FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

Autorizace a autentifikace uživatelů webových aplikací v .NET

Studijní program:	B0613A140005 Informační technologie
Studijní obor:	Aplikovaná informatika
Autor práce:	**Josef Fryč**
Vedoucí práce:	Ing. Jan Kraus, Ph.D. Ústav mechatroniky a technické informatiky

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Autorizace a autentifikace uživatelů webových aplikací v .NET

Jméno a příjmení:	**Josef Fryč**
Osobní číslo:	M19000012
Studijní program:	B0613A140005 Informační technologie
Specializace:	Aplikovaná informatika
Zadávající katedra:	Ústav mechatroniky a technické informatiky
Akademický rok:	2022/2023

Zásady pro vypracování:

1. Proveďte rešerši stávajících standardů, protokolů a řešení pro realizaci funkcí, spojených s přihlašováním uživatelů (autentifikace) a správou oprávnění k různým prostředkům (autorizace) obecně a zejména v prostředí .NET.
2. S využitím vhodných existujících a veřejně dostupných nástrojů implementujte ukázkové řešení autentifikace v různých prostředích, která podporuje současná platforma .NET (webová aplikace, desktop, mobilní aplikace, služba).
3. Navrhněte a s využitím vhodných nástrojů implementujte flexibilní, a pokud možno i obecný systém pro autorizaci oprávnění ke sdíleným prostředkům.
4. Demonstrujte správnou funkci prezentovaných řešení a jejich připravenost reagovat na změny v průběhu životního cyklu aplikací. Své poznatky stručně a přehledně shrňte v textové zprávě a v jejím závěru diskutujte i možnosti dalšího rozvoje tématu.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

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] EL HAJJ HUSSEIN, Mohammad. *Reproducible Examples for Integration with Keycloak*. 2019.

[3] HUANG, Elmo Xuyun. Permissioned blockchain for data provenance in the supply chain with fine grained data authorisation. 2021.

[4] ARCHIP, Alexandru, et al. RESTful Web Services– A Question of Standards. In: *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2018. p. 677-682.

Vedoucí práce:	Ing. Jan Kraus, Ph.D. Ústav mechatroniky a technické informatiky
Datum zadání práce:	12. října 2022
Předpokládaný termín odevzdání:	15. května 2023

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Černohorský, Ph.D.
vedoucí ústavu

V Liberci dne 12. října 2022

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci 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 mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom 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.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna 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 vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

18. dubna 2023 Josef Fryč

AUTORIZACE A AUTENTIFIKACE UŽIVATELŮ WEBOVÝCH APLIKACÍ V .NET

ABSTRAKT

Tato bakalářská práce se zaměřuje na autorizaci a autentifikaci uživatelů aplikací v prostředí .NET. V rámci práce je provedena rešerše stávajících strategií, standardů, protokolů a řešení pro realizaci autentifikace a autorizace uživatelů s důrazem na možnosti a nástroje dostupné pro platformu .NET.

Klíčovým nástrojem, který byl zvolen pro autentifikaci a autorizaci, je Keycloak. Keycloak je open source řešení pro správu identity a přístupu uživatelů. V rámci této práce byly navrženy dvě ukázkové aplikace (webová a mobilní). Obě tyto aplikace využívají Keycloak pro autorizaci a autentifikaci uživatelů, čímž je demonstrována možnost použití Keycloak v různých typech prostředí, které jsou v současné době podporované platformou .NET.

Závěrem práce je diskuse o možnostech dalšího rozvoje tématu, se zvláštním důrazem na adaptabilitu navržených řešení v průběhu životního cyklu aplikací. Práce ukazuje, jak lze efektivně využít existující nástroje pro správu autentifikace a autorizace a jak lze tato řešení integrovat do různých typů aplikací v rámci platformy .NET.

Klíčová slova: Autorizace, Autentifikace, Keycloak, .NET

ABSTRACT

This bachelor thesis focuses on the authentication and authorization of users in .NET application environments. The work involves a review of existing strategies, standards, protocols, and solutions for the implementation of user authentication and authorization with emphasis on the possibilities and tools available for the .NET platform.

The key tool selected for authentication and authorization is Keycloak. Keycloak is an open source solution for managing user identity and access. As part of this work, two exemplary applications (web and mobile) were designed. Both of these applications utilize Keycloak for user authorization and authentication, thereby demonstrating the potential of Keycloak in various types of environments currently supported by the .NET platform.

The conclusion of the thesis discusses possibilities for further development of the topic, with particular emphasis on the adaptability of the proposed solutions throughout the lifecycle of applications. The work illustrates how to effectively use existing tools for managing authentication and authorization, and how these solutions can be integrated into various types of applications within the .NET platform.

Keywords: Authentication, Authorization, Keycloak, .NET

PODĚKOVÁNÍ

Rád bych poděkoval všem, kteří přispěli ke vzniku tohoto díla.

OBSAH

SEZNAM ZKRATEK

ASP	Active Server Pages Network Enabled Technologies
MAUI	Multi-platform App UI
AMS	Access Management System
SW	Software
HW	Hardware
MAC	Mandatory Access Control
DAC	Discretionary Access Control
ABAC	Attribute-Based Access Control
RBAC	Role-Based Access Control
ReBAC	Relationship-Based Access Control
OIDC	OpenID Connect
SAML	Security Assertion Markup Language
JWT	JSON Web Token
LDAP	Lightweight Directory Access Protocol
RADIUS	Remote Authentication Dial-In User Service
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDP	Identity Provider
OS	Operační systém
API	Application Programming Interface
MVC	Model-View-Controller
EF	Entity Framework
HTML	HyperText Markup Language

SEZNAM OBRÁZKŮ

ÚVOD

CÍL PRÁCE

Cílem této bakalářské práce je najít nebo vytvořit vhodný open source nástroj, který umožňuje autentifikaci a autorizaci uživatelů aplikací v prostředí .NET a je schopen pracovat v intranetovém prostředí. Práce se zaměřuje na teoretické základy autorizace a autentifikace, srovnání existujících nástrojů a následnou implementaci ukázkových aplikací, které využívají vybraný nástroj, v tomto případě Keycloak.

METODOLOGIE

Metodologie této bakalářské práce je strukturována do několika klíčových kroků:

1. Literární rešerše: Prvním krokem je prozkoumat existující literaturu a zdroje k tématu autorizace a autentifikace. Cílem je získat hlubší porozumění pro základní koncepty, standardy a protokoly v této oblasti.
2. Porovnání nástrojů: Druhým krokem je analýza a porovnání existujících nástrojů pro správu identit a přístupu. Tato fáze zahrnuje hodnocení různých kritérií, jako jsou funkčnost, flexibilita, podpora komunity, možnost použití na platformě .NET a schopnost pracovat v intranetovém prostředí.
3. Výběr nástroje: Na základě výsledků předchozího kroku je vybrán nejvhodnější nástroj pro další použití v práci. V tomto případě byl vybrán Keycloak.
4. Implementace: Následuje fáze vývoje, během které jsou vytvořeny dvě ukázkové aplikace (webová a mobilní), které využívají vybraný nástroj pro autentifikaci a autorizaci uživatelů.
5. Evaluace a diskuse: Posledním krokem je hodnocení a diskuse o výsledcích implementace. Tato část zahrnuje také diskusi o možnostech dalšího rozvoje a potenciálních vylepšeních.

1 TEORETICKÁ ČÁST

Teoretická část této bakalářské práce se zaměřuje na poskytnutí teoretického základu, který je nezbytný pro pochopení základních konceptů, standardů a protokolů, které se týkají autentifikace a autorizace uživatelů v aplikacích. Tato část také představuje Keycloak, open source řešení pro správu identit a přístupu, které bylo vybráno pro implementaci v praktické části práce.

První sekce se věnuje základním principům autentifikace a autorizace, vysvětluje jejich význam a roli v kontextu bezpečnosti aplikací. Druhá sekce poskytuje přehled různých standardů a protokolů používaných pro autentifikaci a autorizaci.

Poslední sekce je věnována Keycloak, jeho výhodám, nevýhodám, funkcím a možnostem. Tato sekce také diskutuje o důvodech, proč byl Keycloak vybrán pro tuto práci.

Tato teoretická část je nezbytná pro pochopení praktické části práce a poskytuje potřebný kontext pro diskusi o výsledcích a možnostech dalšího rozvoje.

1.1 AUTENTIFIKACE

Autentifikace je proces ověření identity daného subjektu (v informatice převážně uživatel, zařízení, zpráva, nebo data). Toto ověření vždy závisí na určité formě důvěry.

[1] Při kontrole občanského průkazu musíme důvěřovat jeho vydavateli (obecnímu úřadu obce s rozšířenou působností).

Obecně platí, že čím vyšší bezpečnost je vyžadována, tím silnější autentifikaci je potřeba použít.

1.1.1 Metody autentifikace

[2]Pro zjištění autenticity subjektu se používají metody, které lze rozdělit do následujících čtyř kategorií

- podle toho, co uživatel zná (PIN, heslo, symbol, ...)
- podle toho, co uživatel má (hardwarový klíč, čipová karta, privátní klíč, občanský průkaz, ...)

- podle toho, čím uživatel je (otisk prstu, snímek oční sítnice či duhovky, snímek obličeje, ...)
- podle toho, co uživatel umí (správná odpověď na vygenerovaný dotaz)

1.1.2 Typy autentifikace

SFA (Single Factor Authentication)

Nejzákladnější a nejpoužívanější typ ověření totožnosti. Uživatel je ověřen na základě jednoho kroku (typicky zadání uživatelského jména a hesla).

Tento typ ověřování je možné spatřit u většiny webových aplikací a mobilních aplikací (diskuzní fóra, e-shopy, streamovací služby, ...).

Mezi výhody patří rychlost a nenáročnost ověření pro uživatele. Pro vývojáře je zase výhodou velmi snadný návrh a implementace.

Nevýhodou u tohoto typu je především nízká bezpečnost. Je všeobecně známo, že běžní uživatelé si rádi volí jednoduchá hesla (typicky password, heslo123 a podobné), proto je pro případné útočníky velice jednoduché tuto bezpečnostní vrstvu prolomit.

2FA (Two Factor Authentication)

Oproti SFA přidává 2FA další vrstvu zabezpečení. Pro ověření uživatele jsou tedy potřeba kroky dva. Nejčastěji se jedná o zadání přihlašovacích údajů (uživatelské jméno a heslo) a následné zadání SMS kódu, který byl zaslán na ověřené mobilní číslo, nebo potvrzení přihlášení v ověřovací aplikaci, kterou si uživatel už předtím nainstaloval do svého mobilního zařízení.

Tento typ ověřování používají téměř všechny banky v české republice. Typickým příkladem mohou být aplikace jako George Klíč, RB Klíč, atd.

MFA (Multi Factor Authentication)

Typ ověření, který vyžaduje dvě a více bezpečnostních vrstev (spadá sem i 2FA). Může tedy obsahovat libovolnou kombinaci zabezpečovacích vrstev, což z něj činí typ, který je velice odolný proti většině útoků.

Mezi hlavní nevýhody patří časová náročnost a složitost procesu pro uživatele. Po zadání přihlašovacích údajů typicky následuje ověření pomocí mobilní aplikace, či SMS kódu, které může být následováno zadáním kódu z emailu uživatele a snímkem otisku prstu.

Další velkou nevýhodou je nutnost pamatovat si více údajů, či vlastnit a mít přístup k více zařízení, které jsou potřeba k ověření (mobilní telefon, HW token, čipová karta, ...). S počtem údajů a zařízení rovněž roste riziko zapomenutí některého z údajů, či poškození potřebného HW zařízení.

Přidáním dalších a dalších vrstev tedy docílíme větší bezpečnosti, ale zároveň horší uživatelské zkušenosti.

1.1.3 Způsoby autentifikace

HTTP Basic autentifikace

Jedná se o nejjednodušší způsob autentifikace, který je implementován přímo do prohlížeče.

Uživatel se ověřuje zadáním uživatelského jména a hesla, které je zakódované pomocí formátu base64 a posílá se společně s každým požadavkem na server. Tento způsob autentifikace se nepovažuje za příliš bezpečný právě z důvodu formátování pomocí base64, protože tyto údaje lze velmi snadno rozkódovat. Použití se tedy doporučuje pouze v kombinaci s použitím protokolu HTTPS.

HTTP Digest autentifikace

HTTP Digest autentifikace byla vyvinuta jako nástupce Basic autentifikace. Částečně řeší její problém a to přenášení uživatelského hesla v nezašifrované formě. Tento způsob využívá ověřování pomocí hashe, který je vygenerován z uživatelského jména a hesla. K tomuto hashi se přidává také nonce pro generování různých hashů.

Autentifikace pomocí formuláře

Jedna z nejběžnějších forem autentifikace na webu. Uživatel se ověří zadáním uživatelského jména a hesla do formuláře na webové stránce. Tyto údaje jsou odeslány na server, kde jsou ověřeny a pokud jsou údaje správné, server vytvoří session, které se nadále posílá při každém dalším požadavku na server pomocí cookie. Tato session většinou zaniká v případě, kdy uživatel zavře prohlížeč, nebo se odhlásí.

Tento způsob je bezpečný, pokud jsou údaje odesílány přes šifrované spojení, jako je například HTTPS.

OTP (One Time Password)

Jedná se o přihlašování pomocí unikátního jednorázového hesla, které se po použití zneplatní a je potřeba si vygenerovat nové. O generování takového hesla se většinou starají SW aplikace (Microsoft Authenticator, Google Authenticator, ...), nebo HW zařízení.

OTP se hojně využívá při prvotním přihlášení, které následuje po registraci uživatele. Uživatel je po přihlášení pomocí jednorázového hesla vyzván, aby si nastavil své vlastní bezpečné heslo, které bude používat při každém dalším přihlášení.

Biometrické ověření

Využívá jedinečných biometrických charakteristik jedince, jako jsou například otisky prstů, rysy obličeje, vzor duhovky a jiné.

Ačkoliv se jedná o poměrně bezpečný typ ověření, tak jeho obrovská výhoda, jedinečnost, se stává jednou z jeho největších nevýhod. Svá biometrická data nelze změnit. Pokud by tedy došlo k jejich poškození, či odcizení, tak není možnost biometrická data upravit a začít používat jiná.

Autentifikace pomocí tokenu

Při tomto typu ověřování je po předchozím úspěšném přihlášení vygenerován unikátní token, který slouží uživateli k doložení jeho identity při každém dalším požadavku. Tokenu je vždy nastavená určitá platnost (např. 14 dní) a po uplynutí této doby musí pro obdržení nového tokenu uživatel proces přihlašování opakovat.

Tokeny bývají často kryptograficky podepsány, aby se zajistilo, že nebyly případným útočníkem upraveny a zneužity pro odcizení údajů uživatele.

Autentifikace pomocí certifikátu

Namísto tradičních uživatelských jmen a hesel lze k ověření využít také digitální certifikáty. Certifikáty jsou vytvářeny ověřenou autoritou a jsou zašifrované. Společně s certifikátem je vygenerována dvojice klíčů, soukromý a veřejný. Soukromý klíč zůstává tajný a je chráněn vlastníkem. Veřejný klíč je předáván službě, u které se chce uživatel ověřit. Služba certifikát ověří u autority, která certifikát vydala, aby bylo jisté, že není padělaný. Pokud je certifikát v pořádku, dojde k jeho rozšifrování veřejným klíčem, který služba obdržela od uživatele. Tím se potvrdí totožnost uživatele a povolí se přístup.

1.2 AUTORIZACE

Autorizace je proces, který velmi úzce souvisí na autentifikaci a zároveň na ni také navazuje. Tento proces je klíčovým v řízení přístupů a bezpečnosti informací. Umožňuje určovat, co je jednotlivým uživatelům, nebo skupinám povoleno dělat v rámci systému, nebo aplikace.

Je to neustále probíhající proces, jelikož je nezbytné, aby systém neustále sledoval a ověřoval aktivitu uživatelů, která musí být v souladu s pravidly systému.

V oblasti řízení přístupů a autorizace existuje několik různých přístupů a strategií, které se běžně používají.

1.2.1 Mandatorní řízení přístupu (MAC)

MAC je jedna z nejvíce striktních a robustních strategií. Její použití je běžné v prostředích, která mají vysoké bezpečnostní požadavky. Tato strategie spočívá v tom, že všechna přístupová nastavení se pro každého uživatele nastavují zvlášť. Veškerá tato nastavení může měnit jeden jediný uživatel, který je administrátorem.

Tento fakt činí z MACu jednu z nejbezpečnějších strategií. Nevýhodou je samozřejmě velmi malá flexibilita a škálovatelnost. Tato strategie se hodí převážně pro systémy s malým počtem uživatelů a s malým počtem požadavků na změnu v nastavení.

1.2.2 Volitelné řízení přístupu (DAC)

Velmi flexibilní a uživatelsky orientovaná strategie. Každý uživatel je vlastníkem svých dat a k těmto datům si sám spravuje přístup. Může tedy rozhodovat o tom, kdo může jeho soubory číst, kdo je může editovat a kdo je může mazat.

Nevýhodou této strategie je její náchylnost k bezpečnostním rizikům. V případě nedostatečné pozornosti uživatele při nastavování přístupu může velmi snadno dojít k chybě a k následnému zneužití dat uživatele.

1.2.3 Řízení přístupu na základě vlastností (ABAC)

[3]Při použití strategie ABAC získává uživatel přístup na základě jeho vlastností. Pod takovou vlastností si lze představit například věk, či výšku uživatele.

V případě, že uživatel splní předem definovanou podmínku, typickým příkladem může být věkové omezení, je mu udělen přístup pro danou akci.

1.2.4 Řízení přístupu na základě rolí (RBAC)

Jedna z nejběžnějších strategií, jejíž použití můžeme spatřit prakticky v jakémkoliv komplexnějším systému. V této strategii má uživatel přidělenou jednu, nebo více rolí, které mají jednotlivá oprávnění.

Názorným příkladem může být školní prostředí.

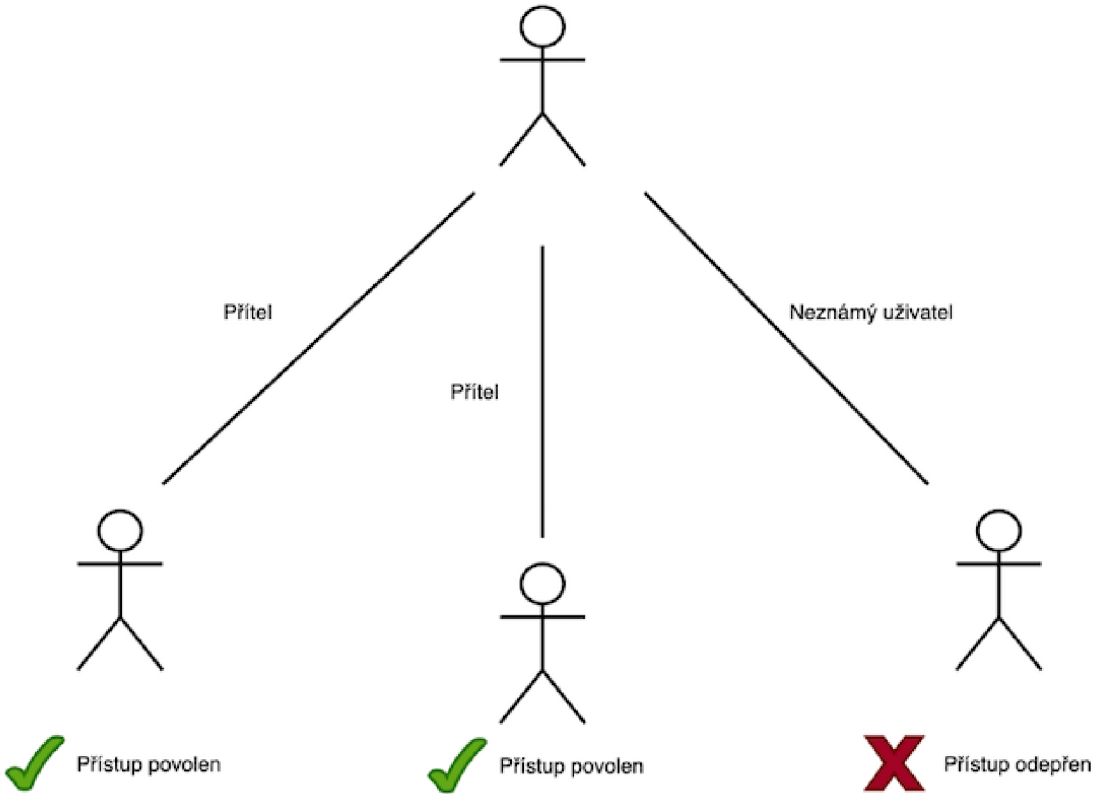
- Ředitel
  - má nejvyšší úroveň a plnou kontrolu nad školou
  - přístup ke všem datům a informacím v systému
  - možnost vytvářet rozvrhy, nabírat nové učitele i studenty
- Učitel
  - přístup ke svým třídám a studentům
  - možnost zadávat známky určité skupině studentů
- Student
  - přístup ke svým vlastním údajům a známkám
  - možnost odevzdávání úkolů a prohlížení svých vlastní výsledků

Struktura těchto rolí má mnoho společného s klasickou hiearchií v reálných organizacích a firmách. To je také jeden z důvodů, proč je použití této strategie velmi časté.

1.2.5 Řízení přístupu na základě vztahů (ReBAC)

RBAC je strategie, která umožňuje rozhodovat o udělení přístupu na základě vztahů mezi subjekty. Tyto subjekty mohou představovat jednotlivce i skupiny. Tyto vztahy jsou reprezentovány jako grafy, kde uzly představují subjekty a hrany představují vztahy, které mezi sebou mají.

Použití této strategie je možné pozorovat na sociálních sítích. Přístup ke svému profilu může být omezen pouze na ”přátele” vlastníka profilu.



Obrázek 1.1: ReBAC: Příklad

Každá z výše uvedených strategií má své výhody i nevýhody. Rovněž každá z nich je vhodnější pro jiný typ systému, či organizace. Při volbě se není nutno omezovat pouze na jednu, ale můžeme využít libovolnou kombinace strategií. Vždy je ale důležité při volbě dbát na potřeby a požadavky na bezpečnost konkrétního systému.

1.3 PŘEHLED STANDARDŮ A PROTOKOLŮ PRO AUTENTIFIKACI A AUTORIZACI

Standardy a protokoly představují dohodnuté postupy, které umožňují kompatibilitu mezi různými systémy a aplikacemi. Jsou potřebné pro zajištění efektivní a hlavně bezpečné komunikace v digitálním světě.

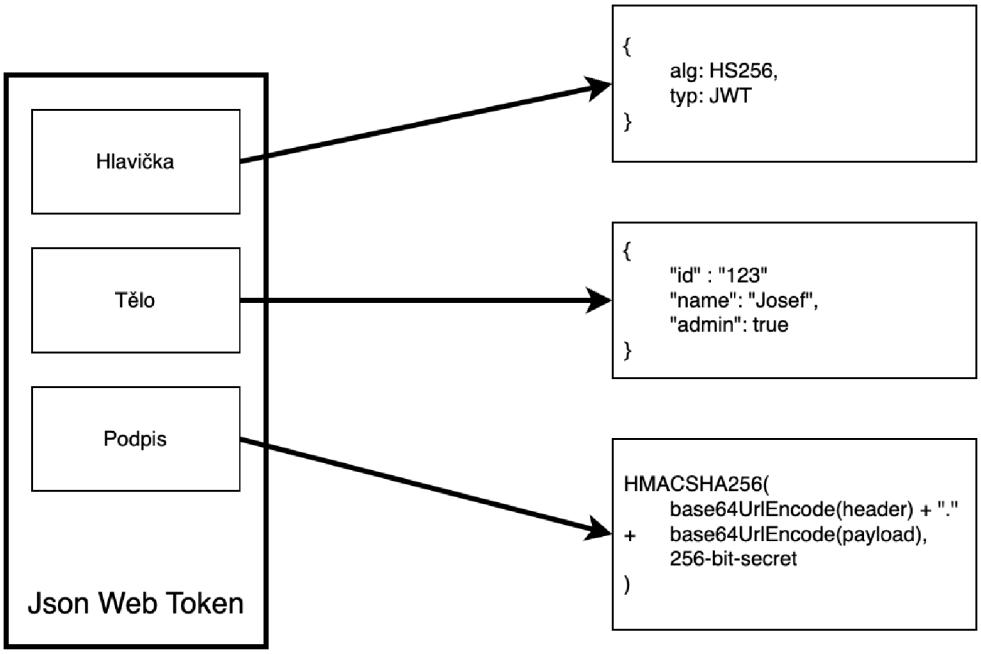
V této kapitole jsou shrnuty nejpopulárnější a nejvíce používané standardy a protokoly pro autentifikaci a autorizaci.

1.3.1 JWT (Json Web Token)

[4]Otevřený standard, který definuje způsob, jakým si mohou dvě strany bezpečně předávat informace ve formě JSON objektu. Důvěryhodnost této informace dokládá digitální podpis. Takový podpis může mít dvě podoby. Podpis pomocí tajného klíče, nebo pomocí klíčového páru (veřejný a privátní klíč).

Struktura JWT

1. Hlavička (Header)
    - typ tokenu (v tomto případě JWT)
    - algoritmus použitý pro podpis (HMAC, SHA256, RSA, ...)
2. Tělo (Payload)
    - údaje o uživateli
    - doplňkové informace
3. Podpis (Signature)
    - zakódovaná hlavička
    - zakódované tělo
    - tajný, nebo veřejný klíč (secret)



Obrázek 1.2: Json Web Token

1.3.2 OAuth 2.0

[5]Standard umožňující aplikacím získat omezený přístup k uživatelským účtům na jiných webových službách (např. Google, Microsoft, GitHub a další) bez potřeby sdílení a ukládání přihlašovacích údajů uživatele. OAuth 2.0 pracuje s protokolem HTTP a může být použit pro autorizaci desktopových, webových i mobilních aplikací.

OAuth 2.0 je tedy protokol určený primárně pro autorizaci, nikoliv pro autentifikaci a neobsahuje tedy žádné mechanismy pro ověření identity uživatele.

Role v OAuth 2.0

1. Klient
    - aplikace, která chce získat přístup k uživatelskému účtu
    - musí být autorizována uživatelem a autentizována poskytovatelem služeb (na základě předchozí registrace a přidělení unikátního client id)
2. Vlastník zdroje
    - uživatel, který autorizuje aplikaci k získání přístupu
    - uživatel může být fyzická osoba, nebo jiný systém

3. Zdrojový server
    - server, na kterém jsou uloženy uživatelské účty
    - reaguje na požadavky o přístup k uživatelským účtům
    - k povolení přístupu musí být součástí požadavku tzv. přístupový token (Access token)
4. Autorizační server
    - server, který ověří identitu uživatele
    - vydává přístupové tokeny po autorizaci aplikace uživatelem

Postup ověření

1. Uživatel je přesměrován z klienta, do kterého se chce přihlásit, na stránku IDP, kde provede přihlášení a následně aplikaci udělí oprávnění k přístupu.
2. Proběhne přesměrování uživatele zpět na klienta s autorizačním kódem.
3. Aplikace posílá požadavek na autorizační server, kde výměnou za autorizační kód získá přístupový token.
4. Získaný přístupový token lze poté použít k získání požadovaných údajů z zdrojového serveru.

1.3.3 OIDC(OpenID Connect)

[6]Dodatečná autentifikační vrsta pro standard OAuth 2.0. Tento standard se tedy na rozdíl od standardu OAuth 2.0 zaměřuje primárně na ověřování identity uživatelů.

[7]Umožňuje jednotné přihlášení (SSO), což velmi zjednodušuje uživatelům proces přihlašování, jelikož pro mnoho různých služeb a aplikací jim stačí pouze jedny přihlašovací údaje.

K samostanému OAuth 2.0 přidává OIDC tzv. identifikační tokeny (ID tokeny), které obsahují údaje o uživateli a zároveň také údaje o proběhlém ověřovacím procesu, jako je například název autorizačního serveru, čas ověření uživatele, platnost tokenu a další.

Používaný způsob autentifikace je pomocí tokenů a všechny tokeny jsou ve standardizovaném formátu JWT.

1.3.4 SAML (Security Assertion Markup Language)

[8]SAML je otevřený standard pro online výměnu citlivých údajů mezi partnerskými službamy. SAML definuje 3 role: uživatel (subject), poskytovatel identity (IDP) a poskytovatel služby (SP).

Postup ověření

1. Uživatel požádá u SP o přístup ke službě
2. SP posílá požadavek na IDP pro ověření totožnosti uživatele
3. Po úspěšném ověření poskytuje IDP SAML Assertion (XML dokument) poskytovateli služby
4. SP obdrží XML dokument obsahující oprávnění daného uživatele

SAML se hojně využívá napříč organizacemi s hlavním využitím pro SSO. Avšak, kvůli závislosti na XML (XML má oproti populárnímu JWT větší velikost a pomalejší zpracování) a jeho složitosti se stále více organizací přeorientovává na jednodušší a flexibilnější řešení, jako je například OAuth 2.0 + OIDC.

1.3.5 LDAP (Lightweight Directory Access Protocol)

[9]Protokol LDAP nebyl primárně navržen pro autorizaci a autentifikaci, ale pro přístup a správu adresářových služeb, neboli distribuovaných databázových systémů. Avšak LDAP právě v těchto systémech umožňuje i vyhledávání uživatelských záznamů. Z tohoto důvodu byl a stále je používán rovněž pro autentifikaci a autorizaci uživatelů.

Autentizace probíhá tak, že obdržené údaje (typicky uživatelské jméno a heslo) se porovnají s údaji, které jsou uložené v databázi a v případě shody je uživatel ověřen.

V případě úspěšného ověření může klient zažádat o vyhledání informace o tom, do jakých skupin uživatel patří. Na základě těchto informací může dále klient rozhodnout, zda je uživatel k dané operaci oprávněn, či nikoliv.

LDAP má však řadu nevýhod, které z něj v dnešní době nečiní nejvhodnější protokol pro autentifikaci a autorizaci. Mezi ně patří například:

- **Bezpečnost** - hesla se v původní verzi protokolu posílají nezašifrovaná
- **Podpora funkcí** - malá podpora moderních funkcí, jako je např. SSO
- **Komplexnost** - s narůstající velikostí systému se LDAP stává velmi složitý na správu

1.4 IDENTITY AND ACCESS MANAGEMENT SYSTÉMY (IAM)

[10]Identity and Access Management, neboli správa identity a přístupů, je nedílnou součástí jakéhokoliv systému. Úkolem těchto systémů je zabezpečení identifikace uživatelů, efektivní řízení a správa přístupu k datům.

V této kapitole je několik příkladů komerčních a open source řešení spolu s hodnocením jejich předností a slabin.

1.4.1 Okta

Okta je cloudové řešení od stejnojmenné firmy, která je jednou z vedoucích firem na trhu v sektoru IAM systémů. Jejich produkt nabízí řešení ověřování identity a správu přístupu uživatelů.

Výhody:

- integrace s mnoha aplikacemi
- škálovatelnost a flexibilita
- silná podpora SSO
- robustní možnost pro MFA

Nevýhody:

- vysoké náklady (neexistuje ani omezená verze zdarma)
- závislost na internetu - výpadek internetu způsobí výpadek celého systému
- komplexnost - jelikož nabízí mnoho funkcí, může nastavení být složité bez předchozích zkušeností

1.4.2 Auth0

IAM cloudový systém založený na otevřených standardech včetně SAML, OAuth 2.0 + OIDC.

Výhody:

- škálovatelnost a flexibilita
- bezpečnost - kvalitní monitoring a podpora MFA
- jednoduchost integrace

Nevýhody:

- závislost na cloudu - sdílení dat s třetí stranou
- náklady - s rostoucím počtem uživatelů se velice rychle zvyšují náklady

1.4.3 Keycloak

open source vyvinutý společností Red Hat. Poskytuje jednotnou platformu pro autentifikaci, autorizaci a správu rolí a práv.

Výhody:

- open source
- podpora standardů
- flexibilita
- jednoduchost integrace

Nevýhody:

- komplexnost - konfigurace a správa mohou být pro řadu uživatelů náročné
- podpora - jedná se open source projekt, takže komerční podpora je omezená a je potřeba se více spoléhat na komunitu

1.4.4 Amazon Cognito

IAM systém poskytovaný společností Amazon Web Services (AWS). Systém nyní funguje pro webové a mobilní aplikace.

Výhody:

- obsáhlá dokumentace
- snadná integrace s dalšími AWS produkty
- flexibilní autentifikace - podpora široké řadu standardů + řada vlastních metod autentifikace

Nevýhody:

- cena - existuje bezplatná verze pro menší aplikace, ale náklady pro aplikace s větším počtem uživatelů mohou být poměrně vysoké
- složitost - konfigurace může být velmi složitá pro uživatele, kteří nemají předchozí zkušenost s AWS produkty
- omezená flexibilita autorizace - oproti autentifikaci jsou možnosti autorizace omezenější, než u konkurenčních IAM systémů
- placená podpora
- absence základních funkcí
  - MFA - zapomenout zařízení
  - MFA - chybějící obnovovací kódy
  - logout metoda v API

1.4.5 Azure Active Directory

Zkráceně Azure AD je IAM systém od společnosti Microsoft a je součástí jejich cloudové platformy Azure.

Výhody:

- dobrá komunikace s podporou
- integrace s Microsoft produkty
- podpora standardů
- podpora MFA a SSO

Nevýhody:

- složitá integrace mimo ekosystém Microsoftu
- cena - cena roste rychle s přidáváním pokročilých funkcí
- závislost na OS Windows - není podpora pro iOS, MacOS, Linux a další

2 PRAKTICKÁ ČÁST

Tato kapitola se zaměřuje na aplikaci teoretických poznatků, na které se zaměřovala teoretická část. Konkrétně se bude zaměřovat na implementaci autorizace a autentifikace v aplikacích vyvíjených na platformě .NET s využitím open source IAM systému Keycloak.

V jednotlivých částech této kapitoly bude postupně rozebrán výběr IAM, jeho nastavení, vývoj jednotlivých aplikací a jejich napojení na zvolený IAM systém Keycloak.

Celý systém simuluje jednoduchý školský systém se studenty a učiteli. Učitel má právo měnit název předmětu a jejich popis. Student má právo si prohlížet nabízené předměty, které jsou uložené v databázi.

Nakonec budou také uvedené možnosti dalšího vývoje a případných vylepšení.

2.1 POUŽITÉ TECHNOLOGIE

- Keycloak (IAM) (verze 20.0.2)
- .NET (verze 6.0)
- ASP.NET (webová aplikace)
- .NET MAUI (mobilní aplikace)
- MySQL (databáze) (verze 8.0.32-1.el8)
- Docker
- Entity Framework (přístup k datům) (verze 7.0.2)

2.2 VOLBA IAM SYSTÉMU

V této části bude probrán rozhodovací proces výběru vhodného IAM systému. Mezi hlavní požadavky patřilo, že systém musí být open source a musí zde být možnost jej provozovat na intranetu. Provoz na intranetu zajistí jeho stabilitu i při výpadku internetu a zároveň nám zajistí vyšší soukromí a bezpečnost našich dat. Data uživatelů tedy nebudou sdílena s třetí stranou.

Po provedení rešerše bylo zjištěno, že nabídka IAM systémů, které splňují předem daná kritéria, je značně omezená. Po tomto zjištění byl vybrán Keycloak.

2.2.1 Keycloak

Obrázek 2.1: Logo Keycloak

Poprvé uveden na trh v roce 2014. Díky open source přístupu se stal populárním a okolo Keycloak se velmi rychle utvořila komunita vývojářů, kteří se začali podílet na jeho vývoji. Postupem času se z Keycloak stal velmi robustní a flexibilní nástroj s širokou řadou funkcí pro autorizaci a autentifikaci uživatelů, který je vhodný pro organizace všech velikostí.

Systém je vyvíjen a udržován společností Red Hat. Celý systém je napsaný v jazyce Java, ale to nebrání v jeho použití v systémech, které jsou vyvíjeny za pomocí jiných technologií.

Podporována je řada moderních a široce používaných protokolů, jako je SAML, OIDC a OAuth 2.0. Vysoká flexibilita zároveň umožňuje integraci s mnoha aplikacemi a platformami. V této práci je použit protokol OIDC.

Keycloak může efektivně fungovat v různých prostředích. Může být nasazen jak na cloudové servery, tak na servery lokální. V lokálním (offline) nasazení jsou však některé funkce omezené, jako například ověřování pomocí externích IDP a přístup k externím službám.

Struktura

Keycloak obsahuje řadu komponent, které jsou organizované do určité hiearchie.

1. Realm
    - nejvyšší úroveň
    - můžeme chápat jako samostatnou instanci IAM systému
    - obsahuje vlastní sadu uživatelů, rolí, klientů, skupin a konfiguračních prvků
    - pomocí více realmů můžeme izolovat sady uživatelů, rolí, skupin a aplikací

2. Klient
    - klient reprezentuje jednotlivou aplikaci, nebo jinou službu
    - obsahuje vlastní konfiguraci a může používat jiné protokoly, než jiní klienti ve stejném realmu
3. Role
    - používané v rámci autorizace (řízení přístupu)
    - můžou být na úrovni realmů, nebo klientů (lze kombinovat)
4. Skupina
    - organizační prvek, který uspořádává uživatele do logických celků
    - může mít přiřazené role, které dostanou všichni uživatelé, kteří jsou součástí skupiny
5. Uživatel
    - jedinec, který se přihlašuje do aplikace (klienta)
    - může mít jednu nebo více rolí
    - může být součástí libovolného počtu skupin

Dokumentace

Oficiální dokumentace je obsáhlá, dobře strukturovaná a řeší většinu problémů, na které vývojář může narazit. Obsahuje detailní vysvětlení, využití a zároveň příklady kódu.

Limity

Každé řešení včetně Keycloak má své limity. Keycloak je postaven na Java technologiích, které jsou známé svými vyššími nároky na výkon.

2.3 NASTAVENÍ KEYCLOAK

2.3.1 Nasazení Keycloak pomocí Dockeru

Nasazení Keycloak pomocí Dockeru je velice jednoduché. Prvním krokem je instalace Dockeru a poté již stačí do terminálu zadat příkaz z oficiální dokumentace.

```
docker run -p 8080:8080 -e KEYCLOAK_ADMIN=admin
-e KEYCLOAK_ADMIN_PASSWORD=admin
quay.io/keycloak/keycloak:20.0.2 start-dev
```

[11]V tomto případě kontejner naslouchá na portu 8080 a je vytvořený administrátorský účet s uživatelským jménem **admin** a heslem **admin**. V produkčním prostředí je samozřejmě vhodné zvolit složitější heslo pro ochranu proti brute force útokům.

2.3.2 Vytvoření realmu

Po nasazení Keycloak je vytvořen nový realm s názvem ”BP”, ve kterém jsou následně vytvořeni dva klienti ”asp-net-app” pro webovou aplikaci a ”maui--app” pro mobilní aplikaci.

2.3.3 Správa uživatelů, rolí a skupin

Jsou vytvořeni testovací uživatelé, role pro čtení, editaci a úpravu předmětů, a dvě skupiny **teachers** a **students**. Pro demonstraci dědičnosti skupin je skupina students vytvořena jako podskupina skupiny teachers. Skupina teachers tedy zdědí všechny role, které jsou přiřazené skupině students.

2.3.4 Nastavení vícefaktorové autentifikace

Pro vícefaktorové ověření je použit způsob OTP. V menu pod položkou **Authentication** je potřeba pod menu **Required actions** zapnout u řádky **Configure OTP** položku **Set as default action** na hodnotu **On**.

Od tohoto momentu je uživatel po přihlášení, popřípadě registraci, vyzván k nastavení generátoru OTP na svém mobilním zařízení. Keycloak nabízí volbu mezi aplikacemi Google authenticator a Free OTP. Poté bude uživatel při každém úspěšném ověření pomocí přihlašovacích údajů vyzván k zadání jednorázového hesla ze svého mobilního zařízení.

2.3.5 Přidání externího poskytovatele identity (IDP)

Pro přidání externího IDP je nejdříve potřeba zaregistrovat naši službu (Keycloak) u daného poskytovatele. Po úspěšné registraci dostane služba přidělené unikátní client_id a client_secret. Touto kombinací údajů ověřuje Keycloak svou totožnost u IDP.

Tyto údaje se také použijí pro přidání IDP v Keycloak. Pod položkou **Identity providers** se vybere požadovaný IDP a zadají se obdržené hodnoty client_id a client_secret.

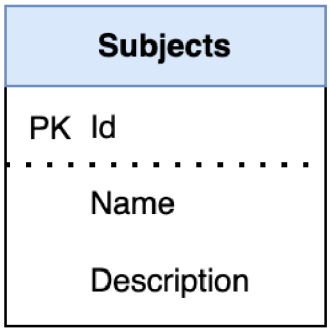
2.4 TVORBA DATABÁZE

Pro tvorbu databáze je použita technologie MySQL. MySQL je open source řešení pro relační databázový systém. Pro nasazení databáze byl použit Docker. Důvod volby MySQL je dostupnost MySQL Docker image, který je kompatibilní s Apple Silicon čipem. Tato kompatibilita je klíčová, protože celý projekt je vyvíjen právě na stroji s tímto čipem.

Databáze je navržena velmi jednoduše pouze pro uchování nejmenšího potřebného množství dat, což jsou v tomto případě vyučované předměty.

Databáze je tedy tvořena pouze jednou tabulkou a to tabulkou **Subjects**. Tato tabulka obsahuje tři sloupce pro základní hodnoty Id, Jméno předmětu (Name) a popis předmětu (Description).

K uložení uživatelů je využito interních mechanismů Keycloak. Ten nabízí interní ukládání uživatelů, rolí a skupin. Struktura celé databáze je tímto značně zjednodušena.



Obrázek 2.2: Tabulka Subjects

2.5 VÝVOJ WEBOVÉ APLIKACE

2.5.1 Specifikace požadavků

Hlavními požadavky bylo:

1. **Integrace s Keycloak**: Uživatel musí být schopen se přihlásit pomocí služby Keycloak.
2. **Řízení přístupu**: Uživatelům je na základě jejich rolí udělen přístup k jednotlivým částem aplikace. Toto zahrnuje možnost číst, editovat a mazat předměty.
3. **Správa předmětů**: Aplikace musí obsahovat funkci pro zobrazení, úpravu a mazání předmětů.

2.5.2 Architektura aplikace

Architektura je založená na návrhovém vzoru MVC (Model-View-Controller).

Modely

Třídy, které popisují objekty uložené v databázi a operace, které je s těmito objekty možné provádět. V případě této aplikace modely představují entity, se kterými je nakládáno pomocí EF.

Pohledy (Views)

U webové aplikace představují HTML šablony, které definují, jakým způsobem budou data prezentována uživateli prostřednictvím webového prohlížeče.

Řadiče (Controllers)

Přijímají vstupy od uživatele prostřednictvím pohledů. Dále tyto změny promítají do modelů a rovněž následné změny v modelech promítají zpět do pohledů.

2.5.3 Implementace

Webová aplikace je vyvíjena na platformě .NET za použití technologie ASP.NET.

Databáze

Pro přístup k databázi a v ní uložených dat je použit Entity Framework. Pro komunikaci s databází slouží třída **RepositoryManager**, která obsahuje další konkrétní repozitáře, které mají na starost manipulaci s konkrétními entitami (např SubjectRepository manipuluje s předměty).

Integrace s Keycloak

Pro autentifikaci pomocí Keycloak je použita knihovna *OpenIdConnectExtensions*.

S použitím této knihovny je nutné pouze nastavit určité parametry autentifikace v souboru **Program.cs**.

```
builder.Services.AddAuthentication(options =>
{
    options.DefaultScheme = CookieAuthenticationDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = OpenIdConnectDefaults.AuthenticationScheme;
}).AddCookie("Cookies")
    .AddOpenIdConnect(options =>
    {
        //Keycloak URL Server
        options.Authority = builder.Configuration.GetSection("KeyCloak:auth-server-url").Value;
        //Client configured in Keycloak
        options.ClientId = builder.Configuration.GetSection("KeyCloak:client_id").Value;
        //Client's Secret configured in Keycloak
        options.ClientSecret = builder.Configuration.GetSection("KeyCloak:client_secret").Value;

        options.Scope.Add("openid");
        options.Scope.Add("profile");

        options.AccessDeniedPath = "/Account/AccessDenied";

        options.RequireHttpsMetadata = false;
        options.SaveTokens = true;
        options.GetClaimsFromUserInfoEndpoint = true;

        //OpenID flow to use
        options.ResponseType = OpenIdConnectResponseType.Code;


        options.Events = new OpenIdConnectEvents()
        {
            OnRedirectToIdentityProviderForSignOut = (context) =>
            {
                var logoutUri =
                    $"{builder.Configuration.GetSection("KeyCloak:auth-server-url").Value}/protocol/openid-connect/logout";

                var postLogoutUri = context.Properties.RedirectUri;

                if (!string.IsNullOrWhiteSpace(postLogoutUri))
                {
                    if (postLogoutUri.StartsWith("/"))
                    {
                        var request = context.Request;
                        postLogoutUri = request.Scheme + "://" + request.Host + request.PathBase + postLogoutUri;
                    }

                    logoutUri += $"?returnTo={Uri.EscapeDataString(postLogoutUri)}";
                }

                context.Response.Redirect(logoutUri);
                context.HandleResponse();

                return Task.CompletedTask;
            }
        };
    });
```

Obrázek 2.3: Nastavení autentifikace v Program.cs

Pro autorizaci slouží třída **KeycloakService**, která je jádrem celé autorizace. Její metoda **Authorize** přijímá jako parametr název oprávnění, které je požadováno, a vrací hodnotu *true*, pokud uživatel má požadované oprávnění, nebo hodnotu *false* v případě opačném.

V rámci třídy jsou oprávnění uživatele uložena v objektu **Token**, který je vytvářen na základě odpovědi od Keycloak. Tento objekt je uložen do paměti

cache, aby byly tyto informace snadno dostupné pro další požadavky. Po vypršení délky uložení v paměti cache je Keycloak znovu dotázán o poskytnutí aktuálních informací uživatele.

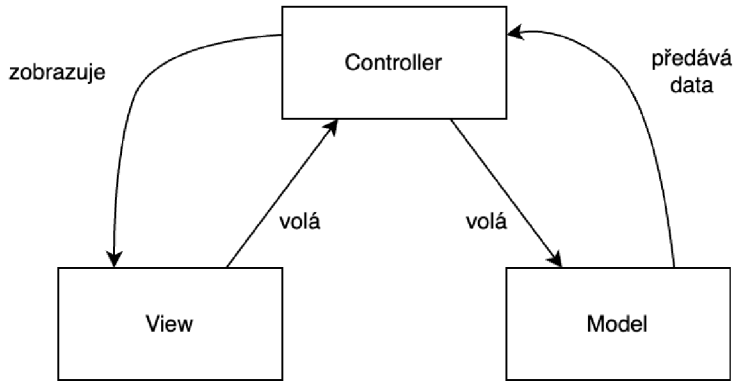
```
public class Token
{
    public Token(string json)
    {
        JObject jObject = JObject.Parse(json);
        var jRealmPermissions =jObject["realm_access"]?["roles"] as JArray;
        RealmPermissions = jRealmPermissions != null ?
            jRealmPermissions.Select(p => p.ToString()).ToArray() : new string[]{};
        var jClientPermissions = jObject["resource_access"]?["account"]?["roles"] as JArray;
        ClientPermissions = jClientPermissions != null ?
            jClientPermissions.Select(p => p.ToString()).ToArray() : new string[]{};
    }

    public string[] RealmPermissions { get; set; }
    public string[] ClientPermissions { get; set; }
}
```

Obrázek 2.4: Třída Token

Vývoj funkcí

Pro hlavní funkčnost aplikace je použit vzor MVC. Díky tomuto vzoru je kód organizovaný a lehce udržovatelný.



Obrázek 2.5: Model-View-Controller

2.6 VÝVOJ MOBILNÍ APLIKACE

Mobilní aplikace je vyvíjena na platformě .NET za použití technologie .NET MAUI. Cílem vývoje mobilní aplikace je demonstrovat možnost integrace různých typů aplikací s Keycloak. Aplikace je navržena tak, aby poskytovala pouze základní funkčnost. Tím se rozumí ověření uživatele a následně ověření jeho pravomocí. V případě dostatečných pravomocí je uživateli zobrazen seznam předmětů.

2.6.1 Specifikace požadavků

Hlavními požadavky pro mobilní aplikaci byly:

1. **Integrace s Keycloak**: Uživatel se přihlásí prostřednictvím Keycloak a aplikace získá jeho oprávnění.
2. **Zobrazení seznamu předmětů**: V případě dostatečných oprávnění je uživatel schopen zobrazit seznam předmětů.

2.6.2 Architektura aplikace

Architektura aplikace je založená na návrhovém vzoru MVVM (Model-View-ViewModel).

Modely

Modely jsou odpovědné za manipulaci s daty a za jejich ukládání.

Pohledy (Views)

Pohled představuje uživatelské rozhraní, pomocí kterého uživatel interaguje.

Prezentační model (ViewModel)

Tyto komponenty se starají o propojení uživatelského rozhraní s datovou logikou.

2.6.3 Implementace

Databáze

Pro mobilní aplikaci je použita stejná databáze, jako pro aplikaci webovou. Pro komunikaci s databází je rovněž použit Entity Framework a stejná třída **RepositoryManager**.

Integrace s Keycloak

Pro autentifikaci je použita knihovna IdentityModel. Tato knihovna obsahuje jednotlivé metody, jako jsou například LoginAsync(metoda sloužící k přihlášení uživatele) a IntrospectTokenAsync(metoda sloužící k získání informací o uživateli výměnou za přístupový token získaný po úspěšném přihlášení).

Pro samotné přihlášení je vytvořena třída **WebAuthenticatorBrowser**. Tato třída přijme požadavek na přihlášení a přesměruje uživatele na stránku v prohlížeči, kde se uživatel následně přihlásí. Po úspěšném přihlášení je uživatel přesměrován zpět do mobilní aplikace.

```
internal class WebAuthenticatorBrowser : IBrowser
    {
        public async Task<BrowserResult> InvokeAsync(BrowserOptions options, CancellationToken cancellationToken = default)
        {
            try
            {
                WebAuthenticatorResult authResult =
                    await WebAuthenticator.AuthenticateAsync(new Uri(options.StartUrl), new Uri(options.EndUrl));
                var authorizeResponse = ToRawIdentityUrl(options.EndUrl, authResult);

                return new BrowserResult
                {
                    Response = authorizeResponse
                };
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex);
                return new BrowserResult()
                {
                    ResultType = BrowserResultType.UnknownError,
                    Error = ex.ToString()
                };
            }
        }

        public string ToRawIdentityUrl(string redirectUrl, WebAuthenticatorResult result)
        {
            IEnumerable<string> parameters = result.Properties.Select(pair => $"{pair.Key}={pair.Value}");
            var values = string.Join("&", parameters);

            return $"{redirectUrl}#{values}";
        }
    }
```

Obrázek 2.6: Třída WebAuthenticatorBrowser

2.7 MOŽNOSTI BUDOUCÍHO VYLEPŠENÍ

Tato práce představuje základní demonstraci možností Keycloak s aplikacemi vyvíjenými na platformě .NET a proto je zde velký prostor pro vylepšení a rozšíření funkcionality.

Mezi případné vylepšení může patřit:

1. **Optimalizace aplikací**: Ačkoliv slouží webové i mobilní aplikace jako čistě demonstrační, je zde prostor pro optimalizaci a zlepšení výkonu.
2. **Vytvoření knihovny pro autorizaci s Keycloak**: Autorizační logika, která je implementovaná v aplikacích, by mohla být převedena do samostatné knihovny pro .NET. Toto by zjednodušilo použití Keycloak pro další aplikace na platformě .NET, jelikož by nebyla potřeba implementovat autorizační logiku do každé z nich zvlášť.
3. **Přidání dalších vrstev MFA**: V současné době je implementována pouze dvoufaktorová autentifikace, která zahrnuje klasické přihlašovací údaje a OTP. Keycloak nabízí více možností včetně biometrického ověření pomocí standardu WebAuthn.
4. **Větší využití Keycloak**: Keycloak nabízí širokou škálu velmi užitečných funkcí, které nebyly v této práci využity. Toto by mohlo zahrnovat např. pokročilejší nastavení politiky hesel, ověření účtu emailem a monitoring uživatelské aktivity.

3 ZÁVĚR

V průběhu této bakalářské práce bylo dosaženo několika klíčových cílů. Hlavním účelem bylo prostudování autentifikace, autorizace, existujících standardů a protokolů, systémů pro ověření identity a správu přístupů. Následoval výběr vhodného systému, který splňoval předem dané parametry. Jako vhodný systém se osvědčil Keycloak.

Následovalo praktické použití Keycloak pro ověření identity uživatelů aplikací vyvíjených na platformě .NET, za kterým nastoupilo ověření jejich pravomocí a tedy správa přístupu k určitým částem aplikace.

Během vývoje aplikací bylo nezbytné se naučit používat technologie platformy .NET, jako jsou ASP.NET a .NET MAUI. Za použití těchto technologií byly vyvinuty dvě aplikace.

Dalším nezbytným krokem bylo prostudování integrace Keycloak s .NET aplikacemi a následné praktické využití těchto znalostí pro integraci Keycloak do výše zmíněných aplikací.

Struktura vytvořených aplikací byla založena na návrhových vzorech MVC a MVVM. Tyto návrhové vzory přispívají k udržitelnosti a modularitě projektu, což přináší možnost jeho dalšího rozvoje.

Ačkoli byly úspěšně splněny hlavní cíle, zůstávají možnosti pro budoucí vylepšení a rozšíření, které by mohly zlepšit výkon, bezpečnost a funkcionalitu aplikací.

Ve finále lze prohlásit, že tato práce přinesla cenné zkušenosti a poznatky v oblasti správy identity a přístupu. Výsledkem je teoretický přehled v této oblasti a praktická implementace, která demonstruje možnosti Keycloak a jeho využití s platformou .NET. Proto tato práce může sloužit jako základ pro další projekty, které se budou zaobírat problematikou autorizace a autentifikace ať už na platformě .NET, nebo na kterékoliv jiné.

BIBLIOGRAFIE

[1] KIM, Hokeun a Edward A LEE. Authentication and Authorization for the Internet of Things. *IT Professional*. 2017, roč. 19, č. 5, s. 27–33.

[2] IDRUS, Syed Zulkarnain Syed et al. A review on authentication methods. *Australian Journal of Basic and Applied Sciences*. 2013, roč. 7, č. 5, s. 95–107.

[3] YUAN, E. a J. TONG. Attributed based access control (ABAC) for Web services. In: *IEEE International Conference on Web Services (ICWS'05)*. 2005, s. 569. Dostupné z DOI: 10.1109/ICWS.2005.25.

[4] JONES, Michael, John BRADLEY a Nat SAKIMURA. *Json web token (jwt)*. 2015. Tech. zpr.

[5] BOYD, Ryan. *Getting started with OAuth 2.0*. " O'Reilly Media, Inc.", 2012.

[6] SAKIMURA, Natsuhiko et al. Openid connect core 1.0. *The OpenID Foundation*. 2014, S3.

[7] EL HAJJ HUSSEIN, Mohammad. *Reproducible Examples for Integration with Keycloak*. 2019. Tech. zpr.

[8] HUGHES, John a Eve MALER. Security assertion markup language (saml) v2. 0 technical overview. *OASIS SSTC Working Draft sstc-saml--tech-overview-2.0-draft-08*. 2005, roč. 13, s. 12.

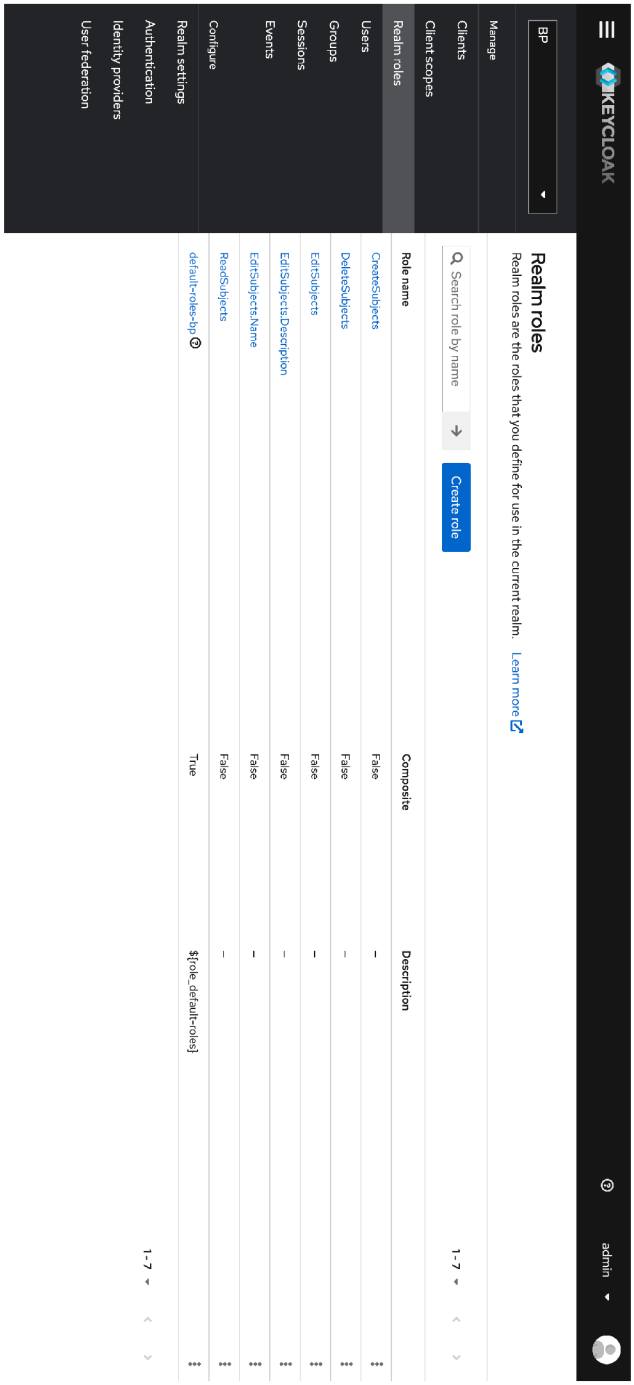
[9] HOWES, Tim, Mark SMITH a Gordon S GOOD. *Understanding and deploying LDAP directory services*. Addison-Wesley Professional, 2003.

[10] MOHAMMED, Ishaq Azhar. Systematic review of Identity Access Management in information security. *International Journal of Innovations in Engineering Research and Technology*. 2017, roč. 4, č. 7, s. 1–7.

[11] TEAM, Keycloak. *Documentation 21.1.1*. WildFly, [b.r.]. Dostupné také z: https://www.keycloak.org/documentation.

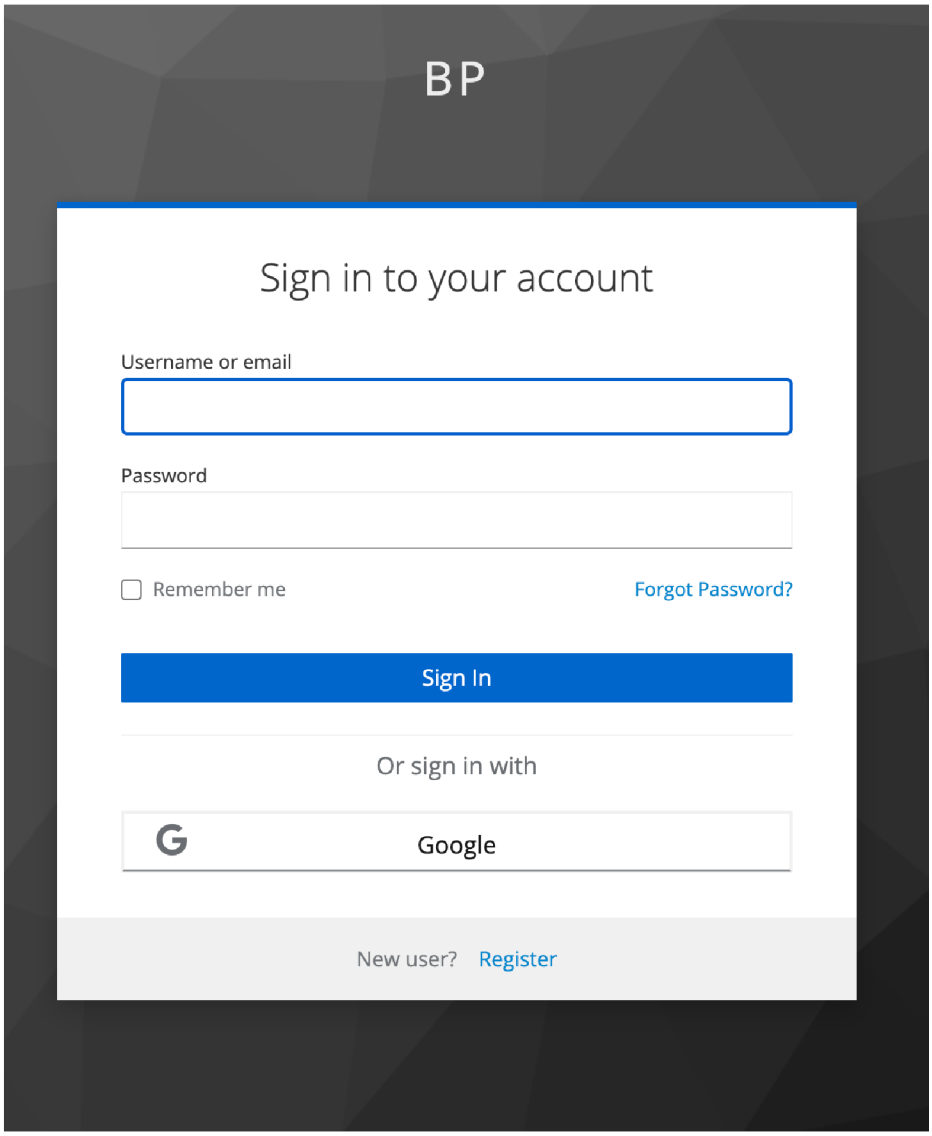
PŘÍLOHY

ADMINISTRÁTORSKÉ ROZHRANÍ KEYCLOAK



Obrázek 3.1: UI Administrátorského rozhraní Keycloak

ROZHRANÍ PRO PŘIHLÁŠENÍ UŽIVATELE



Obrázek 3.2: UI pro přihlášení

UI WEBOVÉ APLIKACE

Obrázek 3.3: UI webové aplikace