FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Projekt

Autorizace a autentizace ve webových aplikacích .NET

Studijní program: 1802T007 – Informační technologie
Studijní obor: AI – Aplikovaná informatika
Autor práce: **Jan Reisiegel**
Vedoucí práce: Ing Jan Kraus Ph.D.

Liberec 2024

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2023/2024**

ZADÁNÍ BAKALÁŘSKÉHO PROJEKTU

Jméno a příjmení:	**Jan Reisiegel**
Název práce:	**Autorizace a autentifikace uživatelů webových aplikací v .NET**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**
Vedoucí práce:	**Ing. Jan Kraus Ph.D.**
Rozsah práce:	**15—20 stran**

Zásady pro vypracování:

1. Proveďte rešerši standardů, protokolů a řešení pro realizaci funkcí, spojených s přihlašováním uživatelů (autentifikace) a správou oprávnění k různým prostředkům (autorizace).
2. Vyberte vhodné kombinace použitelných řešení, porovnejte jejich klíčové vlastnosti a omezení a na výbraných implementujte jednoduché ukazkové systémy pro přihlašování a autorizaci/správu rolí.
3. Shrňte své zkušenosti se zkoumanými technologiemi a vyberte takovou, která se vám bude jevit jako nejperspetivnější pro použití v projektech většího rozsahu.

Seznam odborné literatury:

[1] SUCASAS, Victor, et al. An OAuth2-based protocol with strong user privacy preservation for smart city mobile e-Health apps. In: 2016 IEEE International Conference on Communications (ICC). IEEE, 2016. p. 1-6.

[2] CIRANI, Simone, et al. Iot-oas: An oauth-based authorization service architecture for secure services in iot scenarios. IEEE sensors journal, 2014, 15.2: 1224-1234.

[3] CHANDRA, J. Vijaya; CHALLA, Narasimham; PASUPULETTI, Sai Kiran. Authentication and authorization mechanism for cloud security. International Journal of Engineering and Advanced Technology, 2019, 8.6: 2072-2078.

V Liberci dne .....

.....
Ing. Jan Kraus Ph.D.

Prohlášení

Prohlašuji, že svůj projekt jsem vypracovala samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mého projektu a konzultantem.
Jsem si vědoma toho, že na můj projekt se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.
Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mého projektu pro vnitřní potřebu Technické univerzity v Liberci.
Užiji-li projekt nebo poskytnu-li licenci k jeho využití, jsem si vědoma povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.
Beru na vědomí, že můj projekt bude zveřejněn Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.
Jsem si vědoma následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

23. 5. 2024 Jan Reisiegel

Autorizace a autentizace ve webových aplikacích .NET

Abstrakt

Tento dokument se zaměřuje na autentizaci a autorizaci uživatelů ve webových aplikacích. Obsahuje přehled různých balíčků, které tyto funkce podporují. Dále představuje příklad OAuth serveru implementovaného v .NET a ukázku autentizačního procesu pomocí tohoto OAuth serveru.

Klíčová slova: .NET, autorizace, autentizace, OpenId, OAuth 2.0, IdentityServer

Autorization and Autentification in .NET web applications

Abstract

This document focuses on user authentication and authorization in web applications. It includes a review of various packages that support these functionalities. Furthermore, it presents an example of an OAuth server implemented in .NET, along with a demonstration of the authentication process using this OAuth server.

Keywords: .NET, authorization, authentification, OpenId, OAuth 2.0

Poděkování

Rád bych vyjádřil své upřímné poděkování všem, kteří přispěli k dokončení mého bakalářského projektu. Zvláště pak děkuji vedoucímu mého bakalářského projektu, Ing. Janu Krausovi, Ph.D., za jeho odborné vedení, cenné rady a trpělivost během tvorby projektu. Dále chci poděkovat všem, kteří se podíleli na diskuzích a konzultacích, zejména kolegům a spolužákům, kteří svými připomínkami obohatili obsah této práce.

Obsah

Seznam zkratek

HTTP	Hypertext Transfer Protocol
SHA	Secure Hash Algorithm
REST	representational state transfer
API	Aplication programing interface
RBAC	Role-Based Access Control
ABAC	Attribute-Based Access Control
ReBAC	Relationship-Based Access Control
OIDC	OpenId Connect
WST-STS	Web Services Trust Security Token Service
OAuth	Open Authorization

1 Co je autentizace a autorizace

Autorizace a autentizace nám řídí oprávnění uživatelů v aplikaci. Zatímco autorizace se zabývá omezením funkcí pro uživatele v aplikaci, autentizace se zabývá pouhým přihlášením uživatelů (je-li uživatel znám aplikací).

1.1 Autentizace

Máme-li aplikaci, ve které chceme řídit možnost pohybování uživatelů pouze na oprávněné uživatele, poslouží nám k tomu autentizace, neboli ověření uživatelů. Autentizace se pouze stará o zjištění, zda je uživatel znám v aplikaci, tj. je-li zaregistrován. Pokud zaregistrován je, může se přihlásit v jiném případě je uživateli nabídnuto zaregistrování.

V případě používání autentizace je aplikace propojena s databází. V databázi se uchovávají, kromě dat aplikace, data uživatelů. Minimální údaje (přihlašovací údaje) musí být unikátní, aby nedošlo ke konfliktu při přihlašování. Mnohdy jsou však informace o uživatelích rozsáhlejší (např. datum narození, jméno, příjmení, ...). Pro přihlašování se hodí nejvíce e-mail jako unikátní identifikátor uživatele.

V databázi je také uchováváno heslo k účtu. Heslo, které uživatel zadá při registraci nikdy není uloženo do databáze tak, jak jej uživatel zadal. Vždy musí být uložené v zašifrované podobě např. pomocí šifrovacích algoritmů (Bcrypt, SHA-1, ...). Pro zvýšení zabezpečení lze do hesla zadaného uživatelem, vložit náhodný řetězec, který je uložen v databázi a následně provést šifrování hesla s vloženým řetězcem. Je důležité, aby se stejný řetězec vložil na správné místo při každém přihlašování, aby bylo zaručeno, že při stejném hesle nám vyjde správný hash hesla, který je následně porovnáván s hashem z databáze u daného uživatele.

Proces přihlášení uživatele můžeme chápat HTTP request typu POST. Při odeslání requestu se zjistí zda uživatel existuje, funkce pro zjištění vrátí hodnotu TRUE, nebo neexistuje, vrátí FALSE. Při hodnotě FALSE dojde k oznámení, že uživatel neexistuje a dotázání na registraci nového uživatel. Pokud uživatel existuje, je heslo zadané uživatelem (po případném přidání řetězce) zahashované a porovnané se zahashovaným heslem v databázi. Pokud se hesla schodují, uživatel bude vpuštěn do aplikace, pokud se neschodují je uživateli vyhozena hláška o nesprávně zadaném hesle.

1.2 Autorizace

Autorizace je proce, který dává uživateli přístup ke zdroji dat. Zaručuje, že uživatel, který se ke zdroji dat nemá dostat, se ke zdroji dat nedostane.

Proces autorizace je spojen s procesem autentizace. Při přihlášení dostane uživatel access token, který je pro uživatele specifický. Poté, kdykoli uživatel požádá server o zdroje dat, přiloží do požadavku také access token. Autorizační server access token zkontroluje. Má-li access token přístup k požadovaným zdrojům dat jsou uživateli předány.

Příkladem autorizace, která závisí na identitě uživatele, může být nástup do letadla. Při příchodu k bráně vám nestačí pouze letenka, potřebuje také cestovní pas jako doklad totožnosti. Před projitím brány, její obsluha zkontroluje zda se schoduje jméno na letence se jménem na pasu (Při tomto kontrolování dochází k předpokladu, že váš pas je pravý).

1.2.1 Autorizační strategie

Existuje několik autorizačních strategií, které se v současnosti používají. Mezi nejpoužívanější strategie patří řízení přístupu na základě rolí (RBAC) a atributů (ABAC). Řízení přístupu založené na vztahu (ReBAC)

Řízení přístupu na základě rolí (RBAC) - Pracuje s oprávněními spojenými s rolemi uživatelů. Role je vlastni sbírka oprávnění. Například v systému pro zaznamenávání recenzí je výhodné použít role. Recenzent může recenze psát, ale obyčejný uživatel ne. Zároveň recenzenti mohou být uživatelé, kteří nemají žádné podobné atributy, tudíž by ověření podle atributů nedávalo smysl.

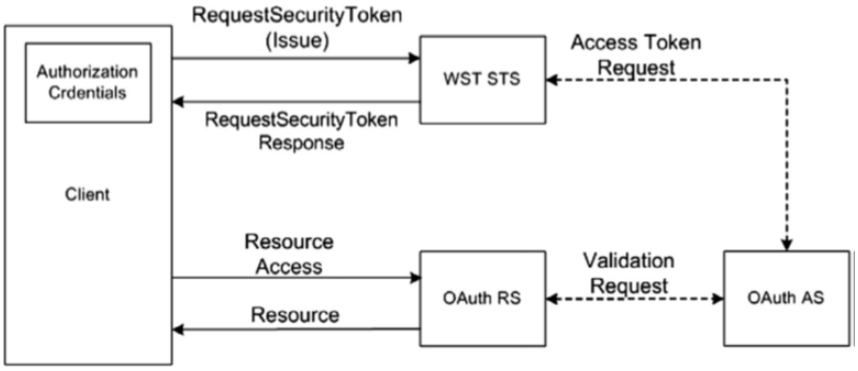
Řízení přístupu na základě atributů (ABAC) - Zda má uživatel oprávnění k provedení akce se zjistí na základě atributů uživatele. Například při nákupu alkoholu je pro elektronický obchod klíčový atribut věk.

Řízení přístupu založené na vztahu (ReBAC) - Řízení přístupu založeného na vztahu, zkoumá zda-li má uživatel dostatečný vztah ke zdroji, aby k němu mohl přistupovat. Vztah může být na základě atributu uživatele, může souviset s rolí uživatele nebo může být vztah přímí (například sdílení dokumentu).

1.3 Ověřování s OAuth

Autorizaci, ať už klientů nebo rovnou uživatelů, můžeme vyžadovat v mnoha různých typech aplikací, od webového REST API přes desktopovou aplikaci až po konzolovou aplikaci. Některé scénáře které bychom mohli použít jsou již dávno definované a popsané. Právě tyto schémata autorizace si teď popíšeme.

V těchto případech již platí, že autentizace i autorizace uživatelů jsou spojené dohromady. Tedy nejsem-li přihlášen, nemůžu být ověřen. Tyto různé návrhy ověřování závisí na mnoha faktorech při použití. Především však záleží na tom, zda



Obrázek 1.1: Zapojení STS-OAuth

máme ověřovací server. Aplikace, které nejsou připojené na některý OAuth serever - namají ověření uživatelů od jiné společnosti např.: Google - těmto scénářům nevyhovují, protože většinou řeší všechny fáze ověřování v jedné aplikaci (i s uchováním dat, pro které jsou uživatelé ověřováni).

Při procesu Ověřování se používá tzv. WST-STS neboli Web Services Trust Security Token Service. WST definuje protokoly a formáty pro zabezpečení komunikace mezi službami pomocí bezpečnostních tokenů. STS je pak služba, která vydává bezpečnostní tokeny, které se následně používají pro autentizaci a autorizaci. Existují dvě možnosti implementace WST-STS s OAuth serverm:

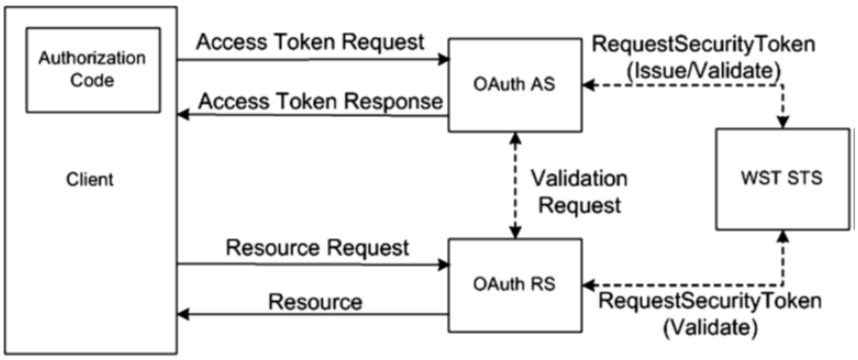
STS-OAuth 1.1 - STS funguje jako autorizační server pro OAuth - klienti nejdříve komunikují s STS, aby získali bezpečnostní tokeny. Klient tedy nejdříve požádá STS o token. STS ověří klienta, při úspěšném ověření vydá klientovi token. Klient tento token použije na OAuth serveru, tak že na základě tohoto tokenu vydá OAuth Ověřovací token, kterým se klient nadále ověřuji při požadování dat

OAuth-STS 1.2 - OAuth serevr funguje jako prostředek pro získávání tokenů. Klient se nejdříve ověří přes OAuth aby získal access token. Následně tento access token klient použije pro získání bezpečnostního tokenu od STS. STS ověří access token a vydá bezpečnostní token, který následně klient využívá pro přístup ke chráněným datům.

Oba dva způsoby mají své výhody a nevýhody, z čehož vyplývá, že se každý způsob používá jinde. STS-OAuth se většinou používá při Enterprise SSO pro přístup do aplikací v rámci podniku, nebo když je vyžadována vyšší úroveň zabezpečení. Na druhou stranu OAuth-STS se častěji vyskytuje při ověřování ve webových či mobilních aplikacích nebo v cloudových službách.

1.4 Vybrané typy ověření uživatelů

Základní ověření heslem - Ověřovací metody, které vyžadují aby si uživatel vytvořil heslo. Toto heslo následně zahashuje funkcí (např. SHA-1, Bcrypt, ...).



Obrázek 1.2: Zapojení OAuth-STS

Zahashované heslo následně uloží do databáze. Při přihlášení odešle uživatel přihlašovací údaje (uživatelské jméno nebo email a heslo). Pokud jsou správné uživatel je ověřen.

Více-faktorové ověřování - Více-faktorové ověření je rozšíření a vylepšení základního ověření. Na rozdíl od základního ověřování má více vrstev/fází, které jsou: ověření heslem, OTP (One Time Password), ověření e-mailu, ověření jedinečné identity nebo speciální otázka. Tyto vrstvy se přidají do autentizačního systému a uživatel jimi musí všemi projít, aby byl autentizován. Příklady více-faktorových ověření jsou např. ověření emailem, kód generovaný z chytrého telefonu, Captcha, QR kód, ...

Ověření pomocí certifikátu - Ověřování pomocí digitálního certifikátu. Digitální certifikát je elektronický dokument, který obsahuje digitální identitu, včetně veřejného klíče a také podpis certifikační autority. Pouze certifikační autorita může vydávat digitální certifikáty.

Biometrické ověření - Když se uživatel chce přihlásit dojde k záznamu a analyzování biometrických údajů uživatele. Při tomto typu ověření se spoléhá na jedinečné biometrické vlastnosti jedince. Mezi běžné techniky biometrického rozpoznání patří rozpoznání obličeje, otisk prstu, rozpoznání řeči, skenery očí a další.

Autentizace na základě Cookies - Při ověřování pomocí cookies je při přihlášení serverem vygenerována relace se speciálním Session ID. Webový prohlížeč (klient) uloží toto Session ID jako cookies. Poté při každé akci, kdy je potřeba zjistit zda je uživatel ověřen se porovná Session ID klienta (z cookies souborů) a serveru, pokud se schodují, server provede akci. Relace zaniká s každým odhlášením uživatele.

Ověření tokenem - Uživatelé odešlou přihlašovací údaje a obdrží zašifrovaný řetězec, který obsahuje údaje o ověření uživatele. Poté se k přístupu používá získaný token namísto opětovného zadávání přihlašovacích údajů. Rozhraní RESTful API, která jsou používána více frameworky a klienty jsou příkladem použití ověřování pomocí tokenů.

Ověření třetí stranou (OAuth, API-Token) - Funguje podobně jako ověřování tokenem. Jedná se ovšem o přihlášení pomocí jiné organizace. Je možné použít protokoly OAuth 1.0 nebo OAuth 2.0, které umožňují přihlásit se k serverům třetích stran jako uživatelé. OAuth 2.0 nabízí zabezpečený přístup ke zdroji na základě uživatele a používá HTTPS protokol pro zabezpečení. Po souhlasu uživatele je vydán ověřovatelem token, který používá aplikace třetí strany k ověřování.

OpenID - OpenID slouží pouze k přihlášení uživatelů. Nejnovější verze OpenID Connect (pro autentifikaci) je přidána do OAuth 2.0 (zaměřeno na autorizaci).

SAML - Security assertion markup language - SAML je založen na XML a je všestranější. Při přihlášení je uživatel odkázán na URL ověřovacího serveru, který po přihlášení přesměruje uživatele na původní stránku s XML daty. Následně je potřeba dekódovat XML.

2 Autentizace a autorizace uživatelů v .NET

Abychom v .NET aplikacích mohli ověřovat uživatele, můžeme si vybrat jakým způsobem to budeme dělat. Můžeme si samozřejmě naprogramovat vlastní algoritmy, které nám budou ověřovat uživatele. Mnohem jednodušší je použít autorizační algoritmy, které již někdo naprogramoval, a u kterých je zaručena funkčnost.

2.1 NuGet balíčky

V této kapitole se zaměřím na vyhledání vhodných balíčků, které umožňují ověřování uživatelů a jejich přihlášení, pro .NET. Každý balíček se hodí na jiné použití. Jelikož .NET aplikace nejsou jenom webové aplikace, je k dispozici celá řada balíčků, které zprostředkovávají přihlašování nebo autorizaci uživatelů.

Při autentizaci a autorizaci uživatelů v .NET je výhodné použít některý z dostupných frameworků. Tyto frameworky se nazývají NuGet balíčky. Níže zmíním pár balíčků, které se hodí pro implementaci autorizace nebo autentizace. Každý balíček se hodí k jiné implementaci.

2.1.1 ASP.NET Identity

Pro ASP.NET je framework ASP.NET Identity, se kterým lze provádět autentifikaci a autorizaci. Zároveň podporuje správu rolí a oprávnění pro autorizaci. Pro použití v projektu ASP.NET je implementace velmi jednoduchá. Tento freamework umožňuje vytvoření databáze s uživateli podle struktury ASP.NET Identity. Framework umožňuje implementaci dvoufaktorového ověřování pomocí usb klíče i externích aplikací. Zároveň je možné vytvářet uživatele, kteří se přihlašují pomocí externího OAuth serveru (Google, Facebook, Microsoft, …)

Zároveň je možné tento framework využít pro vytvoření vlastního ověřovacího serveru. Jen se musejí přidat další balíčky poskytující OIDC a OAuth. Tento framework je tedy dobrý základ ke zprovoznění aplikace, ketrá pracuje s uživvateli, jejich rolemi a dalšími faktory, které jsou používány k autorizaci a autentizaci.

Licence, se kterou je tato skupina balíčků vyvíjena, může být nalezena například na oficiálním GitHub repozitaři vývojáře. Jedná se o MIT licenci, ktrou vývojář určil již před 7 lety, nemělo by se tedy stát, že se tato licence bude měnit.

2.1.2 IdentityServer

Pro vytvoření vlastního OAuth serveru je možné využít otevřenou knihovnu IdentityServer4. Tato knihovna umožňuje spolupráci s ASP.NET Identity, ale také poskytuje svou vlastní Identitu. Knihovna umožňuje také použití pro lokální přihlašování, ale je určena především k vývoji OAuth 2.0 a OpenId serverů.

Lze také využít pro ověřování na serveru se zdrojem dat. Má funkcionalitu založenou na ověřování JWT tokenů. Podle těchto tokenů dokáže zjistit zda klient, který o data požádal, má povolení tato data získat. Vše probíhá pomocí OIDC a OAuth.

Jedná se o starší projekt vývojářů Duende. Nové verze tohoto balíčku již nejsou dostupně a lepší použít Duende. K tomuto přesunu došlo již v roce 2020. Všechna funkcionalita, ovšem stále funguje, jen není plně podporována nejnovější verze .NET. Vývioj se totiž zastavil na .NET 5.0.

Narozdíl od ASP.NET Identity se u tohoto balíčku setkáme s odlišnou licencí a to s Apache 2.0.

2.1.3 Microsoft Identity

Tento framework je založený na identitě od společnosti Miscrosoft. Má možnost být multiplatformní, přičemž pro každou platformu existuje speciální/jiný NuGet balíček. Funguje velmi podobně jako Azure Identity a to z toho důvodu, že oba tyto frameworky vyvíjí společnost Microsoft. Microsoft Identity je tedy předchůdce Azure Identity, svědčí o tom i poslední verze obou balíčků (Mictoroft Identity byl naposled aktualizován v roce 2022, na rozdíl od Azure Identity 2024 a stále je aktualizována)

Jedná se o základní balíček pro práci s Identitami v c. Licencí, kterou jsem našel, je MIT. Lze tedy použít anižbychom museli cokoli platit.

2.1.4 Amazon CognitoIdentity

Amazon se svou službou AWS se samozřejmě nemůže držet stranou. Pro .NET Vyvynul několik NuGet balíčků, které vývojářům usnadňují práci s AWS a vytvoření řešeních se všemi službami Amazon. Samozřejmě nesmí chybět ani možnost ověřování pomocí AWS.

Dle mého názoru k sobě balíčky od Amazonu potřebují ještě vlastní databázi, abychom mohly provádět ověřování společně s přihlášením uživatelů. Bohužel jsem v dokumentaci nenašel jakým způsobem se dá amazon k autorizaci použít, protože mi přijde jejich dokumentace velmi obsáhlá. Je jistě způsobeno tím, že mají větší počet NuGet balíčků a v dokumentaci pro .NET mají zdokumentovány opravdu všechny.

Obdobně jako IdentityServer4 a jistě ještě některé balíčky, Amazon používá licenci Apache 2.0. Je tedy dobré si dohledat ceník služeb pro použití Amazon CognitoIdentity.

2.1.5 Duende IdentityServer

Další možností jak vytvořit vlastní identity server s pomocí balíčku je Duende IdentityServer. Balíček lze využít jak k vytvoření vlastního OAuth serveru s OIDC, tak pouze k ověřování uživatelů v samostatném projektu. Zároveň umožňuje integraci více-faktorového ověření.

Stejně jako u IdentityServer4 je zde možné využít strukturu uživatelů pomocí ASP.NET Identity. Deunde má jednu velkou nevýhodu - je placený. Na oficiálních stránkách je tato nevýhoda velmi dobře schována, neboť není v navigaci stránka, která by odkazovala na ceník. Nicméně po výběru produktu a přescrollování celé stránky se dozvídám, že je placený celkem velkými penězi. Minimální konfigurace dovoluje aplikování pro 5 klientů a neomezené množství uživatelů, ale cena 1.500 $ za rok je již na zvážení.

Samozřejmě existuje i Community verze Duende Identity Serveru. Dokonce by měla být ekvivalentní k verzi Enterprise. Platí zde ovšem některá omezení, především je omezen ziskem společnosti/jednotlivce. Hranice, kdy si nemusíte platit Deunde je nastavena celkem příjemně na 1 milion $. Pokud tedy za rok nevyděláte více než 1 milion $ můžete si bezproblémů užívat Enterprise verzi zdarma s omezením podpory (Comunity verze obsahuje pouze standardní podporu vývojářů).

Jelikož se jedná o novější projekt vývojářů balíčku IdentityServer4, nalezneme i zde licenci Apache 2.0. Základní ceny a podmínky jsem již uvedl výše.

2.1.6 AdminUi

K vytvoření vlastního Identity serveru je možné použít AdminUi. V celekem přehledné dokumentaci mají ukázku toho jak s AdminUi pracovat. Podporuje správu rolí a Calimů uživatelů, ktré se následně využívají k ověřování. Umožňuje také vytvořit OAuth s OIDC.

Jediné co mě u tohoto frameworku zaráží je odkaz na již neaktualizovaný IdentityServer4. Je jasné, že IdentityServer4 je zdarma, ale již od roku 2020 není vyvíjen a aktualizován. Sami vývojáři upřednostňují přechod pod novou vývojovou společnost Duende.

Tento framework má v ceníku 3 kategorie - Starter, Enterprise a Universal (Vzestupně podle ceny). Základní Starter balíček se nastaví po zkušební době, která trvá 30 dní. Na rozdíl od zbylých 2, které se musí platit, nemá neomezený počet uživatelů a klientů. V tomto případě můžeme mluvit o 10 povolených uživatelích a 3 klientských aplikací. Na testovací vývoj by určitě Starter balíček stačil. Na rozdíl od Duende je zde jednotná cena, v librách.

2.1.7 OpenIddict framework

Můžeme také zvolit framework OpenIddict. Tento framework implementuje server pro OAuth 2.0 a OpenId Connect. Tento framework byl původně vytvořen pro .NET 2.1, ale je stále udržován, tudíž je možné jej použít v nových verzích .NET.

Umožňuje implementaci na 3 části - serverová, validační a klientská. V serverové části můžeme hovořit o implementaci při vytvoření OAuth 2.0/OIDC. Validační část dokáže ověřovat implementované tokeny, což znamená, že dokáže Validovat klienty, kteří se snaží přihlásit. Poslední částí je část klienta, kdy framework dokáže připojit externí OAuth 2.0/OIDC servery a zároveň na základě přihlášení uživatelů řídit jejich přístup.

Tento balíček sice podléhá licenci Apache 2.0, je ale specifikováno, že je možné balíček použít pro komerční užití. Můžeme ho tedy považovat za open source.

2.2 Vlastní OAuth server

Jak jsme se dozvěděli v předchozí části, existuje mnoho frameworků a balíčků, které se dají použít pro implementaci OAuth serveru. Ať už se jedná o balíčky, které jsou staršího data vývoje

Vlastní OAuth server, který budeme zprovozňovat, bude moci ukládat uživatele do vlastní databáze. Zároveň je možné do vlastní implementace zahrnout autorizaci uživatelů například podle rolí. Client, který následně používá takto vytvořený server k ověřování uživatelů, nemusí ukládat nové uživatele do své databáze a může vést informaci o přihlášeném uživateli pouze v podobě cookies. Zároveň může používat ověřování rolí nebo claimů.

3 Implementace OAuth serveru

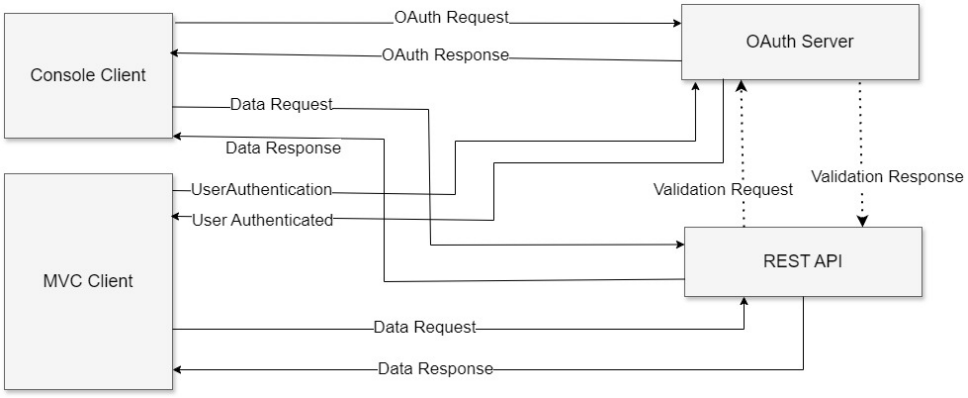
Nyní přejděme k samotné implementaci OAuth serveru v .NET. Jelikož existuje mnoho způsobů jak si vlastní OAuth server neimplementovat, zvolím si jeden ukázkový a zmíním zde zajímavé části kódu. K implementaci použiji knihovnu IdentityServer4.

3.1 Autentizace

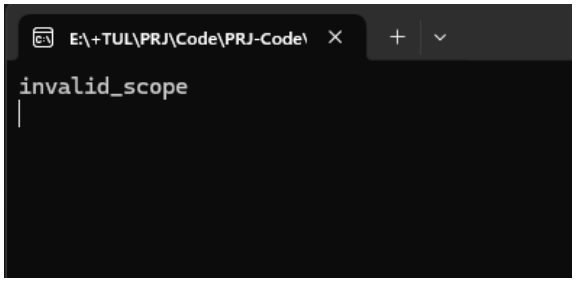
OAuth server v těchto případech pouze ověřuje uživatele/klienty, aby mohli obdržet data z námi vytvořeného REST API (viz obr. 3.1). Toto schéma je jen základní na ověřování klientů. Chtěl-li by požádat o data na našem api nějaký jiný klient, než ten, který je nakonfigurován (má připojený Scope "api"), API server ho odmítne.

Nastavení, zda daný klient má nebo nemá přístup se v kódu nachází v souboru Config.cs, kde u každého klienta definujeme povolené Scopes. Tyto Scopes mají funkci ověřování. Má-li daný klient povolený Scope pro "api1", má povolený přístup k tomuto api. Pokud má přidané další Scopes má více možností ověření. Například v ukázce jsou použité Scopes pro "OpenId" a "profile". Scope pro OpenId přidává přihlašování uživatelů a Scope profile přidává informace o uživateli. Následně je možné si definovat další Scopes, jako například role. Zároveň se musí v konfiguraci přidat příslušný Scope, který hodlá daný klient používat.

V ukázkovém kódu je v projektu OAuthServerMVC naznačeno jak připojit externí OAuth server pro přihlášení na náš OAuth server (v kódu naznačeno pomo-



Obrázek 3.1: Schéma testovacího projektu



Obrázek 3.2: Výpis na konzole při odstraněném ”api” Scope

cí Google). Jedná se o klasický způsob napojení externího přihlašování uživatelů. Podobném způsobem můžeme připojit libovolný počet autorizačních serverů. Tím zaručíme, že se uživatelé na náš OAUth server nemusí přihlašovat ”lokálním” účtem, ale mohou využít služby jiných organizací.

V této podobě se k datům dostane pouze MVC klient, protože v souboru Config.cs v OAuth serveru jsme u konzolového klienta zakomentovali Scope ”api1”. Po spuštění konzolového klienta nám naprogramovaná funkce ve třídě MVC vypíše, že máme nevalidní Scope (viz obr. 3.2), tedy jinými slovy nemáme přístup k api.

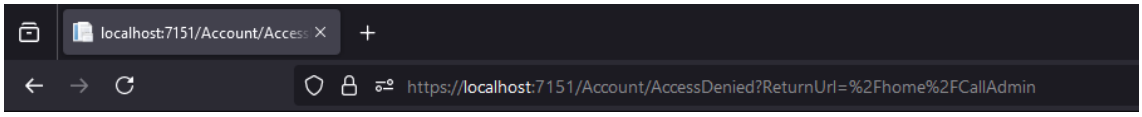
3.2 Autorizace

Autorizace uživatelů je nastíněna v projektu MVC klienta v HomeControlleru. V tomto kontroléru máme 2 metody, které dělají to samé - získávají data z našeho API, jelikož máme pouze jeden Endopoint. Funkce CallApi není nijak zabezpečena, takže ji může spustit každý uživatel.

Autorizaci uživatelů v tomto příkladu provádím pomocí autorizační politiky - Policy. K autorizaci použijeme `[Authorize(Policy = "Admin")]` nad hlavičkou dané funkce. Policy se definují v konfiguraci služeb daného klienta. Autorizační politiku je nutné definovat v konfiguraci klienta, který ji bude používat. Oauth server může pouze připravit Claim, který daná politika bude používat. Příkladem může být role uživatele.

V případě ukázkového kódu, jsem vytvořil roli uživatelům uměle. V klasickém případě by se jednalo nejspíše o další tabulku v databázi. Nejlépe při použití ASP.NET Identity, kde se tato funkcionalita již vyskytuje, a není ji tedy potřeba řešit, jen použít. V případě použití ASP.NET Identity by mohlo fungovat rovnou volání role, ale definování autorizačních politik je používanější.

Pokud uživatel nesplňuje danou autorizační politiku dostaneme Access Denied v url (viz obr. 3.3). Je tedy možné přes kontrolér řídit i tuto stránku. Například zobrazením stránky s příslušným kódem.



Obrázek 3.3: Access Denied po neúspěšné autorizaci

Závěr

Tento bakalářský projekt uvedl základ do problematiky autentizace a autorizace uživatelů. Byly zde zmíněné některé strategie, které se k ověřování uživatelů stále používají. Následná rešerše se zaměřila čistě na možnost implementovat ověření uživatelů v aplikacích .NET. Následně byl v rámci projektu vytvořen vlastní OAuth server pomocí knihovny IdentityServer4. V implementaci OAuth serveru je ukázáno jak řídit přístup k datům pro různé klienty a zároveň je naznačena autorizace uživatelů na jednom z klientů.

V implementaci by se dal také použít Dunde Identity Srever, který je v podstatě novější verzí již zmíněného IdentityServer4. Jedním z rozdílů mezi těmito knihovnami, je ten, že Duende používá pro vytvoření novější metodu ASP.NET a to Razor Pages narozdíl od IdentityServer4, který používá ASP.NET MVC. Funkcionalita těchto dvou knihoven je ovšem velmi podobná, až identická.

Použitá literatura

[1] CIRANI, Simone, Marco PICONE, Pietro GONIZZI, Luca VELTRI a Gianluigi FERRARI. IoT-OAS: An OAuth-Based Authorization Service Architecture for Secure Services in IoT Scenarios. *IEEE Sensors Journal*. 2015, roč. 15, č. 2, s. 1224–1234. Dostupné z DOI: 10.1109/JSEN.2014.2361406.

[2] SUCASAS, Victor et al. A privacy-enhanced OAuth 2.0 based protocol for Smart City mobile applications. *Computers Security*. 2018, roč. 74, s. 258–274. ISSN 0167-4048. Dostupné z DOI: https://doi.org/10.1016/j.cose.2018.01.014.

[3] TORROGLOSA-GARCÍA, Elena, Antonio D. PÉREZ-MORALES, Pedro MARTINEZ-JULIA a Diego R. LOPEZ. Integration of the OAuth and Web Service family security standards. *Computer Networks*. 2013, roč. 57, č. 10, s. 2233–2249. ISSN 1389-1286. Dostupné z DOI: https://doi.org/10.1016/j.comnet.2012.11.027. Towards a Science of Cyber Security Security and Identity Architecture for the Future Internet.

[4] PANT, Piyush et al. Authentication and Authorization in Modern Web Apps for Data Security Using Nodejs and Role of Dark Web. *Procedia Computer Science*. 2022, roč. 215, s. 781–790. ISSN 1877-0509. Dostupné z DOI: https://doi.org/10.1016/j.procs.2022.12.080. 4th International Conference on Innovative Data Communication Technology and Application.

[5] WANG, Shi-Qi, Jing-Ya WANG a Yong-Zhen LI. The Web Security Password Authentication based the Single-block Hash Function. *IERI Procedia*. 2013, roč. 4, s. 2–7. ISSN 2212-6678. Dostupné z DOI: https://doi.org/10.1016/j.ieri.2013.11.002. 2013 International Conference on Electronic Engineering and Computer Science (EECS 2013).

[6] *Popular Authentication Methods for Web Apps* [online]. c2024. [cit. 2024-02-15]. Dostupné z: https://www.baeldung.com/cs/authentication-web-apps.

[7] *What is Authorization?* [online]. 2024. [cit. 2024-02-15]. Dostupné z: https://auth0.com/intro-to-iam/what-is-authorization.

[8] ALLEN, Brock a Dominick BAIER. *IdentityServer4* [online]. 2020. [cit. 2024-05-11]. Dostupné z: https://identityserver4.readthedocs.io/en/latest/index.html.

[9] *NuGet* [online]. 2024. [cit. 2024-05-19]. Dostupné z: nuget.org.

A Přílohy

A.1 Zdrojové kódy

- GitHub Repozitář: github.com/JanReisiegel/PRJ
- Přiložené USB/SD karta