliga SQL tabeller hade jag en preliminär förståelse om hur användaren autentiseras i en applikation. Dessutom token och skapande av token var inte okänd för mig genom olika projekt och grupparbete där jag sysslade med att få resurserna i ett API, exempelvis med Postman.

¹¹ <https://duendesoftware.com/products/identityserver>

¹²

<https://modlogix.com/blog/identityserver4-alternatives-best-options-and-the-near-future-of-identityserver/#:~:text=IdentityServer%20will%20be%20rebranded%20as,for%20the%20fifth%20service%20version.>

6. Referenser

Microsoft (2022) *ASP.NET Core security topics*

<https://docs.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-6.0> [2022-06-06 2:35]

Auth0 docs (2022) *Authentication vs. Authorization*

<https://auth0.com/docs/get-started/identity-fundamentals/authentication-and-authorization> [2022-06-06 2:35]

Sanjay (2021) *How to Implement ASP.NET Core Identity Claims based Authorization*

<https://procodeguide.com/programming/aspnet-core-identity-claims/> [2022-06-06 3:00]

Microsoft (2022) *Introduction to Identity on ASP.NET Core*

<https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio> [2022-06-06 3:03]

Nuget Gallery (2022) *Microsoft.AspNetCore.Identity.UI 6.0.5*

<https://www.nuget.org/packages/Microsoft.AspNetCore.Identity.UI> [2022-06-06 3:06]

Nuget Gallery (2022) *Microsoft.AspNetCore.Identity.EntityFrameworkCore 6.0.5*

<https://www.nuget.org/packages/Microsoft.AspNetCore.Identity.EntityFrameworkCore> [2022-06-06 3:08]

IdentityServer4 docs (2022) *Welcome to IdentityServer4 (latest)*

<https://docs.identityserver.io/en/latest/index.html> [2022-06-06 3:11]

IdentityServer4 docs (2022) *The Big Picture*

https://docs.identityserver.io/en/latest/intro/big_picture.html [2022-06-06 3:19]

Nuget Gallery (2021) *IdentityServer4 4.1.2*

<https://www.nuget.org/packages/IdentityServer4/> [2022-06-06 3:20]

Github (2020) *Quickstart UI for IdentityServer4*

<https://github.com/IdentityServer/IdentityServer4.Quickstart.UI> [2022-06-06 3:25]

Duende software products (2022) *IdentityServer*

<https://duendesoftware.com/products/identityserver> [2022-06-06 3:27]

Pavel K. Modlogix (2021) *IdentityServer4 Alternatives: Best Options and the Near Future of IdentityServer*

<https://modlogix.com/blog/identityserver4-alternatives-best-options-and-the-near-future-of-identityserver/#:~:text=IdentityServer%20will%20be%20rebranded%20as,for%20the%20fifth%20service%20version> [2022-06-06 3:37]

7. Bilagor

Figur 1.

Sanjay (2021) *Autentisering av användaren i interagerade med ASP.NET Core applikation*

<https://procodeguide.com/programming/aspnet-core-identity-claims/> [2022-06-06 4:23]

Figur 6.

IdentityServer4 docs (2022) *Applikationssäkerhets arkitektur*

https://docs.identityserver.io/en/latest/intro/big_picture.html [2022-06-06 4:50]

Figur 7.

Sanjay (2021) *Auktoriserings flöde diagram - IdentityServer4*

https://procodeguide.com/programming/oauth2-and-openid-connect-in-aspnet-core/#Getting_Started_with_IdentityServer4_in_ASPNET_Core_Setup_Identity_Server [2022-06-06 4:53]

Figur 9.

Sanjay (2021) *IdentityServer4 UI*

https://procodeguide.com/programming/oauth2-and-openid-connect-in-aspnet-core/#Getting_Started_with_IdentityServer4_in_ASPNET_Core_Setup_Identity_Server [2022-06-06 4:57]