
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**
**DEPARTMENT OF INFORMATION SYSTEMS**


# NÁSTROJE PRO SPRÁVU IDENTIT V INFORMAČNÍCH SYSTÉMECH

**TOOLS FOR IDENTITY MANAGEMENT IN INFORMATION SYSTEMS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **FRANTIŠEK FÚDOR**
**AUTHOR**

**VEDOUCÍ PRÁCE** **doc. Ing. RADEK BURGET, Ph.D.**
**SUPERVISOR**



**BRNO 2024**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

# Zadání bakalářské práce

154633

| | |
|---|---|
| Ústav: | Ústav informačních systémů (UIFS) |
| Student: | **Fúdor František** |
| Program: | Informační technologie |
| Název: | **Nástroje pro správu identit v informačních systémech** |
| Kategorie: | Informační systémy |
| Akademický rok: | 2023/24 |

Zadání:

1. Seznamte se se standardy Open ID Connect a OAuth2 včetně možnosti Single Sign-On a přihlašování prostřednictvím externích služeb (Facebook, Google apod.)
2. Prostudujte existující open-source produkty pro autorizaci a správu identit v informačních systémech, jako např. MidPoint, OpenIAM a další.
3. Po konzultaci s vedoucím zvolte vhodnou aplikační doménu a navrhněte řešení centralizované správy identit a jednotného přihlášení s využitím vybraných open-source produktů.
4. Implementujte navržené řešení s oddělenou správou identit a autentizace. Vytvořte alespoň dvě jednoduché aplikace pro demonstraci funkčnosti řešení.
5. Otestujte vytvořené řešení v odpovídajícím nasazení.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Bertocci, V.: OAuth2 and OpenID Connect: The Professional Guide, Okta, Inc., 2022
- Dále dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:
Bez požadavků

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

| | |
|---|---|
| Vedoucí práce: | **Burget Radek, doc. Ing., Ph.D.** |
| Vedoucí ústavu: | Kolář Dušan, doc. Dr. Ing. |
| Datum zadání: | 1.11.2023 |
| Termín pro odevzdání: | 9.5.2024 |
| Datum schválení: | 30.10.2023 |

# Abstrakt

Integrácia nástroja midPoint s nástrojom KeyCloak pomocou OpenID Connect. Zabezpečenie jednotného prihlásenia, jednotného odhlásenia a kontroly relácie. Implementácia dvoch demonštračných Spring Boot aplikácií pomocou OIDC klienta. Prvá aplikácia demonštruje konzumáciu tokenu identity a volanie koncového bodu userinfo, demonštruje funkcionalitu jednotného prihlásenia a odhlásenia. Druhá aplikácia demonštruje autorizované volanie midPoint API a iniciuje žiadosť o dovolenku, ktorá vytvorí schvaľovací proces na manažéra daného užívateľa smerom do nástroja midPoint. Zabezpečenie centrálneho manažmentu identít.



# Abstract

Integration between midPoint and KeyCloak using OpenID Connect. Providing single sign-on, single logout and session control. Implementation of two Spring Boot applications using the OIDC client. The first application demonstrates ID token consumption and calling the userinfo endpoint and demonstrates single sign-on and sign logout functionality. The second application demonstrates an authorized call to the midPoint API endpoint which initiates a leave request that creates an approval process on a given user's manager towards the midPoint. This environment provides central identity management.



# Kľúčové slová

Jednotné prihlásenie, manažment identít, centralizovaný manažment, riadenie prístupov, riadenie prístupu založené na rolách, OpenID Connect, OAuth2.0, SAML, JWT, token, prístupový token, token identity, token obnovy, KeyCloak, midPoint, Spring Boot, autorizačný server, poskytovateľ identity, poskytovateľ služby, jednotné odhlásenie, cookies, HTTP, menný priestor, klient

# Keywords

Single Sign-on, Single Logout, Identity and Access Management, Centralized Management, Access Control, RBAC, OIDC, OAuth2.0, SAML, JWT, Token, Access Token, ID Token, Refresh Token, KeyCloak, midPoint, Spring Boot, Authorization Server, IdP, SP, Cookies, HTTP, Realm, Client



# Citácia

FÚDOR, František. *Nástroje pro správu identit v informačních systémech.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Radek Burget, Ph.D.

# Nástroje pro správu identit v informačních systémech

## Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána doc. Ing. Radeka Burgeta, Ph.D.. Uvideol som všetky literálne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
František Fúdor
6. mája 2024


## Poďakovanie

Chcel by som poďakovať vedúcemu práce, ktorý mi poskytol príjemné prostredie pre konzultácie s rýchlou časovou odozvou. Ďalej by som chcel poďakovať Petrovi Holešovi a Ľubomírovi Odlevákovi a celej mojej rodine za podporu počas priebehu štúdia.

# Obsah

# Zoznam obrázkov

# Zoznam tabuliek

# Kapitola 1

# Úvod

V dobe, kedy sa všetko točí okolo počítačov, mobilov a internetu je bežný smrtelník odkázaný na používanie obrovského množstva informačných systémov, webových aplikácií a stránok, do ktorých si musí registrovať enormné množstvo účtov a vytvárať nové heslá. Ale kto si to má všetko pamätať? Hlavne ak človek vykonáva kancelársku, sedavú prácu, v ktorej používa desať aplikácií, medzi ktorými sa celý deň preklikáva. Nedajbože si človek odbehne na toaletu a keď sa vráti k počítaču, zistí, že je odhlásený z každej aplikácie a teraz musí vypĺňať heslo do každej z nich. Ale čo ak by to tak nebolo?

Táto práca sa zameriava na metódy jednotného prihlásenia medzi veľkým množstvom aplikácií, ktoré zaručia, že človek sa do svojich aplikácií prihlási raz a to je asi tak všetko. Takéto pohodlné prihlasovanie je v dnešnej dobe viac než žiadúce. Práca popíše základné pojmy a metódy prihlasovania do informačných systémov, ktoré sa neskôr v práci nabalia na dnešné trendy, výmenu prihlasovacích údajov za malé a bezpečné kusy kódu, ktoré si aplikácie medzi sebou dokážu vymeniť bezpečne, na pozadí a veľmi rýchlo. Až tak, že si to človek ani nestihne všimnúť. Takouto bezpečnou výmenou dokážu aplikácie užívateľa prihlásiť bez nutnosti registrácie, či nadbytočného prihlasovania.

Cieľom práce je vytvoriť pohodlné prostredie s jednotným prihlásením pre spoločnosť s veľkým počtom zamestnancov a rozsiahlou organizačnou štruktúrou, ktoré poskytne zamestnancom príjemný prechod medzi aplikáciami spoločnosti a zároveň poskytne administrátorom detailnú kontrolu a prispôsobivosť prostredia pre špecifické biznisové potreby danej spoločnosti.

Pre zaručenie administrátorských potrieb prispôsobenia a rozšíriteľnosti navrhnutého prostredia budú naštudované a porovnané nástroje, ktoré spravujú identity užívateľov a ktoré sú voľne dostupné ako open-source riešenia. Neskôr budú vybrané dva nástroje, ktoré vo svojej spolupráci zaručia detailnú správu identít a vytvoria centralizovaný manažment, v ktorom bude mať užívateľ jedenu identitu v rámci celého prostredia, ktoré sa bude skladať z dvoch nástrojov a dvoch demonštračných aplikácií.

Nakoniec bude celé prostredie otestované pomocou demonštračných aplikácií a tým bude overená očakávaná funkcionalita celého prostredia.

# Kapitola 2

# Zhrnutie súčasného stavu riadenia prístupov

Táto kapitola vysvetľuje základné pojmy a metódy v autentifikácií, autorizácií a riadení prístupov identít v informačných systémoch. Zahŕňa zaužívané praktiky, ktoré sa dodnes aktívne používajú, a od ktorých sa neskôr v práci budú nabaľovať ďalšie, modernejšie riešenia v riadení prístupov identít.

## 2.1 Digitálna identita

Digitálna identita, ďalej len **identita**, je základným prvkom manažmentu identít a nástrojov, ktoré spravujú identity.

O presnej definícii identity sa vedú dlhé medzinárodné diskusie a nie je ju možné popísať bez kontextu. Identita v informatike môže byť reprezentovaná rôzne, najačastejšie v podobe užívateľa, organizácie, aplikácie, či iného počítača, alebo zariadenia. [20]

Identita, na ktorú sa táto práca zameriava je práve užívateľský účet, ktorý patrí skutočnej osobe z reálneho sveta.

Identita je popísana atribútmi, ktoré sú s ňou spojené, ako napríklad identifikátor, meno, alebo vlastnosti, napríklad roly, ktoré sa používajú v riadení prístupov. Tieto atribúty pomáhajú systémom pri rozhodovaní sa, či daná identita môže pristúpiť k prostriedku systému, alebo má tento prístup zamietnutý. Identita musí byť v rámci systému jednoznačne identifikovateľná, napríklad emailovou adresou, unikátnym menom, alebo iným unikátnym identifikátorom (vhodne zvoleným pre potreby systému). [8]

## 2.2 Autentifikácia

Autentifikácia je proces úspešného **overenia identity** užívateľa, zariadenia, alebo iného subjektu v informačných systémoch, často ako predpoklad povolenia prístupu k prostriedkom systému. Identita môže byť overená po úspešnom porovnaní zdieľaného tajomstva (napríklad hesla), ktoré bolo vopred dohodnuté, najčastejšie pri prvej registrácií užívateľa do systému. Hlavným cieľom autentifikácie je vykonávanie dôveryhodnej komunikácie medzi užívateľom a systémom. [31]

Inými slovami overuje, že osoba, ktorá sa snaží pristúpiť k prostriedkom systému je naozaj osoba z reálneho sveta, ktorej identita patrí.

### Autentifikačné procesy identifikujú tri faktory

Nasledujúci zoznam faktorov je prevzatý z článku od pána O'Gorman, L. [31]

- Čo užívateľ vie (napríklad **heslo**).
- Čo užívateľ vlastní (napríklad kľúč, alebo karta zamestnanca).
- Kto užívateľ je (napríklad odtlačok prsta, alebo hlas).

Nástroje pre správu identít v **komerčných** systémoch sa zameriavajú hlavne na prvý faktor. Hlavným dôvodom je miera prístupnosti[1]. Komerčné systémy nepotrebujú až tak vysokú ochranu pre bezpečný prístup k ich prostriedkom, pretože ich cieľovou skupinou sú bežné osoby z reálneho sveta, teda bežný používateľ internetu. Ak sa jedná o systém, ktorý si žiada vyššiu mieru ochrany, je dobrou praktikou implementovať autentifikačnú metódu, ktorá pracuje s viacerými, alebo všetkými faktormi.

### Autentifikačná metóda pomocou hesla

Táto metóda autentifikuje identitu užívateľa na základe faktoru **čo užívateľ vie**. Tajná informácia, ktorá je zdieľaná medzi informačným systémom a užívateľom je heslo. Heslo je od užívateľa získané pri prvej registrácií do systému. Takto získané heslo je vo forme hašu uložené na strane systému.

Autentifikácia pomocou hesla je najznámejší proces overenia identity. Takáto autentifikácia má mnoho problémov. Heslá sa musia pravidelne meniť, aby sa znížila šanca uhádnutia, tajného odpozorovania, či odpočutia. Užívateľ taktiež môže heslo zdieľať s inými osobami a tak zvýšiť pravdepodobnosť odcudzenia svojej identity. [36]

## 2.3 Autorizácia

Autorizácia je proces, v ktorom sa overuje, či identita snažiaca sa získať prístup k prostriedkom informačného systému je dostatočne **kvalifikovaná** a oprávnená vidieť a operovať s obsahom žiadaného prostriedku. Výsledkom úspešnej autorizácie je udelenie oprávnenia pre prístup k tomuto prostriedku. Spôsob, akým sa overí či je identita dostatočne kvalifikovaná upresňujú metódy *riadenia prístupu*, ktoré sú použité v informačnom systéme. [6] [38]

## 2.4 Autentifikácia vs autorizácia

Medzi autorizáciou a autentifikáciou je podstatný rozdieľ, aj napriek tomu, že znejú podobne.

**Autentifikácia** v jednoduchosti overuje **totožnosť** identity, ktorá sa snaží pristúpiť do informačného systému. Teda zisťuje či identita patrí osobe, ktorá sa snaží o prístup. [11]

**Autorizácia overuje**, či je táto, už prihlásená identita, dostatočne **kvalifikovaná** pre prístup k prostriedku v informačnom systéme. Ak identita spĺňa kritériá špecifikované v použitej metóde riadenia prístupu, autorizácia prebehne úspešne a identita dostane oprávnenie pre prístup k prostriedku. [11]

Pre zaručenie bezpečnosti v informačných systémoch **sa používajú obidva procesy**.

[1] aké komplikované je dostať sa k priestriedkom systému

## 2.5 Riadenie prístupu

Účelom riadenia prístupu je **obmedziť činnosti** a operácie nad prostriedkami informačného systému, ktoré môžu identity vykonávať pri jeho používaní. Taktiež obmedzuje činnosti a operácie aplikáciám, ktoré pristupujú k prostriedkom systému v mene niektorej z registrovaných identít. Riadenie prístupu sa spomínaným obmedzovaním snaží zabrániť činnostiam, ktoré by mohli viesť k narušeniu bezpečnosti systému, úniku informácií, či delegácií oprávnení neoprávneným identitám. [37] [4]

Riadenie prístupu je základným prvkom bezpečnosti v systémoch a aktívne používa metódy autentifikácie a autorizácie. Po úspešnom overení totožnosti identity pomocou autentifikácie, nasleduje autorizácia oprávnení už autentifikovanej identity, pre kontrolovaný prístup len k tým prostriedkom, s ktorými má identita povolené pracovať.

Kontrolované riadenie prístupu **chráni** dôverné informácie, údaje o zákazníkoch a duševné vlastníctvo pred odcudzením. Namiesto manuálneho riadenia prístupu a spravovania oprávnení sa väčšina organizácií, ktoré potrebujú chrániť svoje systémy, pri implementácií riadenia prístupu aktívne opiera práve o manažment identít a ich prístupov a existujúce nástroje pre správu identít. [4]

### Riadenie prístupu založené na rolách (RBAC)

Riadenie prístupu založené na rolách, z anglického slovného spojenia *Role-Based Access Control*, ďalej len RBAC, je politika riadenia prístupu, ktorá využíva existenciu rolí v informačných systémoch. Rolu možno definovať ako **kolekciu oprávnení** spojených s konkrétnymi pracovnými činnosťami, alebo zodpovednosťami nejakej skupiny identít v systéme. Namiesto prideľovania **množstva** oprávnení každej identite **zvlášť**, sú tieto oprávnenia špecifikované pre objekt roly. Neskôr v informačnom systéme môže byť takýto objekt roly pridelený identite, či skupine identít a tým zabezpečená propagácia veľkého množstva oprávnení viacerím identitám súčasne. [19] [37]

Predpokladom funkčného využitia takéhoto riadenia prístupu je nutnosť identifikácie a špecifikácie konkrétnych rolí, ktoré môžu v informačnom systéme vystupovať a teda abstraktné rozdelenie identít do logických skupín na základe ich náplne práce, činností, oprávnení či obmedzení. Najčastejšie je takéto riadenie prístupu implementované v informačných systémoch pre personálne oddelenia v organizáciách, ktoré sebe, alebo inej organizácií spravujú bezpečnostné procesy nad zamestnancami, externistami či dodávateľmi.

RBAC je v súčasnosti najpopulárnejší model riadenia prístupu, ktorý sa používa ako alternatíva k tradičnému diskrétnemu riadeniu prístupu a povinnému riadeniu prístupu v informačných systémoch, ktoré spravujú bezpečnosť organizácie. Najdôležitejšou vlastnosťou RBAC je snaha špecifikovať a nasledovať bezpečnostnú politiku, ktorá je unikátna pre danú organizáciu. Ideálne takým spôsobom, aby sa prirodzene prispôsobila organizačnej štruktúre. [27]

RBAC vďaka jeho vlastnostiam podporuje **tri zaužívané bezpečnostné princípy** [27]:

1. **Princíp najmenej oprávnení**
2. Oddelenie povinností
3. Abstrakcia dát

### Princíp najmenej oprávnení

Princíp najmenej oprávnení je najdôležitejší princíp pri návrhu bezpečnostnej politiky v informačných systémoch. Tento princíp kladie veľký dôraz na to, aby užívateľ (vždy keď je to možné) dostal ten najmenší počet oprávnení, ktoré nutne potrebuje na vykonávanie činností, alebo pre prístup k prostriedkom systému. Nikdy nie viac. [27]

# Kapitola 3

# Zhrnutie súčasného stavu manažmentu identít

Táto kapitola sa zameriava na manažment identít a prístupov. Popisuje zaužívané pojmy, modely a architektúry, ktoré sa využívajú pre úspešnú správu identít v komerčných, alebo súkromných prostrediach. Popisuje životný cyklus identity, federáciu identity naprieč viacerými doménami a základné vlastnosti jednotného prihlásenia naprieč viacerými aplikáciami.

## 3.1 Manažment identít a ich prístupov (IAM)

Manažment identít a ich prístupov vychádza z anglického slovného spojenia *Identity and Access Management* (IAM) a jeho hlavnou úlohou je udržanie integrity dát identít počas ich životného cyklu, s cieľom poskytovania týchto dát ďalším službám bezpečným spôsobom a taktiež s cieľom ochrániť súkromie každej identity. [9]

V technickom ponímaní je manažment identít spôsob, akým manažovať užívateľa, jeho identity a prístupy k prostriedkom v informačných systémoch, alebo internetových službách. V rámci organizácie môže byť manažment identít ako jeden samostatný produkt, alebo to môže byť kombinácia procesov, produktov, cloudových služieb, ktoré poskytujú administrátorom kontrolu nad dátami a prostriedkami informačných systémov, ku ktorým môžu jednotlivé identity pristúpiť. [7]

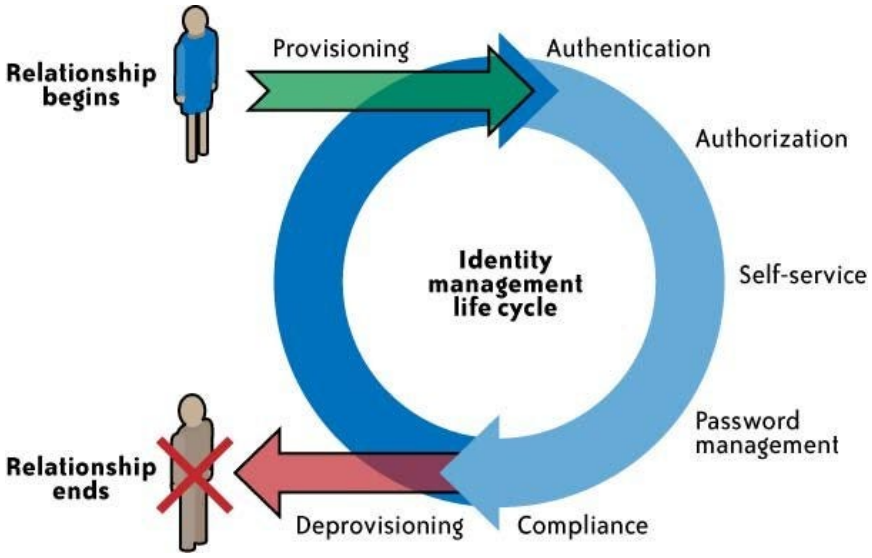
Manažment identít a ich prístupov sa rozdeľuje na dva komponenty. Na **manažment identít** a **manažment prístupov**. Manažment identít sa zaoberá poskytovaním identít, teda automatizovanými procesmi nad identitami užívateľov a ich autentifikáciou. Manažment prístupov riadi autorizáciu, spravuje oprávnenia, manažuje bezpečnostnú politiku a špecifikuje, akým spôsobom sa informácie o identitách propagujú ďalším aplikáciám a službám s cieľom ochrániť súkromie užívateľov. [39]

### Poskytovanie identity (Identity Provisioning)

Je proces vytvárania, manažovania, ukončovania digitálnej identity v informačných systémoch. [3]

Jedná sa o službu, ktorá poskytuje automatizované procesy nad životnými cyklami identít so snahou znížiť časovú náročnosť a cenu. Najlepším príkladom je opäť personálne oddelenie organizácie. Personálne oddelenie musí riešiť vytvorenie príslušných identít pri nástupe nových zamestnancov do organizácie. Je potreba riešiť vytvorenie a pridelenie prístupov do všetkých informačných systémov a aplikácií, s ktorými bude zamestnanec

pracovať. Ak zamestnanec vypadne z pracovnej činnosti, napríklad počas dovolenky, alebo práce neschopnosti, môžu mu byť cielene odobrané prístupy do niektorých aplikácií, na základe bezpečnostnej politiky organizácie. Zamestnanec potrebuje byť upozornený o zmenách v organizácií, napríklad pri zmene organizačnej štruktúry, či zmene jeho pracovnej pozície. Keď zamestnanec odchádza z organizácie musia byť jeho identity adekvátne odstránené zo systému personálneho oddelenia a zablokované, či odstránené prístupy k aplikáciám a informačným systémom organizácie. Všetky tieto procesy sa *Identity Provisioning* snaží riešiť automatizovane a kontrolovane. [42]




Obr. 3.1: Životný cyklus identity, prebratý z [42]

Životný cyklus identity podľa obrázku 3.1 obsahuje:

- **Relationship begins**: Vznik identity v systéme.
- **Provisioning**: Zabezpečenie procesov a služieb.
- **Authentication**: Autentifikácia identity.
- **Authorization**: Autorizácia identity.
- **Self-service**: Samoobsluhovacia služba (napríklad požiadanie o prístupy).
- **Password management**: Správa prihlasovacích údajov.
- **Compliance**: Dodržiavanie ďalších systémovo-špecifických predpisov.
- **Deprovisioning**: Ukončenie poskytovania procesov a služieb.
- **Relationship ends**: Zánik identity v systéme.

## 3.2 Poskytovatelia služieb (SP)

Poskytovateľ služieb je dodávateľ, ktorý poskytuje IT služby koncovým užívateľom a organizáciám. Tento široký pojem zahŕňa všetky podniky v IT, ktoré poskytujú produkty

a riešenia prostredníctvom služieb. Takéto služby môžu byť poskytované na zakázku, prostredníctvom platby za využívanie, alebo inou dohodou medzi dodávateľom a konzumentom. Poskytovateľ služieb zvyčajne nevyžaduje, aby si koncový užívateľ, alebo organizácia daný produkt zakúpili jednorázovo. Väčšina takýchto dodávateľov sa v dnešnej dobe zamierava na vytváranie, poskytovanie a spravovanie svojich produktov, ktoré ďalej poskytujú ako balík služieb pre koncových užívateľov a organizácie prostredníctvom mesačného, či ročného predplatenia. [34]

## 3.3 Poskytovatelia identít (IdP)

Poskytovateľ identít je špecifický dodávateľ, ktorý poskytuje službu pre úchovávanie a správu digitálnych identít užívateľov. Mnoho organizácií využíva túto službu s cieľom umožniť svojim zamestnancom pristupovať k prostriedkom informačných systémov, alebo aplikáciám v inej doméne bez nutnosti zakladania nového užívateľského účtu. Posyktovateľ identít môže autentifikovať identity užívateľov pomocou prihlasovacích údajov, alebo aj iných faktorov spomenutých v sekcií 2.2. Alebo môže ďalej poskytovať zoznam identít užívateľa pre iného poskytovateľa služieb, ktorý s ním ďalej pracuje (napríklad Single Sign-on popísaný v sekcií 3.6). [5]

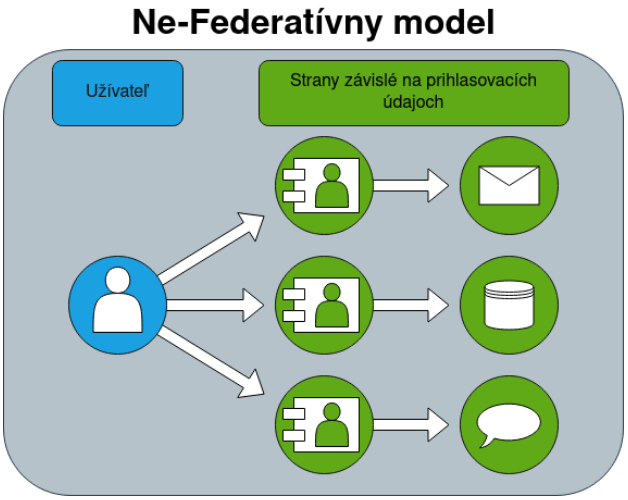
Poskytovateľ identít nie je limitovaný len autentifikáciou osôb z reálneho sveta. IdP dokáže autentifikovať akúkoľvek entitu pripojenú k sieti, vrátané aplikácie, počítača, alebo iného zariadenia. Najčastejšie sa IdP používa v cloudových riešeniach, ako autorita, ktorá manažuje identity užívateľa (skutočnej osoby). [5]

## 3.4 Federatívne systémy identít (FIS)

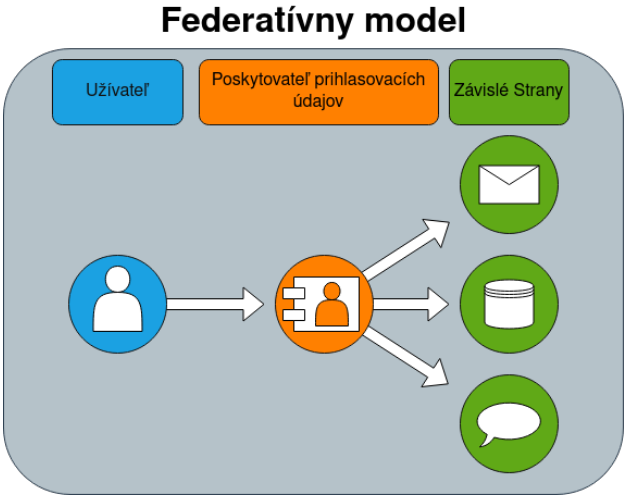
Hlavným cieľom federatívneho systému identít je poskytnúť užívateľom pohodlnejší zážitok pri používaní veľkého množstva služieb medzi viacerými SP. Federatívny systém identít adresuje problém veľkého množstva autentifikačných a registračných procesov, ktorými si užívateľ musí prejsť pri vstupe do každej novej/ďalšej domény. Federatívna identita je výsledkom kolaboratívnej a vzájomne závislej správy identifikačných údajov o identitách naprieč organizáciami, ktoré vo svojej spolupráci tvoria a spravujú federatívny systém identít. [29]

Federácia v kontexte manažmentu identít je biznisový model, v ktorom skupina dvoch, alebo viacerých organizácií tvoria dôveryhodné partnerstvo. Takéto partnerstvo je legálne viazané biznisovou, alebo technickou zmluvou. Vďaka zmluvnému partnerstvu je užívateľom v takto spolupracujúcich organizáciách umožnené pristupovať k prostriedkom a službám partnerských organizácií plynule a bezpečne, aj napriek tomu, že priamo nepatria do ich domény identít. [18]

Užívatelia z jednej domény môžu bezpečne pristúpiť k prostriedkom inej domény bez nutnosti registrácie novej identity. Vďaka tomu, že federatívny systém identít abstraktne agreguje identity užívateľa pod jednu federatívnu identitu, zaniká tým požiadavok na existenciu užívateľského účtu v novej doméne a užívateľ, namiesto registrácie, dokáže pristúpiť k prostriedkom a službám novej domény pomocou autentifikačného procesu na strane jeho vlastnej domény, teda u svojeho poskytovateľa identít. Ten bezpečne a efektívne predá identifikačné údaje o identite užívateľa do novej domény, ktorá je v tomto prípade v role posyktovateľa služby. [29]




Obr. 3.2: Ne-Federatívny model, prebratý z [29]




Obr. 3.3: Federatívny model, prebratý z [29]

Federatívny systém identít sa skladá z jedného, alebo viac poskytovateľov identít a jedného, alebo viac poskytovateľov služieb, takým spôsobom, že užívateľ dokáže byť autentifikovaný do konkrétnej služby od SP cez autentifikáciu na strane IdP. Autentifikačný proces sa skladá z užívateľ-ku-IdP časti, po ktorej nasleduje časť IdP-ku-SP potvrdenie identity, kedy poskytovateľ identít ubezpečí poskytovateľa služby, že indetita užívateľa bola úspešne autentifikovaná. [13]

Vo federatívnom systéme identít existuje niekoľko konkrétne špecifikovaných protokolov, ako napríklad Mobile Connect, alebo Shibboleth, ktorých implementácia vychádza z protokolov s otvorenou špecifikáciou ako je SAML 2.0, OpenID 2.0, OAuth 2.0, alebo OpenID Connect [13]. Jednotlivé protokoly popisujú spôsob výmeny informácie o identite medzi IdP, SP a koncovým užívateľom, spôsob kontroly relácie, výmeny prístupových žetónov (tokenov) a ich ukladanie do úložiska cookies, a ďalších dôležitých procesov. Táto práca sa zameriava práve na protokoly s otvorenou špecifikáciou a budú hlbšie popísané v jednotlivých sekciách.

## 3.5 Centralizovaný manažment identít

Centralizovaný manažment identít predstavujú systémy, v ktorých sú informácie o identite zjednocované, ukladané a manažované pomocou jednej centralizovanej autority, alebo inštitúcie. V tomto modeli sa centralizovaná autorita správa ako „ústredný orgán“, ktorý spravuje identifikačné informácie identít. Takýto manažment je prevažne využívaný organizáciami, finančnými inštitúciami a poskytovateľmi veľkého množstva služieb. [24] [2]

Príkladom centralizovaného manažmentu identít sú federatívne systémy identít spomenuté v sekcií 3.4, ktoré zjednocujú identitu užívateľa naprieč všetkými doménami a aplikáciami, ktoré navzájom spolupracujú v rámci federácie. Tieto identity sú kontrolované z takzvanej centrály, ktorá predstavuje jeden bod, teda jedného poskytovateľa identít. [1]

Opakom centralizované manažmentu je decentralizovaný manažment identít, v ktorom sú autentifikačné procesy a prístupy distrubuované naprieč viacerými doménami samostatne. Užívateľ je v takomto systéme nútený registrovať svoju identitu do každej novej/ďalšej domény osobitne s novými prihlasovacími údajmi. Takýto systém poskytuje užívateľom väčšiu kontrolu nad ich identitami a eliminuje potrebu centralizovanej autority. Príkladom takéhoto systému je blockchain, ktorého popularita vďaka kryptografickej mene Bitcoin rapídne stúpa. [24] [2] [40]

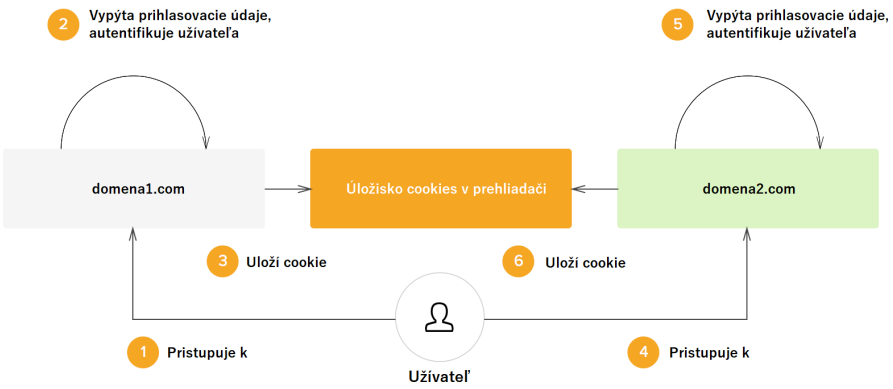
Táto práca sa zamierava na centralizovaný manažment identít a prístupov.

## 3.6 Single Sign-on (SSO)

Single Sign-on (SSO), alebo aj metóda jednotného prihlásenia je súhrnný pojem pre schémy, ktoré umožňujú užívateľom spoliehať sa na jediný autentifikačný proces, po ktorom bude užívateľ automaticky prihlásený a zapamätaný v každej aplikácií v rámci federácie. [13]

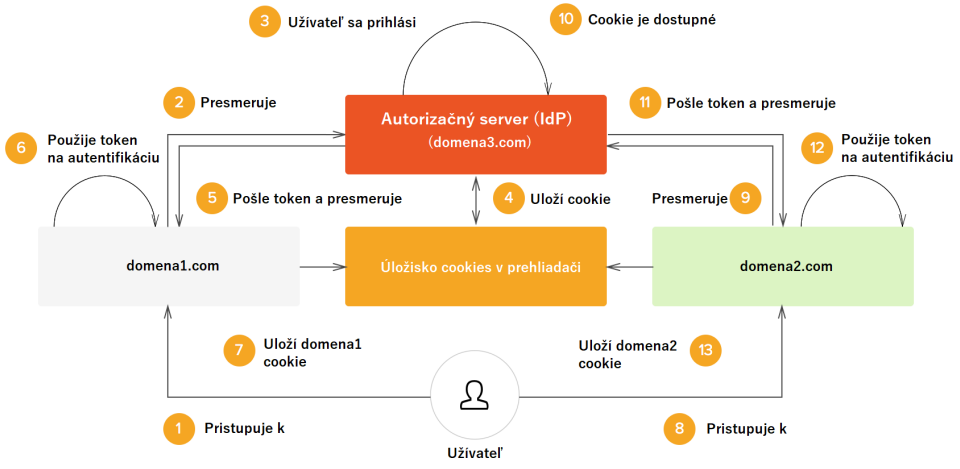
Tieto schémy, alebo aj metódy jednotného prihlásenia, fungujú na princípe centralizovaného manažmentu identít. SSO v cloudových riešeniach a webových applikáciách využívajú ďalšie technológie a metódy, ako napríklad *cookies*, kontrolu relácie, anglicky *session control*, alebo žetóny, anglicky *tokens*, ktoré ďalej zabezpečujú indikáciu, že užívateľ už v minulosti bol autentifikovaný a nie je nutné tento proces opakovať. Najznámejším príkladom aplikovaného SSO v praxi, pre bežného užívateľa internetu, je autentifikačný proces pomocou sociálnej siete ako je Facebook, Google, alebo LinkedIn, ktoré poskytujú dôvernú metódu jednotného prihlásenia [32]. Ak sa užívateľ prihlási do svojho účtu, napríklad od spoločnosti Google a neskôr pristúpi k nejakému produktu od tejto spoločnosti, ako napríklad Gmail, YouTube, Meet a tak podobne, bude automaticky autentifikovaný práve vďaka SSO.

V dobe kedy sa väčšina sociálnych interkacií vykonáva práve pomocou spomenutých sociálnych sietí je obrovská pravdepodobnosť, že bežný užívateľ internetu je zaregistrovaný v aspoň jednej z týchto spoločností. Vďaka popularite a obrovským úložným kapacitám môžu tieto spoločnosti poskytovať informácie o identitách a teda fungovať ako dôverný poskytovateľ identít (IdP) pre veľké množstvo voľne dostupných, komerčných webových aplikácií, teda pre poskytovateľov služieb (SP).




Obr. 3.4: Scenár autentifikácie do dvoch nezávislých domén bez využitia SSO, prebratý z [33]

Na obrázku 3.4 je ukázaný proces autentifikácie do dvoch domén, ktoré predstavujú dvoch poskytovateľov služby vo forme webovej aplikácie a jedno úložisko cookies, ktoré sa nachádza v použitom webovom prehliadači lokálne u užívateľa. [33]




Obr. 3.5: Typický scenár použitia SSO v dvoch spolupracujúcich doménach, prebratý z [33]

Na obrázku 3.5 je ukázaný proces autentifikácie do dvoch domén, ktoré predstavujú spolupracujúcich poskytovateľov služby vo forme webovej aplikácie a opäť jedno úložisko cookies u užívateľa. Ďalej pribudol autorizačný server, ktorý predstavuje poskytovateľa

identity. V bode sedem sa uloží token do úložiska cookies po prvej úspešnej autentifikácií užívateľa a v bode desať je tento token z úložiska cookies prečítaný a overený. Po úspešnom overení je užívateľ presmerovaný naspäť k webovej aplikácií a vďaka rýchlej výmene tokenu na pozadí je užívateľ úspešne autentifikovaný bez povšimnutia. [33]

### Cookies

Cookie je malé množstvo informácií, ktoré ukladá webový server, na ktorý užívateľ pristupuje, do lokálneho úložiska cookies vo webovom prehliadači užívateľa. Typicky je cookie vytvorené pri prvom prístupe k webovej stránke a pri ďalších prístupoch k danému webovému serveru je toto cookie čítané, poprípade upravené a znovu uložené na strane užívateľa. Hlavným zámerom ukladania cookies je uloženie stavu preferovaných nastavení webovej stránky, alebo aplikácie pre pohodlnejšie prehliadavanie a používanie. [21]

Komunikácia medzi webovým prehliadačom a webovým serverom, ktorý poskytuje webovú aplikáciu prebieha pomocou protokolu HTTP [30]. Samotné HTTP je bezstavové. Pokiaľ webový server žiadnym spôsobom neudržiava stav a koreláciu užívateľov, budú užívatelia pre tento webový server neznámi. Preto pre udržiavanie stavu a asociácie užívateľov používajú webové servery HTTP cookies. Webový server vytvára cookies, ktoré obsahujú stav a informáciu o užívateľovi a ukladajú ich do webového prehliadača na strane užívateľa, z ktorého bol vytvorený požiadavok. Webový server potom použije tieto cookies pre autentifikáciu HTTP požiadaviek od toho istého užívateľa a tým udržiava stabilný užívateľský stav. Predvolené parametre HTTP cookie sú: meno, hodnota, dátum exspirácie, cesta URL k webového serveru, pre ktorý je cookie validný, meno domény a príznak, či bol cookie poslaný pomocou SSL protokolu. [41]

Zabezpečené HTTP cookies používajú rôzne kryptografické metódy, ako napríklad enkrypciu, kód na overenie správ (MAC), message-digest (rodina hašovacích funkcií) a digitálny podpis. Enkrypcia je používaná na ochranu dát. Transformuje čistý text (reťazec znakov) pomocou šifrovacieho algoritmu na nečítateľnú šifru, ktorú dokáže prečítať len to zariadenie, ktoré vlastní tajný kľúč. Kód na overenie správ je krátka informácia, ktorá slúži na overenie autenticity správy. Algoritmus z rodiny message-digest berie ľubovoľne dlhú dĺžku vstupu reťazca znakov a vracia fixne danú dĺžku (v bitoch) reťazca znakov a vytvorí tým kryptografický haš. Tým je zabezpečené, že ak je s dátami cookie nejakým spôsobom počas HTTP komunikácie manipulované, vznikne pri výpočte hašu medzi prijímateľom a odosielateľom nezhoda. Digitálny podpis je matematický koncept založený na asymetrickej kryptografií a používa sa na overenie autenticity digitálnej správy, alebo dokumentu. Validný digitálny podpis dáva prijímateľovi dôveru, že správa bola vytvorená známym odosielateľom a nebolo s ňou počas komunikácie manipulované. [41]

Špecifické cookies, ako napríklad spomenuté HTTP cookies sú používané v autentifkácií založenej na cookies pre udržiavanie relácií užívateľov. Autentifikácia založená na cookies funguje na nasledujúcom princípe (prevzaté z [22]):

1. Užívateľ zadá prihlasovacie údaje pri bežnej autentifikácií pomocou hesla. Prehliadač pošle prihlasovací požiadavok na webový server pri odoslaní prihlasovacieho formulára.

2. Webový server autentifikuje užívateľa na základe použitých prihlasovacích údajov. Ak je autentifikácia úspešná, webový server vytvorí nasledujúce:

   - Reláciu na základe informácií užívateľa
   - Unikátny identifikátor, známy ako identifikátor relácie (session ID)

Webový server prepošle identifikátor relácie do webového prehliadača užívateľa, ktorý si ho uloží. Webový server taktiež udržiava informáciu o aktívnych reláciách.

3. Webový prehliadač pridáva identifikátor relácie v rámci každého požiadavku, ktorý odosiela webovému serveru a každý krát, čo webový server dostane správu s požiadavkou, tento identifikátor relácie overí. Identifkátor relácie pomáha pri autentifikačnom procese identifikovať užívateľa a adekvátne poskytuje prístup.

4. Relácia je na oboch stranách ukončená pri odhlásení užívateľa. Ukončenou reláciou sa ukončí aj proces overovania prostredníctvom identifikátora relácie.

Autentifikácia založená na cookies má mnohé nevýhody. Takáto autentifikácia nepodporuje prekrytie domén. Cookies dokáže byť posielané medzi doménami a poddoménami, ktoré patria jednej hlavnej doméne. Cookies nepracujú správne s mobilnými aplikáciami. Taktiež treba brať do úvahy limity cookies. Napríklad veľkosť informácie jedného cookie nemôže presiahnuť 4KB. Do cookies sa môžu ukladať informácie, ktoré môžu narúšať súkromie užívateľa. S narastajúcim počtom užívateľov narastá aj počet zápisov a čítaní do/z databázy a môžu spôsobovať oneskorenie pri využití maximálnej veľkosti cookie. [22]

Pre elimináciu väčšiny problémov, ktoré autentifikácia založená na cookies prínaša, boli vytvorené frameworky, ktoré implementujú tokeny a tým podporujú autentifikáciu založenú na tokenoch, alebo inak nazývaná aj autentifikácia bez-cookies. Najčastejšie sa využíva kombinácia autentifikácie založenej na tokenoch a komplementárne k tomu autentifikácia založená na cookies, ktoré dokopy užívateľom poskytujú autentifikáciu bez povšimnutia.

## 3.7 Autentifikácia založená na tokenoch

Tokeny sú systémom generované konštrukcie kódu, ktoré potvrdzujú overenie identity toho, kto sa snaží o prístup. Takáto autentifikácia stelesňuje výmenu prihlasovacích údajov užívateľa za serverovo-generovaný token. [17]

Autentifikáciu založenú na tokenoch je možné popísať aj ako protokol, ktorý umožňuje overiť identitu užívateľa (najčastejšie heslom), ktorý naspäť získa unikátny token. Počas doby platnosti tokenu je užívateľovi umožnené pristupovať k webovej stránke, alebo aplikácií, či systému, pre ktorú/ý bol tento token vystavený, takým spôsobom, že namiesto toho, aby sa užívateľ autentifikoval pomocou hesla každý krát pri vstupe, je použitý vygenerovaný token pre automatickú autentifikáciu. Z toho vyplýva, že užívateľ môže pristupovať k prostriedkom systému pokiaľ je token validný. Validita, alebo doba platnosti tokenu sa môže líšiť od systémovej implementácie. Doba platnosti tokenu môže byť určená fixne v časových jednotkách, alebo môže byť platnosť tokenu ukončená keď užívateľ opustí systém. [12]

Vďaka tomu, že token dokáže substituovať prihlasovacie údaje užívateľa, ktoré sa opätovne pri autentifikácií od užívateľa nevyžadujú, ale miesto toho sa použije token, je táto metóda autentifikácie považovaná za formu bezheslovej autentifikácie, pretože znemožňuje odposluch, a tým prináša ďalšiu vrstvu ochrany [28]. Táto metóda ďalej poskytuje administrátorom detailnú kontrolu nad akciami a transakciami, pretože implementácia tokenu je špecificky programovaná pre potreby riadenia prístupu daného systému [12].

Autentifikácia založená na tokenoch pozostáva zo štyroch krokov (prevzaté z [12]):

1. **Požiadanie**: Užívateľ požiada o prístup k chránenému prostriedku. Systém môže vyžadovať prihlásenie pomocou hesla, alebo inú autentifikačnú metódu.

2. **Overenie**: Server rozhodne, či užívateľ má mať prístup k prostriedku. Systém môže rozhodnúť na základe úspechu autentifikácie z predošlého kroku.

3. **Token**: Server naviaže komunikáciu a na základe úspešného overenia z predošlého kroku vystaví užívateľovi token.

4. **Uloženie**: Token je následne uložený do úložiska cookies vo webovom prehliadači na strane užívateľa.

Takmer všetky tokeny používané v manažmente identít a ich SSO implementácií sú obdoby JSON Web Tokenu (JWT), slovensky „džot token“.

# Kapitola 4

# Protokoly a štruktúry v manažmente identít a prístupov

Táto sekcia je venovaná hlbšej špecifikácií protokolov a štruktúr tokenov, ktoré sa aktívne používajú v manažmente identít a prístupov pre bezpečnú výmenu prihlasovacích údajov užívateľov medzi veľkým množstvom aplikácií a domén, ktoré zabezpečujú federáciu identity a jednotné prihlásenie.

### JSON Web Token (JWT)

Táto podsekcia je prevzatá z [25].

JWT je kompaktný, bezpečný prostriedok na reprezentáciu tvrdení, anglicky *claims*, ktoré sa majú premiestniť medzi dvoma stranami. Tvrdenia v JWT sú zakódované ako objekt JSON, ktorý sa používa ako užitočné zaťaženie, anglicky *payload*, štruktúry JSON Web Signature (JWS), alebo ako čistý text štruktúry JSON Web Encryption (JWE), čo umožňuje tieto tvrdenia digitálne podpísať, alebo ochrániť ich integritu pomocou kódu na overenie správy (MAC) a/alebo šifrovať. Takýto token môže obsahovať nula, alebo viac tvrdení.

Tvrdenie je malý kus informácie, ktorý popisuje dôkaz o subjekte. Je reprezentovaný ako pár meno/hodnota. Meno je vždy reprezentované ako reťazec znakov (string). Hodnota môže byť hociaká JSON hodnota (reťazec znakov, číslo. . . ).

Obsah hlavičky, nazývaný *JOSE Header* popisuje kryptografické operácie, ktoré boli aplikované na set tvrdení. Ak je hlavička určená pre JWS, token bude reprezentovaný štruktúrou JWS a jeho tvrdenia budú digitálne podpísané, alebo overené pomocou kódu na overenie správy, takým spôsobom, že samotné tvrdenia sa budú nachádzať v užitočnom zaťažení JWS. Ak je hlavička určená pre JWE, token bude reprezentovaný štruktúrou JWE a jeho tvrdenia búdu zašifrované ako čistý text.

Samotný JWT dokáže byť uzavretý do jednej zo spomenutých štruktúr a tým sa vytvorí vnorený JWT, ktorý umožňuje vnorené digitálne podpisovanie, alebo šifrovanie.

JWT je reprezentovaný ako sekvencia bezpečných URL častí oddelených bodkou. Každá časť obsahuje base64url-encoded hodnotu. Počet častí je závislí na použitej štruktúre. Pre JWS sa použije JWS serializácia a pre JWE sa použije JWE serializácia. JWT používaný pre SSO typicky obsahuje tieto tri časti:

1. **Hlavička**: Dáta o type tokenu a použitého algoritmu pri vytvorení tokenu
2. **Užitočné zaťaženie**: Tvrdenia

3. **Podpis**: Overená hlavička a užitočné zaťaženie pomocou použitého algoritmu MAC

Nasledujúci príklad hlavičky znázorňuje, že zakódovaný objekt je JWT a že tento JWT je v JWS štruktúre, ktorý je overený kódom na overenie správy s použitým algoritmom HMAC SHA-256:

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

Base64url-encoded hodnota pre túto hlavičku by bola:

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
```

Nasledujúci príklad znázorňuje užitočné zaťaženie, v ktorom sa nachádzajú tvrdenia o subjekte:

```
{"iss":"joe",
 "exp":1300819380,
 "http://example.com/is_root":true}
```

Base64url-encoded hodnota pre tieto tvrdenia by bola (pre lepší nákres bol použitý oddeľovač riadku):

```
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsD
QogImh0dHA6Ly9leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ
```

Vypočítaný digitálny podpis z predchádzajúcej hlavičky a užitočného zaťaženia v base64url-encoded hodnote pomocou použitého algoritmu HMAC SHA-256 by bol:

```
dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

Keď sa tieto tri časti spoja a oddelia sa bodkou vznikne kompletný JWT (pre lepší nákres boli použité oddeľovače riadku):

```
eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9
.
eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsD
QogImh0dHA6Ly9leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ
.
dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

Hlavička JWT typicky obsahuje pred-registrované mená tvrdení:

- **„typ“**: Špecifikuje typ JWT. Pokiaľ aplikácia nepoužíva viac typov pre odlíšenie implementácie tak sa toto tvrdenie nepoužíva.
- **„cty“**: Špecifikuje typ obsahu JWT. Pokiaľ aplikácia nepoužíva vnorené JWT tak sa toto tvrdenie nepoužíva.

- **„alg“**: Špecifikuje použitý algoritmus MAC.

Užitočné zaťaženie typicky obsahuje pred-registrované mená tvrdení:

- **„iss“**: Identifikuje entitu, ktorá vystavila JWT. Spracovanie tohto tvrdenia závisí od špecifikácie aplikácie.
- **„sub“**: Identifikuje subjekt, ktorý požiadal o vystavenie JWT. Hodnota tohto tvrdenia musí byť lokálne unikátna.
- **„aud“**: Identifikuje publikum, pre ktoré je JWT určený. Spracovanie tohto tvrdenia závisí od špecifikácie aplikácie.
- **„exp“**: Identifikuje čas exspirácie, po ktorom musí byť JWT odmietnutý.
- **„nbf“**: Identifikuje čas, pred ktorým nesmie byť JWT akceptovaný.
- **„iat“**: Identifikuje čas, kedy bol JWT vystavený.
- **„jti“**: Identifikuje samotný JWT. Zabezpečuje, aby rovnaká hodnota nebola pridelená viacerím objektom.

## 4.1 Security Assertion Markup Language (SAML)

Táto sekcia je prevzatá z [18].

SAML je štandard založený na XML, ktorý umožňuje zdieľať autentifikačné a autorizačné informácie o identite medzi bezpečnostnými doménami. Je založený na protokole požiadavok/odpoveď, v ktorom jedna strana (všeobecne poskytovateľ služby) požiada o konkrétnu informáciu o identite užívateľa a druhá strana (najčastejšie poskytovateľ identít) odpovedá so žiadanou informáciou takým spôsobom, že užívateľ dokáže byť na konci tohto procesu identifikovaný a autentifikovaný. SAML sa skladá zo štyroch kľúčových elementov: tvrdenia, protokoly, väzby a profily.

SAML tvrdenie, anglicky *assertion*, je vyhlásenie o užívateľovi, že tento užívateľ bol úspešne potvrdený (napríklad autentifikovaný) na strane poskytovateľa identít pre konkrétneho poskytovateľa služby. Tvrdenie môže obsahovať tri rôzne typy vyhlásení, ktoré poskytovateľ služby môže použiť pre adekvátne poskytovanie jeho služieb užívateľovi:

1. **Autentifikačné vyhlásenie**: je použité ako dôkaz, že užívateľ bol autentifikovaný poskytovateľom identít v konkrétnom čase.
2. **Vyhlásenie o atribútoch**: je požité pre špecifikovanie atribútov užívateľa.
3. **Autorizačné vyhlásenie**: je použité ako dôkaz, že je užívateľ oprávnený vykonávať špecifické činnosti nad konkrétnym prostriedkom za určených podmienok.

SAML protokol je použitý pre definovanie pravidiel, ktoré určujú, akým spôsobom sa budú SAML elementy zabaľovať do paketu požiadavok/odpoveď a akým spôsobom sa budú tieto elementy rozbaľovať a spracovávať na strane prijímateľa.

SAML väzba, anglicky *binding*, sa používa pre mapovanie SAML protokolu na konkrétny komunikačný protokol, najčastejšie HTTP, pomocou ktorého sa budú SAML elementy posielať, a špecifikuje mechanizmus, pomocou ktorého sa budú SAML elementy vkladať, napríklad do spomenutého HTTP paketu.

SAML profily kombinujú predchádzajúce elementy a popisujú, akým spôsobom môžu byť implementované pre konkrétne prípady užitia. SAML Bol navrhnutý modulárne a rozšíriteľné, aby mohol byť použitý v rôznych prípadoch užitia. Najdôležitejší prípad užitia a na čo sa SAML predovšetkým zameriava je poskytovanie SSO medzi doménami webového prehliadača.

Dôvera medzi poskytovateľom identít a poskytovateľom služby v SAML hrá dôležitú úlohu. Dôvera je naviazaná pomocou výmeny metadát medzi IdP a SP, ktoré sú ukladané do adekvátnych repozitárov, ktoré pomáhajú vytvoriť každej strane list dôverných strán. Táto výmena nastáva po podpísaní technickej zmluvy medzi IdP a SP a musí byť uskutočnená pred akoukoľvek interakciou medzi spomenutými stranami.

Základnou konštrukciou SAML je SAML token. Tento token sa často preberá a adaptuje pre potreby iných protokolov a ich špecifikácií [15].

## 4.2 OAuth 2.0

Táto sekcia je prevzatá z [23].

OAuth 2.0 je **autorizačný** framework, dnes už štandard, ktorý umožňuje aplikáciám tretích strán získať obmedzený prístup k službe HTTP. Tento prístup dokážu získať v mene vlastníka prostriedku tým, že sa uskutoční schvaľovacia interakcia medzi vlastníkom prostriedku a službou HTTP, alebo tým, že sa povolí získať prístup priamo aplikácií tretej strany vo vlastnom mene.

OAuth 2.0 nahrádza už nepoužívaný protokol OAuth 1.0 a bol vytvorený na základe špecifického profilu pre SAML 2.0, anglicky nazývaný *SAML 2.0 Bearer Assertion Profile for OAuth 2.0*, ktorý bol dizajnovaný pre prostredia využívajúce autorizačné oprávnenia pre RESTovú službu [15].

Tento protokol definuje štyri dôležité roly:

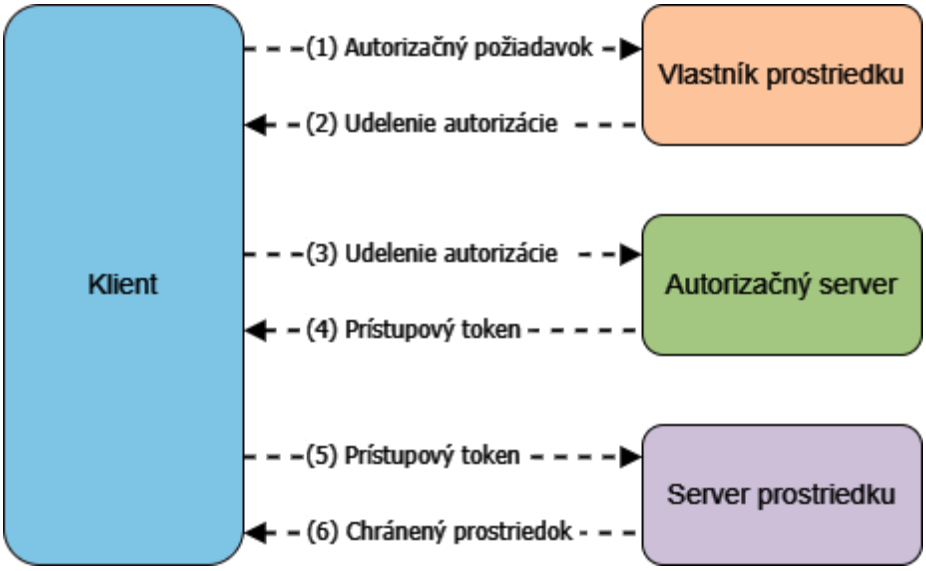
1. **resource owner**: Vlastník prostriedku je subjekt, ktorý dokáže udeliť prístup k chránenému prostriedku. Ak je vlastník prostriedku osoba, označuje sa ako **koncový užívateľ (identita)**.

2. **resource server**: Server prostriedku, na ktorom je umiestnený chránený prostriedok. Je schopný prijímať požiadavky o chránené prostriedky a odpovedať na ne pomocou prístupového tokenu. Typicky sa jedná o poskytovateľa služby vystavujúceho API.

3. **client**: Klient je **aplikáca**, ktorá vytvára požiadavok o prístup k chránenému prostriedku v mene vlastníka prostriedku s jeho oprávneniami. Termín „klient“ neimplementuje žiadnu špecifickú charakteristiku (ako napríklad, či je aplikácia spúšťaná na servery, z počítača, alebo iného zariadenia).

4. **authorization server**: Autorizačný server (môže byť súčasťou serveru prostriedku, alebo poskytovateľ identít), ktorý vystavuje prístupový token pre klienta po úspešnej autentifikácií vlastníka prostriedku.

Autorizačný server môže predstavovať zároveň aj server prostriedku, alebo úplne iný, separátny subjekt. Jeden autorizačný server dokáže vystaviť prístupové tokeny akceptované viacerými servermi prostriedkov.




Obr. 4.1: Abstraktný komunikačný tok protokolu OAuth 2.0, prebratý z [23]

Abstraktný komunikačný tok protokolu OAuth 2.0 na obrázku 4.1 popisuje interakciu medzi spomenutými rolami a skladá sa z:

(1) Klient požiada o autorizáciu od vlastníka prostriedku. Autorizačný požiadavok môže byť požiadaný priamo od vlastníka prostriedku (ako je znázornené), ale preferovaný spôsob požiadania je cez autorizačný server.

(2) Klient obdrží udelenie autorizácie.

(3) Klient požiada o prístupový token pomocou autentifikácie na strane autorizačného serveru, ku ktorej pridá udelenie autorizácie.

(4) Autorizačný server autentifikuje klienta, overí udelenie autorizácie a ak je validné, vystaví prístupový token.

(5) Klient požiada o prístup k chránenému prostriedku zo serveru na ktorom je prostriedok umiestnený a autentifikuje sa pomocou prístupového tokenu.

(6) Server prostriedku overí prístupový token a ak je validný, poskytne požiadavok.

Preferovaná metóda pre udelenie autorizácie od serveru prostriedku (bod (1) a bod (2)) je prostredníctvom požiadania cez autorizačný server.

Udelenie autorizácie, anglicky *authorization grant*, je oprávnenie, ktoré reprezentuje autorizáciu vlastníka prostriedku (pre prístup k zabezpečeným prostriedkom). Toto udelenie je použité klientom, pre získanie prístupového tokenu. OAuth 2.0 špecifikuje štyri typy autorizačného udelenia:

1. **Authorization Code**: Udelenie autorizácie pomocou autorizačného kódu.
2. Implicit: Implicitné autorizačné udelenie.
3. Resource Owner Password Credentials: Udelenie autorizácie heslom vlastníka prostriedku.

4. Client Credentials: Udelenie autorizácie autentifikáciou klienta.

OAuth 2.0 protokol špecifikuje dva koncové body, anglicky *endpoints* (HTTP prostriekdy) pre autorizačný server:

- Koncový bod autorizácie, anglicky *authorization endpoint*: klient posiela na tento koncový bod požiadavok o autorizáciu vlastníka prostriedku skrz webový prehliadač vlastníka prostriedku.
- Koncový bod tokenu, anglicky *token endpoint*: klient posiela na tento koncový bod udelenie autorizácie a autorizačný server ho vymieňa za prístupový token.

a jeden koncový bod pre klienta:

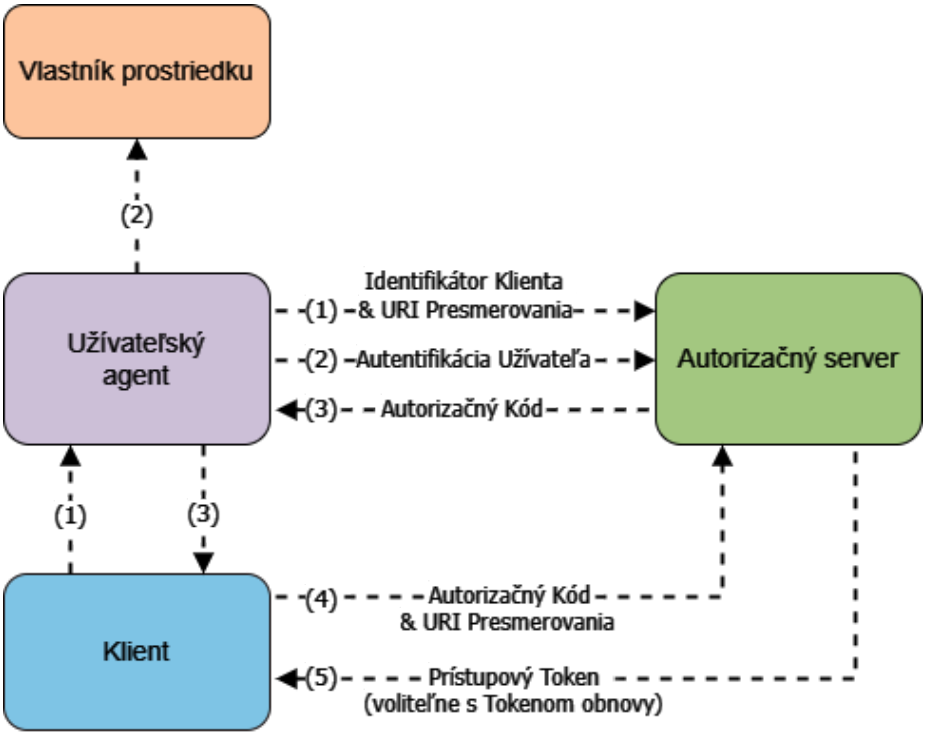
- Koncový bod presmerovania, anglicky *redirection endpoint*: autorizačný server posiela na tento koncový bod odpoveď, obsahujúcu autorizačné oprávnenia vlastníka prostriedku pre klienta skrz webový prehliadač vlastníka prostriedku.

### Udelenie autorizačného kódu

Táto podsekcia je prevzatá z [23].

Autorizačný kód je získaný skrz autorizačný server, ktorý sa nachádza medzi klientom a vlastníkom prostriedku. Namiesto toho, aby klient požiadal o autorizačné oprávnenia priamo vlastníka prostriedku, klient presmeruje vlastníka prostriedku na autorizačný server (pomocou jeho užívateľského agenta, anglicky *user-agent*, definovaný v HTTP protokole), ktorý vlastníka prostriedku presmeruje naspäť ku klientovi už s autorizačným kódom.

Pred tým, ako autorizačný server presmeruje vlastníka prostriedku naspäť ku klientovi s autorizačným kódom, vlastník prostriedku bude autentifikovaný na strane autorizačného serveru a po úspešnej autentifikácií získa autorizačný server autorizáciu, ktorú prepošle klientovi. Takýmto spôsobom nebudú prihlasovacie údaje vlastníka prostriedku nikdy zdieľané s klientom, pretože vlastník prostriedku sa autentifikuje len pomocou autorizačného serveru.




Obr. 4.2: Komunikačný tok autorizačného kódu skrz autorizačný server, prebratý z [23]

Na obrázku 4.2 je bod (1), (2) a (3) rozdeľený na dve časti. Toto je spôsobené prechodom cez užívateľského agenta. Tento prípad užitia získania autorizačného udelenia je založený na presmerovaní a klient musí byť schopný komunikovať s užívateľským agentom vlastníka prostriedku (typicky webový prehliadač) a musí byť schopný prijímať prichádzajúce požiadavky a odpovede (skrz presmerovanie) z autorizačného serveru. Popis komunikačného toku je nasledovný:

(1) Klient zaháji komunikáciu tým, že presmeruje užívateľského agenta vlastníka prostriedku na koncový bod autorizácie, anglicky *authorization endpoint*. Klient prídava svoj identifikátor, vyžadovaný *scope*, lokálny stav a presmerovaciu URI, na ktorú bude užívateľský agent vlastníka prostriedku presmerovaný po ukončení toku.

(2) Autorizačný server autentifikuje vlastníka prostriedku (pomocou užívateľského agenta) a rozhodne, či má dostatočné oprávnenia pre klienta.

(3) Za predpokladu, že vlastník prostriedku má dostatočné oprávnenie, autorizačný server presmeruje jeho agenta naspäť ku klientovi. Presmerovacia URI v tomto momente obsahuje autorizačný kód a lokálny stav poskytnutý klientom.

(4) Klient požiada autorizačný server o prístupový token skrz koncový bod tokenu, anglicky *token endpoint*. V tomto momente sa klient autentifikuje na strane autorizačného serveru a poskytne presmerovaciu URI, ktorú obdržal.

(5) Autorizačný server autentifikuje klienta, overí autorizačný kód a porovná presmerovaciu URI. Ak je validná, autorizačný server vystaví prístupový token a voliteľne aj token obnovy, anglicky *refresh token*.

Udelenie pomocou autorizačného kódu začína HTTP požiadavkom klienta o získanie autorizácie, ktorý posiela na koncový bod autorizácie. Príklad (pre lepší nákres boli v nasledujúcich príkladoch použité oddeľovače riadku):

```
GET /authorize?response_type=code
    &client_id=s6BhdRkqt3
    &state=xyz
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

Ak je vlastník prostriedku oprávnený, autorizačný server vystaví autorizačný kód a pošle ho klientovi skrz užívateľského agenta vlastníka prostriedku, ktorého presmeruje pomocou presmerovacej URI na klientsky koncový bod presmerovania už s autorizačným kódom. Príklad autorizačnej HTTP odpovede:

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA
          &state=xyz
```

Klient následne pošle HTTP požiadavok na koncový bod tokenu s autorizačným kódom, ktorý obdržal od autorizačného serveru, pre získanie prístupového tokenu. Príklad:

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

Po úspešnom overení vystaví autorizačný server prístupový token (voliteľne aj token obnovy). Príklad:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"example",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
  "example_parameter":"example_value"
}
```

### Access Token

Prístupový token stelesňuje **autorizačné** oprávnenia klienta. Prístupový token je používaný vždy pri prístupe klienta k chránenému prostriedku. Takýto token je vo forme reťazca znakov a reprezentuje autorizáciu, ktorá bola klientovi vystavená v mene vlastníka prostriedku. Tento reťazec znakov je väčšinou pre klienta nepriehľadný. Prístupový token špecifikuje rozsah, anglicky *scope* a trvanie platnosti prístupových oprávnení, ktoré boli udelené vlastníkom prostriedku a uplatnené spoluprácou medzi serverom prostriedku a autorizačným serverom. [23]

Prístupový token poskytuje abstraktnú vrstvu, premieňa rozličné autorizačné konštrukcie (napríklad meno a heslo v kombinácií s rolami) na jeden token, ktorý je čitateľný a prijímaný serverom prostriedku. Táto abstrakcia umožňuje viac reštriktívne udeľovanie oprávnení a taktiež eliminuje potrebu implementácie množstva autentifikačných metód na strane serveru prostriedku. [23]

Prístupový token sa používa pre klientov (aplikácie), ktorí chcú vykonávať API požiadavky v mene identity konkrétneho užívateľa v RESTových službách [16].

Prístupový token vo forme reťazca znakov dokáže byť zabalený do rôznych formátov a štruktúr, ale najčastejšie sa používa práve vo formáte JWT spomenutom v sekcií 4, vďaka kryptografickým prvkom, ktoré pomáhajú ochrániť informácie o identite a jej oprávneniach v HTTP komunikácií a jeho malej pamäťovej veľkosti. Takýto prístupový token sa nazýva JWT prístupový token.

Špecifikácia, že sa jedná o JWT prístupový token a nie hociaký JWT token prebieha na základe konkrétne použitých hodnôt tvrdení v štruktúre JWT tokenu spomenutej v sekcií 4. JWT prístupový token musí obsahovať tvrdenie o použitom podpisovacom algoritme, to znamená, že hodnota tvrdenia „alg“ nesmie byť „none“. Pre účely JWT prístupového tokenu je dôležité aby bol digitálne podpísaný. Ďalej autorizačný server a server prostriedku musí podporovať RS256 podpisovací algoritmus. Hodnota tvrdenia „typ“ musí byť nastavená na „at+jwt“, aby bolo pre server prostriedku rozoznateľné, že sa nejedná o *JWT ID token* z protokolu OpenID Connect, ktorého úloha bude neskôr popísaná. [14]

Nasledujúci zoznam tvrdení, ktoré sa nachádzajú v užitočnom zaťažení, sú nevyhnutne potrebné pre JWT prístupový token [14]:

- **„iss“**
- **„exp“**
- **„aud“**
- **„sub“**
- **„iat“**
- **„jti“**
- **„client_id“**: identifikátor klientskej aplikácie. Tento identifikátor môže predstavovať aj sociálnu sieť, ktorej identifikátor je verejne známy.

Ak bol v autorizačnom požiadavku HTTP použitý parameter *scope*, ktorý definuje rozsah oprávnení, o ktoré si klient žiada v mene vlastníka prostriedku, musí byť obsiahnutý v užitočnom zaťažení JWT prístupového tokenu tvrdenie s menom „scope“. Toto tvrdenie bude obsahovať reťazce znakov, ktoré reprezentujú jednotlivé autorizačné oprávnenia

vlastníka prostriedku. Ak sa vyžiada oprávnenie, ktoré nie je podporované v žiadnom klientovi z publika (v tvrdení „aud“), mal by byť požiadavok odmietnutý s chybovou hláškou „invalid_scope“. [14]

V súkromnej implementácií JWT prístupového tokenu je možné špecifikovať ďalšie tvrdenia, v rámci užitočného zaťaženia, pre špecifické potreby aplikácie, ktorá s nimi ďalej pracuje. Mená týchto tvrdení môžu byť definované ľubovolne, pokiaľ sa nezhodujú s predregistrovanými menami tvrdení spomenutými v sekcií 4. Príklad autorizačného požiadavku HTTP s použitým *scope* parametrom [14]:

```
GET /as/authorization.oauth2?response_type=code
    &client_id=s6BhdRkqt3
    &state=xyz
    &scope=openid%20profile%20reademail
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
    &resource=https%3A%2F%2Frs.example.com%2F HTTP/1.1
 Host: authorization-server.example.com
```

Po overení bude vystavený JWT prístupový token s hlavičkou:

```
{"typ":"at+JWT","alg":"RS256","kid":"RjEwOwOA"}
```

a užitočným zaťažením (s tvrdeniami):

```
{
  "iss": "https://authorization-server.example.com/",
  "sub": "5ba552d67",
  "aud": "https://rs.example.com/",
  "exp": 1639528912,
  "iat": 1618354090,
  "jti" : "dbe39bf3a3ba4238a513f51d6e1691c4",
  "client_id": "s6BhdRkqt3",
  "scope": "openid profile reademail"
}
```

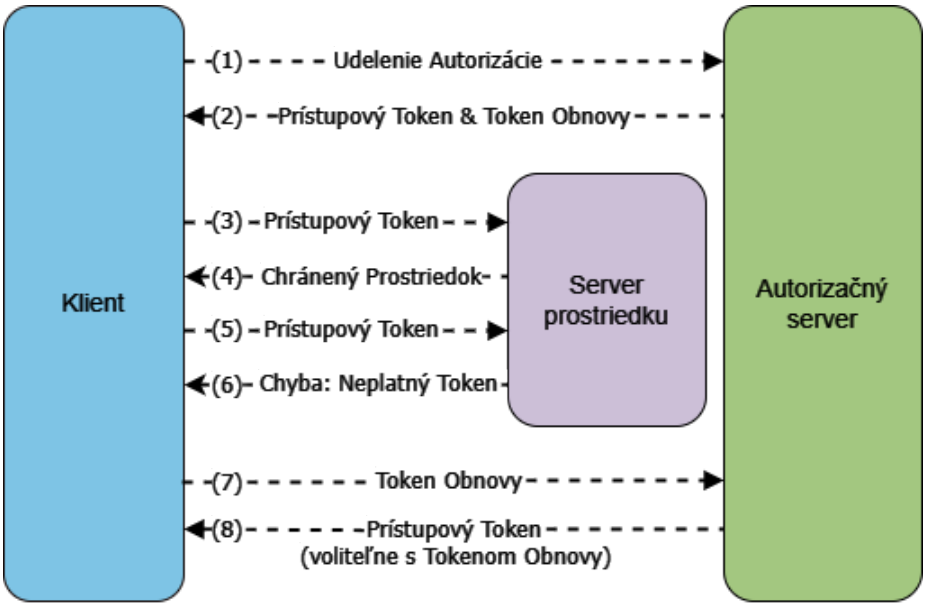
Užitočné zaťaženie je kritická časť, ktorá určuje či bude token použiteľný pre úspešný prístup k žiadanému prostriedku, alebo nie. Ak chce užívateľ pristúpiť k prostriedku vo webovej aplikácií, ale nemá dostatočné oprávnenia uložené v tejto časti tokenu, bude mu prístup zamietnutý. [10]

### Refresh Token

Táto podsekcia, obrázok aj popis komunikačného toku, sú prevzaté z [23].

Token obnovy stelesňuje oprávnenie pre získánie prístupového tokenu. Token obnovy vystavuje autorizačný server klientovi a je použitý pri opätovnom vyžiadaní prístupového tokenu vtedy, keď sa aktuálny prístupový token stane neplatným, alebo mu vyprší platnosť. Klient s takýmto tokenom dokáže vyžiadať aj doplnkové prístupové tokeny, ktoré majú identické, alebo menšie oprávnenia definované v „scope“ tvrdení, alebo kratšiu dobu

platnosti. Vystavenie tokenu obnovy je voliteľné. Ak autorizačný server podporuje vystavenie tokenu obnovy, bude vystavený práve vtedy, keď je vystavený aj prístupový token (bod (5) na obrázku 4.2). Narozdiel od prístupového tokenu je token obnovy určený len pre autorizačný server a nikdy nie je posielaný na server prostriedku.




Obr. 4.3: Obnovenie exspirovaného prístupového tokenu, prebratý z [23]

Na obrázku 4.3 je znázornený komunikačný tok obnovenia exspirovaného prístupového tokenu a skladá sa z:

(1) Klient požiada o prístupový token pomocou autentifikácie s autorizačným serverom a predloží autorizačné udelenie.

(2) Autorizačný server autentifikuje klienta, overí udelenie a ak je validné, vystaví prístupový token a token obnovy.

(3) Klient požiada o chránený prostriedok tým, že predloží prístupový token.

(4) Server prostriedku overí prístupový token a ak je validný, dodá chránený prostriedok.

(5) Bod (3) a (4) sa opakuje, pokým nevyprší platnosť prítupového tokenu. Ak klient vie, že prístupový token nie je validný, rovno skočí na bod (7). Ak nevie, vykoná požiadavok o chránený prostriedok s neplatným prístupovým tokenom.

(6) Kedže je prístupový token neplatný, server prostriedku vráti chybu.

(7) Klient požiada o nový prístupový token pomocou autentifikácie s autorizačným serverom a predloží token obnovy.

(8) Autorizačný server autentifikuje klienta, overí token obnovy a ak je validný, vystaví nový prístupový token (a voliteľne aj nový token obnovy).

## 4.3 OpenID Connect (OIDC)

OpenID Connect pridáva vrstvu identity nad protokolom OAuth 2.0. Umožňuje klientským aplikáciám overovať identitu koncového užívateľa na základe **autentifikačného** procesu na strane autorizačného serveru a tým získať základné informácie o profile identity koncového užívateľa spôsobom, ktorý je inšpirovaný RESTovou architektúrou. [35]

OAuth 2.0, ktorý poskytuje otvorenú špecifikáciu pre aplikácie tretích strán, je využívaný pre obmedzený prístup aplikáciám k HTTP prostriedkom v RESTových službách v mene vlastníka prostriedku (užívateľa). Definuje mechanizmus získania prístupového tokenu a tokenu obnovy, ale nedefinuje štandardnú metódu pre získanie informácií o prebehnutej autentifikácií a konkrétnych atribútov identity. OAuth 2.0 nie je schopný poskytnúť špecifické informácie o autentifikácií koncového užívateľa, dokáže len potvrdiť, že nejaká autentifikácia prebehla úspešne (alebo aj neúspešne), na základe ktorej bol (alebo nebol) vystavený prístupový token. Prístupový token obsahuje len autorizačné oprávnenia pre aplikáciu (klienta), ktorá ich môže využiť na prístup k chráneným prostriedkom (na servery prostriedku) v mene vlastníka prostriedku. Tento problém adresuje práve OpenID Connect.

OpenID Connect implementuje autentifikáciu ako rozšírenie pre proces autorizácie v OAuth 2.0. Použitie tohto rozšírenia je požiadané klientom, ktorý špecifikuje hodnotu „openid“ v tvrdení „scope“ autorizačného požiadavku. Informácie o autentifikácií a identite koncového užívateľa sú vrátené opäť vo forme JWT tokenu, ktorý sa nazýva token identity, anglicky *ID Token*. Autentifikačné servery, ktoré podporujú OpenID Connect nad OAuth 2.0 autorizáciou sa nazývajú poskytovatelia OpenID, anglicky *OpenID Providers* (OPs). OpenID Connect vykonáva autentifikáciu koncového užívateľa, alebo rozhoduje, či je koncový užívateľ už prihlásený. Výsledok autentifikácie je bezpečne posielaný na klienta, ktorý sa naň spolieha a preto je v tomto kontexte anglicky nazývaný *Relying Party* (RP). [35]

### ID Token

Tak ako prístupový token, aj token identity obsahuje špecifické tvrdenia v štruktúre JWT, ktoré ho pomáhajú rozlíšiť od ostatných JWT tokenov. Všetky tvrdenia, ktoré su nevyhnutne potrebné pre token identity v štruktúre JWT ostávajú rovnaké, ako pre prístupový token.

Navyše pribudlo niekoľko voliteľných tvrdení [35]:

- **„auth_time“**: Čas, kedy sa koncový užívateľ autentifikoval.
- **„nonce“**: Reťazec znakov, ktorý asociuje klientskú reláciu s tokenom identity.
- **„acr“**: Referencia na triedu autentifikačného kontextu. Indikuje, že autentifikácia bola uspokojivá.
- **„amr“**: Referencia na autentifikačné metódy, ktoré boli použité pri autentifikácií koncového užívateľa.
- **„azp“**: Strany autorizácie, pre ktoré bol token identity vystavený.

### Autentifikácia pomocou autorizačného kódu

V tomto prípade užitia sú všetky tokeny vrátené z koncového bodu tokenu na strane autorizačného serveru. [35]

Ako pri OAuth 2.0, tak aj v tomto prípade bude získaný autorizačný kód pre klienta, ktorý ho dokáže vymeniť za prístupový token a token identity zároveň [35] [23]. Tento komunikačný tok je obdobou komunikačného toku popísanom v ref flow s niekoľkými zmenami a skladá sa z [35]:

(1) Klient pripraví autentifikačný HTTP požiadavok, ktorý obsahuje špecifické parametre, ktoré ho rozdeľujú od autorizačného HTTP požiadavku.

(2) Klient pošle tento HTTP požiadavok na koncový bod autorizácie, ktorý sa nachádza na autorizačnom servery.

(3) Autorizačný server autentifikuje koncového užívateľa.

(4) Autorizačný server obdrží autorizáciu od koncového užívateľa.

(5) Autorizačný server presmeruje koncového užívateľa naspäť ku klientovi už s autorizačným kódom.

(6) Klient obdrží autorizačný kód a pošle ho na koncový bod tokenu, ktorý sa nachádza na autorizačnom servery.

(7) Klient obdrží odpoveď od autorizačného serveru, ktorá obsahuje prístupový token a aj token identity.

(8) Klient overí token identity a obdrží informácie o identite koncového užívateľa a prebehnutej autentifikácií.

Autentifikácia koncového užívateľa prebehne na koncovom bode autorizácie (na strane autorizačného serveru) takým spôsobom, že webový prehliadač koncového užívateľa bude presmerovaný na tento koncový bod, kde bude autentifikovaný a následne autorizovaný pomocou jedného HTTP požiadavku, ktorý sa skladá z parametrov špecifikovaných pre OAuth 2.0 a zároveň parametrov špecifikovaných pre OpenID Connect [35]. Príklad autentifikačného požiadavku:

```
GET /authorize?
    &response_type=code
    &scope=openid%20profile%20email
    &client_id=s6BhdRkqt3
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb HTTP/1.1
  Host: server.example.com
```

Validácia autentifikačného HTTP požiadavku prebieha na strane autorizačného serveru tak, že sa skontrolujú všetky OAuth 2.0 parametre (autorizačné) a zároveň sa overí parameter scope, v ktorom sa musí nachádzať hodnota openid, ktorá špecifikuje, že sa bude koncový užívateľ autentifikovať pomocou OpenID Connect. [35]

Autentifikačná HTTP odpoveď je rovnaká, ako pri OAuth 2.0, klient obdrží autorizačný kód, a pošle HTTP požiadavok (vrátane získaného kódu) na koncový bod tokenu na strane autorizačného serveru pre získanie tokenov [23]. Autorizačný server overí, že prijatý autorizačný kód bol pôvodne vystavený v HTTP odpovedi pomocou OpenID Connect autentifikačného požiadavku a následne vystaví všetky tokeny, ktoré pošle skrz webový prehliadač koncového užívateľa klientovi [35]. Príklad úspešnej odpovede s vystavenými tokenami pre klienta [35]:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
 "access_token": "SlAV32hkKG",
 "token_type": "Bearer",
 "refresh_token": "8xLOxBtZp8",
 "expires_in": 3600,
 "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
   yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
   NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3FOMyIsCiAibm9uY2UiOiAibi0wUzZ
   fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
   AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
   Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
   NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
   QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
   K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
   XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
}
```

Typ tokenu nosič, anglicky *bearer* je používaný vo webových aplikáciách a volaní API pomocou HTTP cez TLS. Nosič sa vzťahuje pre prístupový token (nie token identity) a špecifikuje, že ten klient, ktorý vlastní takýto nosič, je oprávnený použiť ho v hlavičke HTTP autorizačného požiadavku o prístup k chránenému prostriedku bez nutnosti overenia, že tento klient vlastní kryptografický kľúč. [26]

Ak chce klient získať detailné informácie o koncovom užívateľovi, pošle HTTP požiadavok na koncový bod informácie o užívateľovi, anglicky *UserInfo endpoint*, ktorý sa nachádza na strane autorizačného serveru [35]. HTTP požiadavok pre získanie detailných informácií o užívateľovi vyzerá takto:

```
GET /userinfo HTTP/1.1
Host: server.example.com
Authorization: Bearer SlAV32hkKG
```

Autorizačný server overí požiadavok a nosič, ktorý bol v rámci požiadavku poslaný a ak overenie prebehne úspešne, pošle klientovi HTTP odpoveď s detailnými informáciami o užívateľovi:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
 "sub": "248289761001",
 "name": "Jane Doe",
 "given_name": "Jane",
 "family_name": "Doe",
 "preferred_username": "j.doe",
```

```
"email": "janedoe@example.com",
"picture": "http://example.com/janedoe/me.jpg"
}
```

V odpovedi je možné nájsť štandardné (pred-registrované) tvrdenia o identite koncového užívateľa, ktoré boli získané pomocou OpenID Connect. Zoznam pred-registrovaných tvrdení [35]:

- **sub** (string): identifikátor subjektu
- **name** (string): meno
- **given_name** (string): krstné meno
- **family_name** (string): priezvisko
- **middle_name** (string): stredné meno
- **nickname** (string): prezývka
- **preffered_username** (string): preferovaná prezývka
- **profile** (string): URL na profil koncového užívateľa
- **picture** (string): profilová fotka
- **website** (string): URL webovej stránky koncového užívateľa
- **email** (string): email
- **email_verified** (boolean): či bol email overený
- **gender** (string): pohlavie
- **birthdate** (string): dátum narodenia
- **zoneinfo** (string): časová zóna
- **locale** (string): lokácia koncového užívateľa
- **phone_number** (string): telefónne číslo
- **phone_number_verified** (boolean): či bolo telefónne číslo overené
- **address** (JSON object): adresa
- **updated_at** (string): čas, kedy boli informácie naposledy aktualizované

Ďalšie tvrdenia o identite koncového užívateľa môžu byť špecifikované ľubovolne, pokiaľ sa nezhodujú s tými pred-registrovanými a nenastáva pri tom kolízia typov hodnôt [35].

## 4.4 Access Token vs ID Token

Prístupový token zabezpečuje prístup (autorizáciu) pre klientov tretích strán v mene vlastníka prostriedku a je používaný na volanie API. Token identity zabezpečuje dodatočné informácie o autentifikácií a samotných (štandardných) atribútov identity koncového užívateľa [16]. Keďže je OpenID Connect rozšírenie pre OAuth 2.0, OpenID Connect dokáže všetko to, čo dokáže OAuth 2.0 a navyše definuje token identity. Koniec koncov záleží na klientovi, či potrebuje narábať s dátami koncového užívateľa, alebo koncový užívateľ nehrá žiadnu kľúčovú rolu v procese získavania prostriedkov v klientskej webovej aplikácií. Použitie jednotlivých protokolov bude znázornený na nasledujúcich prípadov užitia.

### Prvý prípad užitia

Klient je webová aplikácia, ktorá agreguje záznamy o počasí, kde tieto záznamy nijak nesúvisia s užívateľom, ktorý je v klientskej webovej aplikácií prihlásený (užívateľ nie je vlastník prostriedku). Záznamy (prostriedky) sa nachádzajú na verejnom servery počasia (server prostriedkov), ktorý poskytuje prístup k jeho API. Aj napriek tomu, že je server prostriedkov verejný, je v jeho záujme chrániť svoje prostriedky a bude príjmať len autorizované požiadavky. Jediné čo klient potrebuje, je autentifikovať samého seba na autorizačnom servery (jeden zo serverov, ktorý poskytuje informácie o počasí). Tento prípad užitia je typický pre samostatné OAuth 2.0 s využitím prístupového tokenu. Autorizačný server vystaví (po úspešnej autentifikácií klienta) prístupový token, ktorý bude klient (vždy pri aktualizácií dát počasia) používať pri HTTP požiadavkoch smerom na server prostriedkov. Ukážka HTTP odpovede vystaveného prístupového tokenu od autorizačného serveru pre klienta:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
 "token_type": "Bearer",
 "access_token": "<jwt-hlavička>.<jwt-užitočné-zaťaženie>
                  .<jwt-podpis>",
 "expires_in": 3600,
}
```

Modrá časť užitočného zaťaženia predstavuje tvrdenia JWT prístupového tokenu, medzi ktorými je definovaný rozsah autorizačných oprávnení udelených pre klienta:

```
{
  "iss": "https://auth.pocasie.com",
  "aud": "api.pocasie.com",
  "sub": "nahodny_uzivatel",
  "exp": 1615369919,
  "iat": 1615366319,
  "scope": "pocasie:read predpoved:read"
}
```

### Druhý prípad užitia

Klient je interná webová aplikácia spoločnosti, ktorá poskytuje nástenku svojim zamestnancom. Nástenka poskytuje informácie o spoločnosti prostredníctvom príspevkov a zamestnanci ich možu komentovať. Aplikácia je jednoduchá a každý zamestnanec v tejto spoločnosti dokáže komentovať príspevky bez ďalších nutných oprávnení. V tomto prípade sa použije OpenID Connect s prístupovým tokenom a tokenom identity, ktorý klientovi zabezpečí základné atribúty o identite (meno, priezvisko, email). Autorizačný server (môže byť napríklad *Active Directory*), vystavý prístupový token ako nosič a token identity pre klienta:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
 "token_type": "Bearer",
 "access_token": "<jwt-hlavička>.<jwt-užitočné-zaťaženie>
                  .<jwt-podpis>",
 "id_token": "<jwt-hlavička>.<jwt-užitočné-zaťaženie>.<jwt-podpis>",
 "expires_in": 3600,
}
```

Modrá časť užitočného zaťaženia predstavuje tvrdenia JWT prístupového tokenu, medzi ktorými je definovaný rozsah autorizačných oprávnení. V tomto prípade je použitá len hodnota öpenid", ktorá identifikuje použitie OpenID Connect, pretože klient nepotrebuje ďalšie autorizačné oprávnenia:

```
{
  "iss": "https://adfs.spolocnost.com/adfs",
  "aud": "api.spolocnost.com",
  "sub": "johndoe_id",
  "exp": 1615369919,
  "iat": 1615366319,
  "scope": "openid"
}
```

Červená časť užitočného zaťaženia predstavuje tvrdenia JWT tokenu identity, medzi ktorými sú definované štandardné tvrdenia (atribúty) o identite užívateľa a identifikátor klienta, pre ktorého bol tento token identity vystavený:

```
{
  "iss": "https://adfs.spolocnost.com/adfs",
  "sub": "johndoe_id",
  "aud": "nastenka",
  "exp": 1615369919,
  "iat": 1615366319,
```

```
  "name": "John Doe",
  "email": "johndoe@spolocnost.com"
}
```

## 4.5 Externé služby

V dnešnej dobe existuje mnoho webových aplikácií, ktoré podporujú autentifikáciu pomocou externej služby (poskytovateľa identít), ako je napríklad Google, Facebook, LinkedIn a podobné. Týto poskytovatelia podporujú OAuth 2.0, ale aj OpenID Connect, ktorý je uprednostňovaný práve kvôli vrstve identity. Zvolenie protokolu opäť záleží od potrieb webovej aplikácie. Na nasledujúcom prípade užitia bude opísaný komunikačný tok OpenID Connect medzi webovou aplikáciou a verejným poskytovateľom identít, spoločnosťou Google (scenár je čisto ilustračný a nevedie k propagácií žiadnej spoločnosti).

Webová aplikácia (klient) podporuje autentifikáciu pomocou poskytovateľa identít Google, ktorý v tomto prípade predstavuje autorizačný server. Google je zároveň aj server prostriedku, pretože ukladá chránené prostriedky užívateľa (užívateľské dáta v Google aplikáciách), ktorý je vlastník prostriedku. Užívateľ vstúpi do webovej aplikácie a zvolí možnosť autentifikácie pomocou Google. Webová aplikácia presmeruje užívateľa do prihlasovacieho formulára na autorizačnom servery Google.

Užívateľ sa prihlási do jeho Google účtu a po úspešnej autentifikácií sa Google opýta užívateľa, či webová aplikácia, ktorá ho presmerovala, môže obdržať atribúty o jeho identite (meno, priezvisko, profilovú fotku) a autorizáciu pre prístup, napríklad ku jeho fotkám v aplikácií Google Photos. Užívateľ súhlasí a autorizačný server vytvorí autorizačný kód, ktorý cez webový prehliadač úžívateľa (užívateľský agent) pošle webovej aplikácií. Webová aplikácia obdrží autorizačný kód a pošle ho naspäť na autorizačný server a tento krát požiada o vystavenie tokenov.

Autorizačný server overí autorizačný kód a vystaví prístupový token a token identity, ktoré pošle webovej aplikácií. Webová aplikácia obdrží tokeny, asociuje si ich s užívateľom, ktorému patria a vytvorí relačný cookie, ktorý uloží do webového prehliadača užívateľa. Následne je užívateľ presmerovaný na webovú aplikáciu, v ktorej je už autentifikovaný. Relačný cookie zabezpečí, že ak užívateľ vykonáva požiadavky na webovej aplikácií, ostane prihlásený a nebude nútený sa znovu autentifikovať. Webová aplikácia má k dispozícií prístupový token, ktorý je autorizovaný pre volanie API koncového bodu fotiek Google v mene užívateľa a zároveň má k dispozícií token identity, ktorý poskytuje webovej aplikácií atribúty o identite užívateľa (meno, priezvisko, profilovú fotku).

Celý scenár prebehol bezpečne bez toho, aby užívateľ niekedy zverejnil svoje prihlasovacie údaje webovej aplikácií a bez toho, aby sa užívateľ musel u klienta registrovať s novými prihlasovacími údajmi. Ak užívateľ pristúpi k inej webovej aplikácií, ktorá tiež podporuje autentifikáciu pomocou poskytovateľa identít Google, webová aplikácia presmeruje užívateľa na autorizačný server od Google, ktorý overí, že užívateľ má vo svojom webovom prehliadači validný relačný cookie a prebehne úspešná autentifikácia. Bez nutnosti vypĺňania prihlasovacieho formulára, bude užívateľ vyzvaný potvrdiť (alebo zamietnuť) webovej aplikácií autorizáciu o prístup, napríklad ku aplikácií Google Sheets. Užívateľ potvrdí autorizáciu a bude presmerovaný do webovej aplikácie, v ktorej bude autentifikovaný a vďaka SSO nemusel vypĺňať prihlasovacie údaje vôbec (prebehne autentifikácia bez povšimnutia).

# Kapitola 5

# Porovnanie nástrojov pre správu identít

Táto kapitola popisuje tri zvolené nástroje pre správu identít, ich sadu procesov a služieb, ktoré poskytujú.

## 5.1 midPoint

Nástroj midPoint[1] od spoločnosti Evolveum je open-source riešenie, ktoré poskytuje manažment identít a administratívu pre spoločnosti, ktoré chcú riadiť procesy zamestnancov na jednom mieste. Zameriava sa na životné cykly identít, automatizované procesy, identity provisioning, roly, prístupy a propagáciu dát do koncových systémov. Nástroj poskytuje vysokú mieru škálovania. Je programovaný v jazyku Java a objekty sú reprezetované ako XML dokumenty.

Objekty, ako sú napríklad užívatelia, role, organizačné jednotky, úlohy, či schvaľovacie procesy, sú reprezentované ako modifikovateľné XML dokumenty, ktorých definícia sa nachádza v dokumentácií. MidPoint ponúka užívateľom samoobsluhu, ako napríklad zmenu hesla, žiadosť o priradenie roly, alebo organizačnej jednotky, kde takáto žiadosť podlieha schvaľovacím procesom, ktorý môže byť automatizovaný, alebo vďaka skriptovaniu aj algoritmicky priradený cieleným schvalovateľom. Dokáže plánovať úlohy a procesy do kalendára, ktoré môžu byť pravidelne spúšťané. Nástroj poskytuje funkcionalitu reportovania do CSV súborou a tieto reporty dokážu byť špecificky prispôsobené. Poskytuje detailné logy, monitorovanie a auditné reporty, ktoré posilňujú regulačné požiadavky klienta. MidPoint dokáže pracovať s veľkými množstvami dát veľmi efektívne a preto je vhodný pre spoločnosti s veľkým počtom zamestnancov, externistov, dodávateľov, organizačných jednotiek, či rolí.

Základným komponentom je repozitár, ktorý slúži ako úložisko dát vo forme relačnej databázy. Ukladá všetky dáta, užívateľské záznamy a udalosti, auditné logy a bezpečnostné informácie. Podporuje PostgreSQL, MySQL, Oracle a SQL Server, ktoré umožňujú flexibilitu pre potreby klienta.

Ďalší kľúčový komponent je Identity Connector Framework (ConnId), ktorý poskytuje integráciu veľkého množstva koncových systémov, medzi ktoré patrí napríklad LDAP, alebo Active Directory. Jednotlivé konektory zabezpečujú pripojenie a manažovanie identít medzi veľkým množstvom koncových systémov na jednom mieste a bez povšimnutia.

[1] https://evolveum.com/midpoint/

Nástroj ponúka Provisioning Engine, ktorý poskytuje synchronizáciu dát o identitách medzi koncovými systémami a zabezpečuje propagáciu zmien (vytvorenie, vymazanie, modifikáciu) do jednotlivých koncových systémov, ak spĺňajú kritériá konfigurácie bezpečnostnej politiky a schvaľovacích procesov.

Ďalej obsahuje Workflow Engine, ktorý manažuje komplexné biznisové procesy. Tento komponent poskytuje automatizáciu procesov schvaľovania pri priradzovaní rolí, ogranizačných jednotiek, zmenách atribútov a podobných procesoch, kde sa vyžaduje kontrola a schválenie nadriadenou osobou.

Ďalší komponent je užívateľské rozhranie, postavené na HTML5, ktoré podporuje responzívne zobrazenie a poskytuje funkcie, ktoré sú užívateľom zobrazené na základe priradených oprávnení. Užívateľské rozhranie je graficky veľmi pokročilé, ale pre neznalých administrátorov môže byť orientácia náročná a komplexná.

Riadenie prístupov je založené na rolách, organizačných jednotkách a fine-grained autorizačnej politike, ktorá okrem rolí používa atribúty identity a účel pre vypočítanie autorizačných oprávnení. Nástroj ďalej ponúka RESTful API, ktorá zabezpečuje interakciu s externými systémami a integráciu midPoint s vlastnými klientskymi aplikáciami. API hrá kritickú úlohu pri integrácií nástroja do väčšieho IT ekosystému.

MidPoint poskytuje obrovskú mieru konfigurácie a prispôsobenia a taktiež rozšírené prispôsobenie pomocou skriptovania v Groovy, Python, alebo JavaScript. Táto funkcionalita je veľmi často využívaná pre zabezpečenie špecifických procesov spoločností a tvorí značnú časť prispôsobovania automatizovaných procesov, procesov schvaľovania, propagácie a synchronizácie dát. Nástroj zaostáva v metódach jednotného prihlásenia, ale na druhú stranu ponúka veľmi detailnú správu procesov identít a extrémnu prispôsobivosť. Jeho silnou stránkou je možnosť automatického spustenia dodatočných, externých skriptov, ktoré môžu byť uložené v adresárovej štruktúre na servery midPoint. Tieto skripty dokážu byť volané z adresárovej štruktúry po dokončení naplánovanej úlohy, napríklad ako post-processing výstupných reportov. Plánovač úloh je založený na crontab vzoroch.

### Podpora

Spoločnosť poskytuje profesionálnu podporu a konzultácie, zverejňuje Java dokumentáciu pre Groovy skripty a dokumentáciu XML schémy, v ktorej sú špecifikované jednotlivé objekty a ich štruktúra, ďalej zverejňuje vlastnú Wiki stránku a GitHub s príkladmi konfigurácie a použitia skriptovania. Spoločnosť poskytuje aktívny emailový zoznam, v ktorom klienti reportujú problémy, alebo zdieľajú riešenia s komunitou. Spoločnosť usporiadava online stretnutia, v ktorých komentujú rozšírenia nástroja a ukazujú ich na praktických príkladoch. Ďalej poskytuje svojim klientom možnosť sa aktívne zapojiť do vývoju a svojimi návrhmi ovplyvniť smer, akým sa bude nástroj uberať. Klienti si taktiež môžu predplatiť podporu, ktorá sa primárne zaoberá odstránením chýb produkčného prostredia midPoint, alebo pomáha pri jeho implementácií.

### Zoznam kľúčových vlastností

- **Manažment identít**
  - Správa životného cyklu identít
  - Definovanie a dodržiavanie nastavených pravidiel
  - Propagácia dát do koncových systémov

  - Centrálna identita
  - Notifikácie (SMTP/SMS)
  - Serverové úlohy hromadného spracovania
- **Synchronizácia**
  - Rekonciliácia identít
  - Detekcia zmien a synchronizácia v reálnom čase
  - Korelácia dát
- **RBAC**
  - Definícia hierarchických rolí
  - Podmienené priradenie rolí
  - Katalóg rolí
- **Riadenie prístupu**
  - Propagácia oprávnení do koncových systémov
  - Definovanie vlastníka oprávnení
  - Fine-grained autorizačná politika
  - Podmienené priradenie oprávnení
  - Delegácia administratívy
- **Schvaľovanie**
  - Voliteľný počet úrovní schvalovacích procesov
  - Certifikácia oprávnení
  - Eskalácia schvaľovania
- **Užívateľské rozhranie**
  - Profil
  - Správa hesiel
  - Podpora obnovy hesla
  - Samoobsluha požiadania o priradenie roly
  - Schvaľovacie procesy
  - Zobrazenie elementov na základe oprávnení
  - Podpora lokalizácie národného jazyka (slovenčina)
  - UI zobrazenie členstva roly (oprávnenie)
- **Konektory**
  - Viac ako 60 pripravených konektorov
  - Implementácia vlastných konektorov
  - Podpora ConnId a OpenICF frameworkov

- **Bezpečnosť**
- **Rozšíriteľnosť a prispôsobenie**
  - Definícia vlastných atribútov a rozšírenie schémy
  - Klasterizácia
  - RESTful API
  - Docker/Kubernetes
  - Volanie skriptov priamo v XML objektoch
  - Vysoká miera prispôsobiteľnosti
- **Podpora**
  - Dokumentácia
  - Wiki, GitHub s príkladmi konfigurácie
  - Kniha o najlepších praktikách
  - Online stretnutia
  - Aktívny emailový zoznam

### Zoznam nedostatkov

- **Metódy jednotného prihlásenia**
- Vysoké znalostné požiadavky na administrátora

## 5.2 OpenIAM

Nástroj OpenIAM[2] od spoločnosti OpenIAM LLC je open-source IAM riešenie, ktoré poskytuje komplexné vlastnosti, vrátane manažmentu webových prístupov, riadenie identít, automatizovaný identity provisioning a správu hesiel. Ponúka komerčnú a enterprise verziu. Enterprise verzia sa predpláca a poskytuje viac funkcionalít a lepšiu podporu. Nástroj je založený na programovacom jazyku Java, ktorý ponúka multi-platformové nasadenie a využíva Spring Boot pre mikroslužby. Podporuje SQL aj NoSQL databázové systémy, ktoré hrajú dôležitú úlohu pri uspokojení potrieb klientov. Ponúka komplexnú RESTful API, ktorá pomáha integrovať veľké množstvo služieb a aplikácií tretích strán.

Identity provisioning je zameraný na propagáciu prihlasovacích údajov a ich synchronizáciu medzi rôznymi koncovými systémami. Mierne zaostáva v niektorých pokročilejších vlastnostiach provisioningu, ako je napríklad spájanie si súvislostí pri priradzovaní rolí. OpenIAM poskytuje hlbšiu funkcionalitu samoobslužného portálu, ktorý užívateľom umožňuje manažovať svoje profily, heslá a žiadať o svoje prístupy, či obnoviť heslo, čo značne znižuje zaťaženie IT personálu.

Procesy v OpenIAM dodržiavajú interne nakonfigurované predpisy a pravidlá. Nástroj sa zameriava na revíziu a certifikáciu prístupov, ktoré zabezpečujú užívateľom adekvátne prístupové oprávnenia. OpenIAM ponúka nástroj pre analýzu rizík, ktorý pomáha identifikovať možné riziká spojené s priradenými oprávneniami konkrétnej identity.

[2] https://www.openiam.com/

Pri nasadení OpenIAM je možné využiť kontajnerizáciu pomocou Dockeru, alebo Kubernetes. Nástroj sa viac zameriava na podporu natívnej funkcionality cloudových riešení, ktoré sú dobre optimalizované a to hlavne pre známych, vyhľadávaných poskytovateľov cloudu. Ďalej sa pri konfigurácií procesov zameriava na jednoduchosť a poskytuje predpripravené nastavenia, ktoré fungujú out-of-the-box pre typické prípady užitia. Pre bežnú spoločnosť, je integrácia OpenIAM dostatočne robustná, ale mierne zaostáva vo funkciách prispôsobenia. Cieľom nástroja je poskytnúť štandardný, jednoduchý a dobre fungujúci prístup pri správe identít. OpenIAM ponúka veľmi intuitívne užívateľské rozhranie. Podporuje SSO pomocou SAML, OAuth 2.0 a aj OpenID Connect, kde nástroj dokáže fungovať aj ako autentifikačný server, ktorý autentifikuje užívateľov a ich oprávnenia propaguje do koncových systémov.

Silnou stránkou OpenIAM je autentifikácia pomocou viacerých faktorov (MFA), podpora autentifikácie pomocou sociálnych sietí a samo-registrácia užívateľov. Riadenie prístupov je založené na rolách, atribútoch identity a fine-grained autorizačnej politike. Prináša množstvo konektorov a integráciu SaaS riešení.

### Podpora

OpenIAM v tomto smere značne zaostáva. Klienti evidujú dlhú reakčnú dobu odpovedí zo strany podpory. V manažmente identít sú administrátori častokrát nútený tvoriť ad-hoc riešenia a rýchlo reagovať na kritické chyby, ktoré môžu v open-source riešeniach ľahko vzniknúť. Klienti taktiež evidujú neprofesionalitu zo strany podpory a častokrát aj nekvalitné odpovede, ktoré nevedú k vyriešeniu problému. Aj napriek tomu sa OpenIAM LLC snaží vybudovať komunitu, ktorá si v prípade núdze dokáže medzi sebou pomôcť. Spoločnosť adresuje tento problém pomocou videí a tutoriálov, ktoré aktívne zverejňujú na ich YouTube kanále. Taktiež usporiadávajú online konferencie, v ktorých svojich klientov vzdelávajú o samotnom nástroji.

### Zoznam kľúčových vlastností

- **Manažment identít**
  - Správa životného cyklu identít
  - Definovanie a dodržiavanie nastavených pravidiel
  - Centrálna identita
  - Notifikácie (SMTP/SMS)
  - RBAC
  - Podporuje účty bez vlastníka
- **Synchronizácia**
  - Rekonciliácia identít
- **RBAC**
  - Definícia hierarchických rolí
  - Podmienené priradenie rolí
- **Riadenie prístupu**
  - Viac faktorová autentifikácia

  - Metódy jednotného prihlásenia
  - RBAC
  - Fine-grained autorizačná politika
  - Podmienené priradenie oprávnení
  - Delegácia administratívy
- **Schvaľovanie**
  - Konfigurovateľné schvaľovanie
  - Certifikácia oprávnení
  - Eskalácia schvaľovania
- **Užívateľské rozhranie**
  - Profil
  - Správa hesiel
  - Obnova hesla
  - Samo-registrácia užívateľov
  - Samoobsluha požiadania o priradenie oprávnení
- **Konektory**
  - Množstvo predpripravených konektorov
  - Implementácia vlastných konektorov
- **Bezpečnosť**
- **Rozšíriteľnosť a prispôsobenie**
  - RESTful API
  - Docker/Kubernetes

### Zoznam nedostatkov

- Rozdieľ vlastností komunitnej a enterprise verzie
- Enterprise verzia len pomocou predplatenia
- Menej rozšíriteľný a prispôsobiteľný
- Zlé ohlasy na podporu
- Chýba lokalizácia do národneho jazyka
- Malá komunita
- Zastaralé grafické prvky užívateľského rozhrania

## 5.3 KeyCloak

Nástroj KeyCloak[3] od spoločnosti Red Hat je open-source IAM nástroj, ktorý sa zameriava na autentifikáciu a autorizáciu. Silná stránka tohto nástroja je práve manažment prístupov. KeyCloak bol primárne programovaný v jazyku Java a využíva ďalšie techonlógie, ako napríklad JBoss WildFly aplikáčný server (taktiež založený na jazyku Java), ktorý ponúka solídny základ pre poskytovanie služieb. Nástroj zabezpečuje užívateľom reláciu, vystavuje tokeny a poskytuje metódy jednotného prihlásenia medzi veľkým množstvom aplikácií. KeyCloak nezabezpečuje identitiy provisioning do takej miery ako midPoint, alebo OpenIAM. Jeho hlavným účelom je byť autorizačným a autentifikačným serverom pre veľké množstvo aplikácií. Podporuje viac faktorovú autentifikáciu, autentifikáciu pomocou poskytovateľov identít a sociálnych sietí a zabezpečuje prihlasovací formulár, ktorý je možné graficky prispôsobiť pre potreby klienta. Vďaka federácií dokáže nástroj udržiavať stav užívateľov a synchronizovať ich dáta.

KeyCloak je známy pod dvoma verziami. Komunitnú verziu KeyCloak, ktorá je voľne dostupná a konfigurovateľná a enterprise verziu, ktorá je samostatným produktom spoločnosti Red Hat s názvom Red Hat Single-Sign-on (RH-SSO), ktorá patrí do väčšieho balíku služieb. RH-SSO verziu je možné získať predplatením balíku služieb a prináša profesionálny prístup podpory a out-of-the-box riešenie pre veľké organizácie, ktoré potrebujú zabezpečiť svoj biznis pred hrozbami a rôznymi útokmi. Tieto verzie ponúkajú rovnaké vlastnosti, rozdieľ je práve v podpore.

Hlavným komponentom nástroja KeyCloak je server, ktorý spravuje autentifikáciu, autorizáciu a vystavuje tokeny. Podporuje SAML 2.0, OAuth 2.0 a aj OpenID Connect.

Realm je základným prvkom nástroja. Každý realm je separátny menný priestor a obsahuje chránený zoznam užívateľov, ich oprávnení, rolí a skupín.

Ďalším dôležitým komponentom je administrátorská konzola vo forme webovej aplikácie, ktorej účel je poskytnúť administrátorom priehľadné užívateľské prostredie, v ktorom môžu konfigurovať jednotlivé menné priestory, poskytovateľov identít, federácie identít, tokeny a ich tvrdenia a ďalšie bezpečnostné nastavenia.

Nástroj obsahuje RESTful API, pre jednoduchšiu integráciu s externými aplikáciami. API zabezpečuje automatizáciu procesov a to hlavne synchronizáciou identít v rámci menných priestorov. Podporuje adaptéry pre množstvo aplikačných platforiem.

Riadenie prístupu je založené na RBAC, fine-grained autorizačnej politike a centralizovanom manažmente identít.

Keďže je KeyCloak zameraný na manažment prístupov, identity provisioning necháva na ostatné nástroje a väčšinou je použitý ako služba v rámci väčšieho IAM prostredia, kde sa využíva hlavne kvôli vlastnostiam jednotného prihlásenia a kvalitnej konfigurácie tokenov. Inými slovami, KeyCloak nie je taký robustný ako midPoint, alebo OpenIAM, ale kvalita a konfigurovateľnosť jeho funkcií je naozaj excelentná.

Aj napriek tomu, že administrátorská konzola je intuitívne užívateľské rozhranie, vysoká miera prispôsobenia a konfigurovateľnosť dokáže nejedného administrátora zmiasť. Vývojári na tento fakt mysleli a preto sú predvolené nastavenia jednotného prihlásenia a tokenov rozumne nastavené.

Nástroj bol navrhnutý aj pre klasterizáciu, kedy dokáže udržať veľkú trafiku a rozložiť záťaž. Ponúka kontinuálny chod a zabezpečenie relácie proti zlyhaniu pomocou replikácie dát. Ďalej poskytuje auditné logovanie a načúva pri udalostiach (prihlásenie, odhlásenie, registrácia), na ktoré dokáže reagovať.

[3] https://www.keycloak.org/

### Podpora

KeyCloak je populárne open-source riešenie s veľkou komunitou. Existuje mnoho fór a skupín, kde si klienti vymieňajú skúsenosti a znalosti. Dokumentácia je postavená veľmi dobre a obsahuje množstvo prípadov užitia a príklady riešenia problémov. Taktiež existuje GitHub repozitár, do ktorého dokážu klienti reportovať chybovosť nástroja. S predplatenou verziou enterprise prichádzajú ďalšie výhody podpory a kvalitnejšie bezpečnostné riešenia, ktoré môžu hrať kľúčovú rolu pre veľké organizácie. Ponúka profesionálny tím podpory, ktorí je ochotný riešiť chyby, alebo pomáhať pri implementácií nástroja pre špecifické potreby klienta. Red Hat ponúka tréningové kurzy pre administrátorov, ktorí implementujú tento nástroj. Nástroj je pravidelne aktualizovaný.

### Zoznam kľúčových vlastností

- **Manažment identít**
  - Viac faktorová autentifikácia
  - Definovanie rolí
  - Definovanie skupín
  - Centrálna identita
  - Auditné logovanie
  - Správa relácií v reálnom čase
- **Manažment prístupov**
  - Fine-grained autorizačná politika
  - RBAC
  - Federácia užívateľov
- **Metódy jednotného prihlásenia**
  - SAML 2.0, OAuth 2.0, OpenID Connect
  - Konfigurovatelnosť tokenov
  - Špecifikovanie tvrdení tokenov
  - Autentifikácia cez sociálne siete
  - Výmena tokenov medzi poskytovateľmi identít
  - Podpora implementácie vlastného protokolu
- **Užívateľské rozhranie**
  - Administrátorská konzola
  - Prihlasovací formulár
  - Existujúce témy a predlohy
  - Grafické prispôsobenie formulára na mieru
  - Podpora internacionalizácie (i18n)

- **Adaptéry**
  - Podpora zabezpečenia služieb
  - Podpora množstva programovacích jazykov klientskych aplikácií
- **Rozšíriteľnosť a prispôsobenie**
  - RESTful API
  - Docker/Kubernetes
  - Škálovanie
  - Vysoká miera prispôsobenia
  - Menné priestory
  - Klasterizácia
- **Bezpečnosť**
  - Zabudovaný bezpečnostný mechanizmus proti útokom
  - Ukladá informácie o zariadeniach pre neskoršie monitorovanie

### Zoznam nedostatkov

- Identity provisioning
- Životné cykly identít
- Vyššie znalostné požiadavky na administrátora

## 5.4 Porovnanie a výber aplikačnej domény

Vyššie uvedené nástroje pre správu identít sú kvalitné IAM open-source riešenia a každé z nich adresuje manažment identít iným spôsobom. Aplikačná doména tejto bakalárskej práce pozostáva z oddelenej správy identít a oddelenj správy prístupov.

Čo sa týka vyššie uvedených nástrojov, KeyCloak je výbornou voľbou práve pre oddelenú správu prístupov, v ktorej exceluje vďaka: menným priestorom, vysokej miere prispôsobenia, metódam jednotného prihlásenia, definovaniu rolí a skupín a obrovskou komunitou.

Pre oddelenú správu identít bol vybraný nástroj midPoint, ktorý vyniká v: identity provisioning, automatizovaných procesoch, extrémnej prispôsobitelnosti vďaka skritpovaniu, spracovávaní vysokého objemu dát, definovaní rolí a organizačných jednotiek a výbornej dokumentácií.

OpenIAM v tomto porovaní vyšiel ako hybrid, ktorý ponúka kvalitnú správu identít a prístupov v jednom open-source riešení, ale oproti midPoint, alebo KeyCloak, zaostáva v: miere prispôsobenia, automatizácie procesov (iných ako prístup a odchod identity), skriptovacích schopnostiach, v komunite a podpore.

# Kapitola 6

# Návrh implementácie

Hlavným cieľom výslednej architektúry je dosiahnuť generické prostredie pre manažment identít a manažment prístupov s vysokou mierou prispôsobenia a rozšírenia, pre spoločnosti s veľkým počtom zamestnancov, externistou, dodávateľov a rozsiahlou organizačnou štruktúrou. Výsoká miera prispôsobenia je pri riadení životných cyklov identít a ich prístupov kľúčová. Architektúra by mala spĺňať oddelenú správu životných cyklov identít (identity provisioning) a oddelenú správu prístupov do aplikačnej domény, ktorú bude spoločnosť svojim zamestnancom ponúkať. Vďaka spolupráci vybraných nástrojov pre správu identít vznikne centralizovaná správa identít a ich prístupov a pre každého užívateľa (zamestnanca spoločnosti) vznikne jedna federatívna identita, ktorá zabezpečí jednotné prihlásenie do všetkých aplikácií v aplikačnej doméne spoločnosti bez povšimnutia, čo je v dnešnej dobe užívateľsky žiadúce. Jednotné prihlásenie bude prebiehať pomocou protokolu OpenID Connect (v kombinácií s praktikami OAuth 2.0) s využitím autorizačného kódu (komunikačný tok s webovým prehliadačom užívateľa). Architektúra bude ďalej obsahovať dve jednoduché aplikácie pre demonštráciu komunikácie medzi komponentami architektúry a potvrdenie funkcionality centralizovaného manažmentu. Implementácia týchto dvoch aplikácií by mala a demonštrovať len dôležité časti tejto architektúry a poslúžiť ako predloha pre rozvoj ďalších klientskych aplikácií do tohto prostredia.

## 6.1 Aplikačná doména

Táto sekcia popisuje výber aplikácií do aplikačnej domény a popisuje funkciu a procesy jednotlivých aplikácií v navrhnutej architektúre.

### 6.1.1 midPoint

Pre oddelenú správu identít bol vybraný nástroj midPoint. Nástroj midPoint bude v architektúre pôsobiť ako zdroj pravdy (repozitár identít), do ktorého môžu byť identity importované, alebo manuálne vytvárané. Pre tieto identity bude nástroj prideľovať oprávnenia (pomocou rolí), zabezpečovať automatické rozdelenie užívateľov do organizačných jednotiek, pre ktoré budú určení manažéri, ďalej bude propagovať zmeny do nástroja KeyCloak. Nástroj midPoint bude obsahovať dve roly určené pre koncových užívateľov a to `End User` a `Approver`. Rola `End User` bude popisovať autorizačné oprávnenia bežného zamestnanca, ktoré budú minimálne (splňujú princíp najmenej oprávnení) a umožnia koncovým užívateľom prezerať svoj profil, žiadať o priradenie aplikačných rolí (len pre seba) a meniť svoje heslo. Rola `Approver` bude priradená manažérom organizačných jednotiek a bude posky-

tovať dodatočné autorizačné oprávnenia, ktoré manažérom umožnia prezerať informácie o svojich podriadených a schvaľovať ich žiadosti o aplikačné role (popríade priamo priradiť) pomocou midPoint modulu pre schvaľovacie procesy. Každá zmena bude propagovaná do nástroja KeyCloak pomocou `midPoint-KeyCloak` konektoru. Nástroj midPoint bude v tejto architektúre pôsobiť ako poskytovateľ identít, server prostriedkov a klient zároveň.

### 6.1.2 KeyCloak

Pre oddelenú správu prístupov a zabezpečenia jednotného prihlásenia bol vybraný nástroj KeyCloak. Nástroj KeyCloak bude obsahovať kópiu identít skrz propagáciu zo strany nástroja midPoint. Pre tieto identity bude nástroj ponúkať prihlasovací formulár, ktorý bude jeden v rámci celej architektúry. Po úspešnom prihlásení bude užívateľom v nástroji KeyCloak vytvorená relácia a zabezpečené jednotné prihlásenie pomocou protokolu OpenID Connect do všetkých aplikácií v rámci aplikačnej domény. V nástroji KeyCloak bude definovaný menný priestor s názvom `midpoint` a v ňom budú vytvorení klienti, ktorí budú obsahovať konfiguráciu jednotlivých aplikácií z aplikačnej domény. Všetci klienti budú využívať OpenID Connect a autorizačný kód pre získanie prístupu. KeyCloak poskytne užívateľom aj natívnu aplikáciu pre zobrazenie všetkých aplikácií, ktoré má koncový užívateľ v rámci aplikačnej domény k dispozícií (a dokáže k nim pristúpiť pomocou presmerovania). KeyCloak bude kontajnerizovaný pomocou aplikácie Docker Desktop[1]. Nástroj bude v tejto architektúre pôsobiť ako autentifikačný a autorizačný server a vďaka kópií identít z nástroja midPoint aj ako proxy poskytovateľ identít, ktorí bude vystavovať klientom všetky potrebné tokeny v mene koncových užívateľov.

### 6.1.3 Aplikácia demonštrácie OpenID Connect

Pre demonštráciu jednotného prihlásenia, jednotného odhlásenia a konzumáciu tokenov bude vytvorená webová aplikácia s názvom *demo-app-1*, ktorá bude komunikovať s nástrojom KeyCloak, pomocou ktorého sa koncový užívateľ (zamestnanec spoločnosti) prihlási do svojeho účtu a budú mu zobrazené jeho atribúty, ktoré budú získané pomocou tokenu identity, ktorý bude posielaný na koncový bod `userinfo` na strane nástroja KeyCloak. Táto aplikácia bude obsahovať presmerovanie do nástroja KeyCloak a to konkrétne do natívnej aplikácie, ktorá užívateľovi ukáže všetky aplikácie, ktoré má v rámci aplikačnej domény k dispozícií. Aplikácia bude v tejto architektúre pôsobiť ako klient s názvom `demo-app-1`.

### 6.1.4 Aplikácia demonštrácie midPoint API

Pre demonštráciu midPoint API bude vytvorená druhá webová aplikácia, ktorá bude postavená z aplikácie *demo-app-1* a bude nazvaná *demo-app-2*. Taktiež bude komunikovať s nástrojom KeyCloak, ktorý bude užívateľov autentifikovať, bude zobrazovať dodatočné atribúty a poskytne koncovým užívateľom možnosť požiadať si o dovolenku. Dovolenku si budú žiadať skrz webový formulár, kde vyberú dátum od kedy a dátum do kedy žiadajú o dovolenku. Aplikácia bude získavať informácie o užívateľovi cez koncový bod `userinfo`, z ktorých najzaujímavejším atribútom bude počet dní dovolenky, ktoré užívateľovi ostávajú a adekvátne na to bude môcť užívateľ požiadať o konkrétny dátum (bez prekročenia počtu dní dovolenky). Úspešné odoslanie formulára spôsobí zavolanie midPoint API, ktorá následne príjme požiadavok a vytvorí schvaľovací proces pre manažéra koncového užívateľa,

[1] https://www.docker.com/products/docker-desktop/

ktorý sa získa dynamickým hľadaním v nástroji midPoint. Koncový užívateľ bude musieť počkať na výsledok schvaľovacieho procesu a úspešnú zmenu uvidí až v samoobsluhe nástroja midPoint. Tak ako *demo-app-1*, aj táto aplikácia bude obsahovať presmerovanie do natívnej aplikácie KeyCloak pre zobrazenie všetkých dostupných aplikácií. Aplikácia bude v tejto architektúre pôsobiť ako klient s názvom `demo-app-2`.

## 6.2 Architektúra

Výsledná architektúra bude pozostávať z piatich komponentov: nástroj midPoint, nástroj KeyCloak, *demo-app-1*, *demo-app-2* a webový prehliadač koncového užívateľa.

Nástroj midPoint bude propagovať identity do nástroja KeyCloak pomocou `midPoint-KeyCloak` konektoru. Tento konektor je komunitný open-source Java konektor od spoločnosti Openstandia, ktorý je možné nájsť v GitHub repozitári[2]. Konektor ponúka funkcionalitu napojenia nástroja midPoint do menného priestoru v nástroji KeyCloak pod konkrétnym (technickým) užívateľom a tým manažovať identity v mennom priestore pomocou propagácie zmien (prepisovanie hodnôt). Verzia konektoru bola vybraná ako najnovšia podporovaná verzia `v1.1.6`. Verzia nástroja midPoint bola vybraná ako najnovšia podporovaná verzia a to `4.8`, ktorá je kompatibilná s verziou `midPoint-KeyCloak` konektoru.

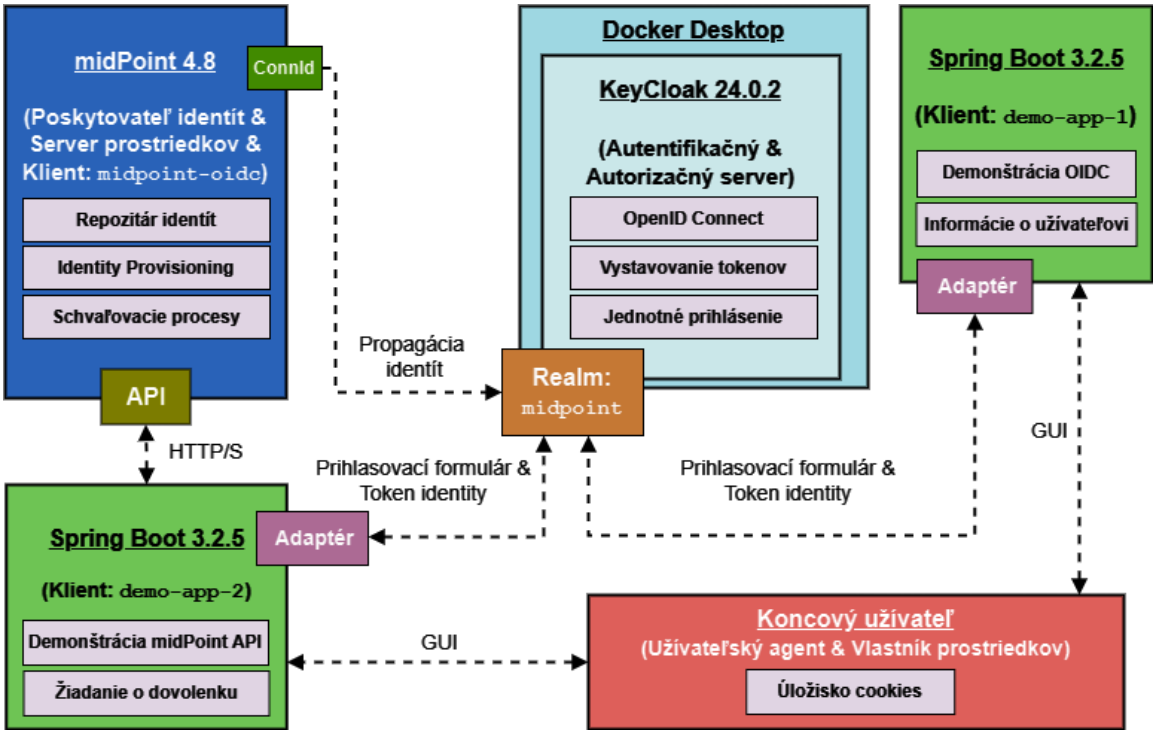
Aplikácie *demo-app-1* a *demo-app-2* budú postavené ako Spring Boot[3] webové aplikácie najnovšej verzie `3.2.5`, ktorá výborne pasuje do celej architektúry vďaka vysokej podpore adaptérov a využitia programovacieho jazyka Java, ktorý bol použitý pri postavení oboch zvolených nástrojoch pre správu identít. Oba vybrané nástroje dokonca túto klientsku aplikáciu vyslovene odporúčajú použiť.

V nástroji KeyCloak bude vytvorený samostatný menný priestor s názvom `midpoint`, do ktorého sa budú propagovať identity z nástroja midPoint, pre ktorý bude vytvorený technický užívateľ s menom `midpoint-admin` a oprávneniami, ktoré dovolia manažovať menný priestor `midpoint`. Menný priestor zabezpečí komunikáciu a reláciu medzi všetkými klientskymi aplikáciami, ktoré budú v rámci menného prostoru vytvorené. V mennom priestore `midpoint` budú vytvorení traja klienti. Pre nástroj midPoint to bude klient s menom `midpoint-oidc`. Pre *demo-app-1* to bude klient s menom `demo-app-1` a pre *demo-app-2* to bude klient s menom `demo-app-2`. Verzia nástroja KeyCloak bola vybraná ako najnovšia verzia `24.0.2`, ktorá je ešte podporovaná zvoleným `midPoint-KeyCloak` konektorom spomenutým vyššie.

Tabuľka 6.1: Aplikačná doména

| **Aplikácia** | **Verzia** | **Doména** | **Docker** |
|---|---|---|---|
| **midPoint** | midPoint 4.8 | localhost:8080 | nie |
| **KeyCloak** | KeyCloak 24.0.2 | localhost:8081 | áno |
| **Aplikácia 1** | Spring Boot 3.2.5 | localhost:8082 | nie |
| **Aplikácia 2** | Spring Boot 3.2.5 | localhost:8083 | nie |

[2] https://github.com/openstandia/connector-keycloak
[3] https://spring.io/projects/spring-boot




Obr. 6.1: Architektúra

Tabuľka 6.2: Architektúra KeyCloak klientov

| Aplikácia | Menný priestor | Meno klienta |
|---|---|---|
| **midPoint** | midpoint | midpoint-oidc |
| **KeyCloak** | master/midpoint | account-console |
| **Aplikácia 1** | midpoint | demo-app-1 |
| **Aplikácia 2** | midpoint | demo-app-2 |

Implementácia architektúry bude prebiehať na Windows 10, ďalej len hostitel. Nástroj midPoint bude stiahnutý a nainštalovaný ako *Stand-Alone*[4] verzia pre Windows. Po rozbalení stiahnutých súborov sa vytvorí pracovný adresár s názvom `midpoint-4.8`. Automaticky bude vytvorený administrátorský účet s menom `administrator` a heslom `5ecr3t`. V pracovnom adresáry sa nachádzajú dva zaujímavé adresáre. Adresár s menom `var` na ceste `midpoint-4.8/var`, ktorý sa berie ako domovský adresár nástroja midPoint (ďalej sa bude spomínať ako `${midpoint.home}`) a obsahuje logovacie súbory, konektory, import objekty a rozšírenia schémy. Ďalej adresár `bin` na ceste `midpoint-4.8/bin`, v ktorom sa nachádzajú skripty pre spustenie midPoint serveru, ktorý pobeží na hostitelovi s portom `8080` a prístup k samotnej aplikácií midPoint bude pomocou webového prehliadača na adrese `http://localhost:8080`.

Na hostitelovi bude nainštalovaná najnovšia verzia Docker Desktop, na ktorom pobeží nástroj KeyCloak. Pomocou aplikácie Docker Desktop sa stiahne obraz nástroja KeyCloak, pomocou príkazu:

```
docker pull quay.io/keycloak/keycloak:24.0.2
```

[4] https://docs.evolveum.com/midpoint/install/bare-installation/distribution/

Po stiahnutý obrazu sa vytvorí kontajner a spustí KeyCloak servis pomocou príkazu (pre lepší nákres boli použité oddeľovače riadku):

```
docker run -name KeyCloak-24.0.2
-net=bridge -p 8081:8080
-volume=/opt/keycloak/
-env=KEYCLOAK_ADMIN=admin
-env=KEYCLOAK_ADMIN_PASSWORD=admin
quay.io/keycloak/keycloak:24.0.2 start-dev
```

Nástroj KeyCloak následne pobeží v kontajneri a z príkazu je vidieť, že bude vytvorený prvotný administrátorský účet s menom `admin` a heslom `admin`. Kontajner bude odhalený na porte `8081` a pre prístup k samotnej aplikácií KeyCloak bude pomocou webového prehliadača na adrese `http://localhost:8081` (Docker vytvorí hostitelský DNS záznam).

Projekty pre obidve Spring Boot aplikácie budú inicializované pomocou Spring Initializr[5] vo verzií 3.2.5 pre programovací jazyk Java a Maven projekt s využitím Java 17 a zabaľovania do Jar súboru. Závislosti budú pre obe aplikácie rovnaké a to:

- **Spring Web**: Framework pre Spring MVC vrátane RESTful. Používa Apache Tomcat.
- **Spring Security**: Framework pre autentifikáciu a riadenie prístupu.
- **OAuth2 Client**: Integrácia OAuth2/OpenID Connect klietna.
- **Thymeleaf**: Grafické rozhranie pomocou predlôh, ktoré bežia na strane serveru.
- **Lombok**: Anotačný developerský nástroj.

Aplikácia *demo-app-1* pobeží na hostitelovi s portom `8082` a *demo-app-2* pobeží taktiež na hostitelovi s portom `8081`. Prístup ku grafickému rozhraniu aplikácií bude pomocou webového prehliadača na adresách `http://localhost:8082` a `http://localhost:8083`.

[5] https://start.spring.io/

# Kapitola 7

# Implementácia

Táto kapitola popisuje hlbší implementačný popis, konfiguráciu a obrázky rozhrania jednotlivých nástrojov a aplikácií.

## 7.1 MidPoint

Objekty v nástroji midPoint sa dajú konfigurovať pomocou grafického rozhrania cez administrátorský účet, alebo pomocou zmeny objektov, ktoré sú reprezentované pomocou XML dokumentov. Každý modifikovateľný objekt v nástroji midPoint je definovaný v XML dokumente (s príslušným unikátnym identifikátorom objektu), ktorý je podľa midPoint schémy možné detailne upraviť (tu prichádza na rad aj skriptovanie v Groovy). Nástroj midPoint po čerstvej inštalácií obsahuje preddefinované objekty: zopár rolí (`Role`), niekoľko predlôh objektov (`ObjectTemplate`) a archetypov (`Archetype`) a jedného užívateľa (`User`), ktorý predstavuje administrátorský účet. Ďalšie zaujímavé objekty sú organizácia (`Org`) a zdroj (`Resource`).

### 7.1.1 Rozšírené atribúty

Pre vytvorenie vlastných atribútov pre jednotlivé objekty bolo nutné upraviť schému v domovskom adresáry na ceste `${midpoint.home}/schema`, kde boli vytvorené rozšírené (`extension`) atribúty vo forme XSD pre úpravu XML schémy. Pre užívateľov boli vytvorené nasledujúce rozšírené atribúty:

- **managerId**: Identifikátor užívateľa, ktorý je nadriadený aktuálnemu užívateľovi.
- **isManager**: Príznak, či aktuálny užívateľ je zároveň aj manažér.
- **vacDays**: Počet dní dovolenky.

Pre roly boli vytvorené nasledujúce rozšírené atribúty:

- **roleCatalog**: Hodnota odkazujúca sa na číselník rolí typu hodnota-kľúč, pre užívateľsky prívetivé zobrazenie výberu rolí.

Pre organizácie boli vytvorené nasledujúce rozšírené atribúty:

- **approverId**: Identifikátor schvaľovateľa/manažéra organizácie.

### 7.1.2 Predlohy objektov

Ďalej boli vytvorené a upravené nové predlohy objektov pre užívateľa, rolu, a organizáciu. Tieto predlohy nastavujú počiatočné podmienky pri vytvorení nového objektu daného typu v nástroji midPoint a špecifikujú, ktoré atribúty sa majú zobraziť v grafickom rozhraní pri rozkliknutí detailu objektu daného typu, a ktoré atribúty majú byť neviditeľné.

Pre užívateľa boli nastavené počiatočné podmienky ako napríklad priradenie roly koncového užívateľa `End User`, vygenerovanie emailovej adresy, vygenerovanie prezývky, vygenerovanie prvotného hesla, nastavenie predvoleného jazyka grafického rozhrania, nastavenie prvotných počtov dní dovolenky na 30, zaradenie užívateľa do organizačnej jednotky a ďalšie.

Pre rolu boli nastavené počiatočné podmienky ako napríklad zaradenie roly do katalógu rolí, nastavenie delegovateľnosti roly a ďalšie.

Pre organizáciu boli nastavené počiatočné podmienky hlavne pre zobrazenie atribútov na objekte typu organizácia.

Tieto objekty sa dajú dohľadať cez grafické rozhranie administrátorského účtu na ľavom okraji obrazovky v menu: `Object templates -> All object templates`.

### 7.1.3 Archetypy

Ďalej boli vytvorené a upravené nové archetypy, ktoré predstavujú zoskupenie objektov do logického celku a dokážu definovať autorizačné oprávnenia. Boli vytvorené nasledujúce archetypy:

- **Employee**: Archetyp pre užívateľov, ktorý identifikuje, že sa jedná o zamestnanca so základnými oprávneniami.
- **Application role**: Archetyp pre roly, ktorý identifikuje, že sa jedná o rolu typu aplikačná (napríklad prístup do aplikácie Total Commander).
- **Line organizational structure**: Archetyp pre organizácie, ktorý identifikuje že sa jedná o hlavnú (líniovú) organizačnú jednotku.

Tieto objekty sa dajú dohľadať v menu: `Archetypes -> All archetypes`.

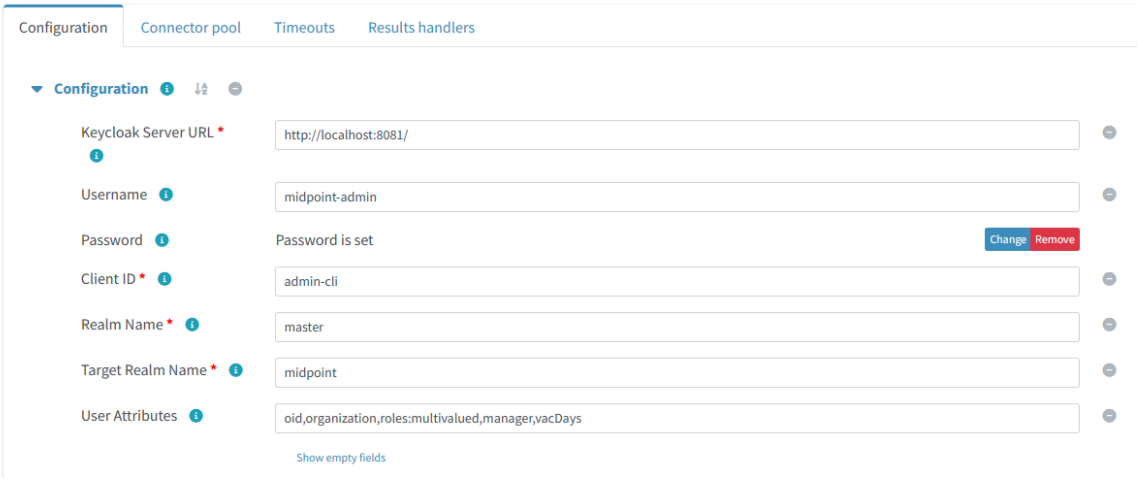
### 7.1.4 Zdroje

Boli vytvorené a upravené nové zdroje, ktoré predstavujú externé systémy (pomocou konektorov), ako napríklad databázové tabuľky, CSV súbory, LDAP konektor. Medzi tieto konektory bol pridaný aj `midPoint-KeyCloak` konektor.

Po úspešnom nastavení rozšírených atribútov, predlôh objektov a archetypov boli vytvorené tri zdroje typu CSV súbor. Každý zdroj bol upravený tak, aby mapoval objekty na riadkoch CSV súboru na objekty v nástroji midPoint a taktiež mapoval ich atribúty. Jeden z vytvorených zdrojov typu CSV je zdroj pre počiatočný import užívateľov, a ďalšie dva pre počiatočný import rolí a organizácií.

Tieto CSV súbory sa dajú dohľadať v domovskom adresáry midPoint na ceste `${midpoint.home}/import/resources` a obsahujú prvotný import dát do midPointu: okolo 50 užívateľov, niekoľko aplikačných rolí a celá organizačná štruktúra so štyrmi organizáciami, ktoré spadajú pod jednu hlavnú organizáciu. Tieto dáta boli importované pomocou úloh (`Task`), ktoré sa dajú dohľadať v menu: `Server tasks -> Import tasks`.

Ďalej bol vytvorený zdroj typu `midPoint-KeyCloak` konektor s názvom `KeyCloak Resource`. Tento zdroj zodpovedá za propagáciu užívateľov a ich atribútov do nástroja KeyCloak, konkrétne do menného priestoru `midpoint`.




Obr. 7.1: Konfigurácia midPoint-KeyCloak konektoru

Konfigurácia jednotlivých parametrov bude vysvetlená neskôr. Všetky zdroje sa dajú dohľadať v menu: `Resources -> All resources`.

### 7.1.5 Roly

Rola `End User` je preddefinovaná rola nástrojom midPoint a preto bola upravená. Užívatelia, ktorí budú mať túto rolu priradenú budú mať minimálne oprávnenia pre zobrazenie grafického rozhrania a to len pre samoobsluhu, do ktorej spadá: zobrazenie profilu, zmena hesla, požiadanie o rolu (len samému sebe).

Ďalej bola upravená preddefinovaná rola `Approver`, ktorá bude priradená užívateľom, ktorí sú manažéri. Táto rola bola upravená tak, že obsahuje autorizačné oprávnenia pre prezeranie profilov podriadených zamestnancov, prezeranie svojej organizačnej jednotky, dokáže požiadať o rolu pre svojich podriadených a tým vytvoriť schvaľovací proces, ktorý užívateľ v tejto role dokáže potvrdiť, alebo zamietnuť (alebo ani jedno).

Ďalej bola vytvorená rola s názvom `META: Approval by Manager`, ktorá sa priraďuje nie užívateľom, ale samotným rolám, nad ktorými bude vytvorený schvalovací proces (pri priradení roly, ktorá obsahuje túto `META` rolu). V tejto role bol naprogramovaný skript Groovy, ktorý dynamicky hľadá manažéra, ktorému by bol schvaľovací proces priradení. Existuje množstvo spôsobov, akými hľadať manažéra. V tejto práci bol použitý spôsob zistenia manžéra na základe organizačnej jednotky pod ktorú užívateľ (pre ktorého sa rola žiada) spadá. V iných prípadoch to nemusí byť manažér, môže to byť akýkoľvek užívateľ.

Ďale bola vytvorená rola `VACATION`, ktorú keď má užívateľ priradenú identifikuje to, že mal, má, alebo bude mať dovolenku. To či je dovolenka aktívna, alebo nie je určené aktívačným časom roly a to konkrétne hodnotami `validTo` a `validFrom`. Táto rola má priradenú vyššie spomenutú `META` rolu, čo znamená, že podlieha schvaľovaciemu procesu. O túto rolu si neskôr budú užívatelia žiadať skrz aplikáciu *demo-app-2*.

Rola `REST_API` bola vytvorená tak, aby obsahovala autorizačné oprávnenia pre technický účet, ktorý sa bude používať pri volaní midPoint API. Táto rola zabezpečuje, že užívateľ,

ktorý ma túto rolu priradenú dokáže vykonávať iba žiadosť o pridelenie roly `VACATION`, po ktorej sa vytvorí schvaľovací proces na manažéra užívateľa. Hneď na to bol vytvorený technický účet, užívateľ s menom `vacation_system`, ktorému bola rola `REST_API` pridelená.

Všetky roly je možné dohľadať v menu: `Roles -> All roles`.

### 7.1.6 Bezpečnostná politika

V bezpečnostnej politike nástroja midPoint bol pridaný modul a sekvencia autentifikácie pomocou OpenID Connect. V nastavení modulu bolo uvedené, ktorej autorite dôverovať (kto bude vystavovať tokeny) pomocou definície adresy, v tomto prípade adresy nástroja KeyCloak menného priestoru `midpoint` v XML elemente:

```
.
.
.
<oidc>
  <client>
    <clientId>midpoint-oidc</clientId>
    <clientSecret>XXXX</clientSecret>
    <nameOfUsernameAttribute>preferred_username</nameOfUsernameAttribute>
    <openIdProvider>
      <issuerUri>http://localhost:8081/realms/midpoint</issuerUri>
    </openIdProvider>
  </client>
</oidc>
.
.
.
```

Ďalej bolo nutné definovať identifikátor klienta, ktorý bol nastavený na `midpoint-oidc` a klientske heslo, ktoré bolo vygenerované v nástroji KeyCloak (popísané neskôr) pri vytvorení klienta `midpoint-oidc`.

Sekvencia bola pridaná ako predvolená sekvencie, takže pri prístupe do nástroja midPoint na adrese `http://localhost:8080`, bude komunikácia presmerovaná na KeyCloak prihlasovací formulár na adrese `http://localhost:8081/*`.

Pre pôvodné prihlásenie cez midPoint (mimo KeyCloak), napríklad pre administrátora, ktorý nebude do KeyCloak menného priestoru propagovaný, bola nastavená dedikovaná adresa prihlásenia do nástroja midPoint na adrese `http://localhost:8080/midpoint/auth/gui-default`. Bezpečnostná politika a jej definícia v XML formáte sa dá dohľadať v menu: `Repository objects -> All objects -> Type:SecurityPolicy -> Basic search -> Default Security Policy`.

### 7.1.7 Odchytenie

V systémovej konfigurácií nástroja midPoint bolo vytvorené odchytenie (`Hook`). Toto odchytenie, vo forme skritpu Groovy, zabezpečí, že po úspešnom potvrdení schvaľovacieho procesu priradenia roly `VACATION` (na užívateľovi archetypu `Employee`) bude užívateľovi odčítaný počet dní dovolenky (`extension` atribút `vacDays`) na základe atribútov `validTo`

a `validFrom`, ktoré identifikujú od kedy a do kedy je rola aktívna. Ďalej toto odchytenie vyvolá prepočítanie identity a následnú propagáciu zmeny do nástroja KeyCloak.

Odchytenie sa dá dohľadať v menu: `Repository objects -> All objects -> SystemConfiguration -> (XML element) -> modelHooks -> (XML element) hook`.

## 7.2 KeyCloak

Po štarte nástroja KeyCloak je preddefinovaný jeden hlavný menný priestor s názvom `master`. V tomto mennom priestore bol vytvorený ďalší menný priestor s názvom `midpoint`. Hneď po vytvorení menného priestoru `midpoint` bol v mennom priestore `master` vytvorený klient s názvom `midpoint-realm`. Tento klient obsahuje roly, ktoré definujú autorizačné oprávnenia nad manažmentom celého menného priestoru.

Následne bol v mennom priestore `master` vytvorený nový užívateľ s menom `midpoint-admin`, ktorému bolo vytvorené heslo `"midpoint"` a priradené role pre manažment menného priestoru `midpoint`:

- **midpoint-realm** `manage-users`: manažovanie užívateľov.
- **midpoint-realm** `view-users`: prezeranie užívateľov.
- **midpoint-realm** `view-realm`: prezeranie menného priestoru.
- **midpoint-realm** `query-users`: vyhľadávanie užívateľov.
- **midpoint-realm** `query-groups`: vyhľadávanie skupín.

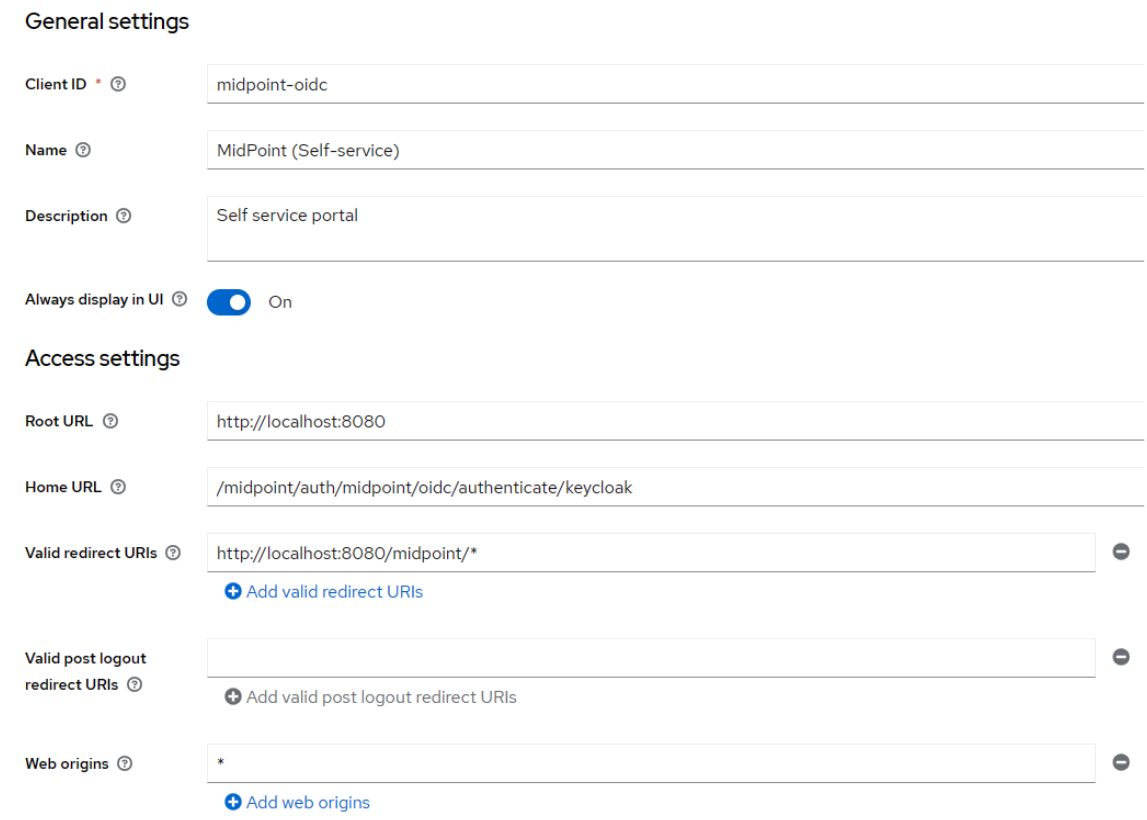
Tento užívateľ bol použitý v konfigurácií `midPoint-KeyCloak` konektoru (obrázok 7.1), v ktorom sa tento užívateľ pripája na klientsku KeyCloak aplikáciu `admin-cli` v mennom priestore `master`. Konektor zabezpečí volanie príkazov v tejto administrátorskej konzole tak, aby dokázal propagovať všetkých užívateľov z nástroja midPoint do nástroja KeyCloak, konkrétne do menného priestoru `midpoint`.

### 7.2.1 Menný priestor midpoint

V mennom priestore `midpoint` sa nachádzajú preddefinované klientske aplikácie, medzi ktorými sa nachádza `account-console`. Táto aplikácia zobrazuje personálne informácie o užívateľovi v KeyCloak a obsahuje preddefinované zobrazenie všetkých klientskych aplikácií, ktoré sú užívateľovi v rámci menného priestoru k dispozícií. Na túto aplikáciu sa budú *demo-app-1* a *demo-app-2* odkazovať možným presmerovaním v grafickom rozhraní oboch aplikácií (pre pohodlný prechod medzi aplikáciami v rámci federácie).

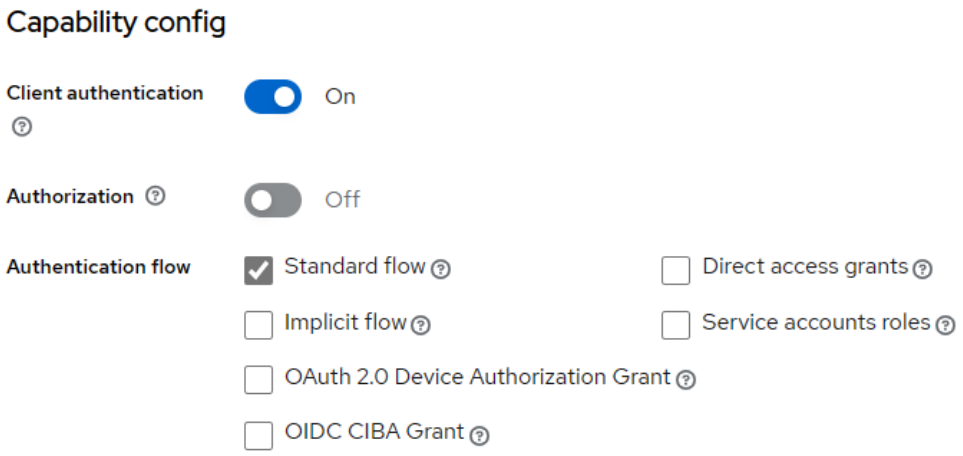
### 7.2.2 Klient midpoint-oidc

V mennom priestore bol vytvorený OpenID Connect klient s menom `midpoint-oidc`, ktorý predstavuje klientskú aplikáciu nástroja midPoint. Na tohto klienta sa odkazuje autentifikačná sekvencia v objekte bezpečnostnej politiky nástroja midPoint v podsekcií 7.1.6, kde viaže autentifikovaného užívateľa pomocou tvrdenia v tokene identity:
`preffered_username`, kde hodnota tohto tvrdenia je prezývka užívateľa pod ktorou sa prihlasuje.



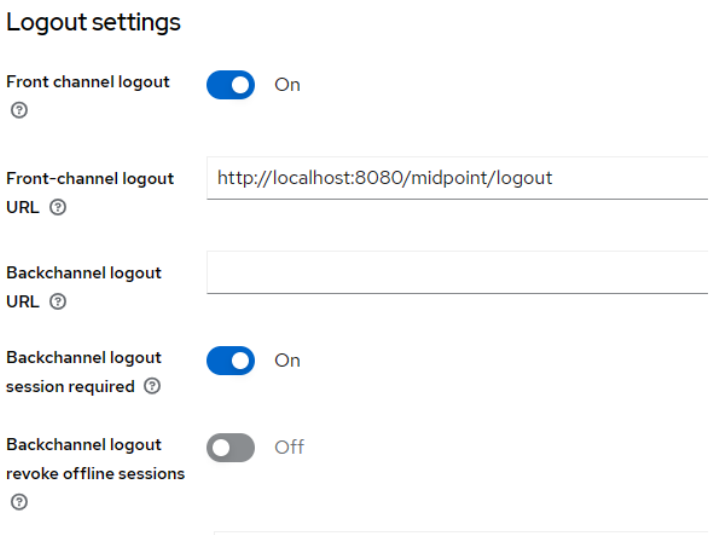

Obr. 7.2: Konfigurácia adries klienta midpoint-oidc.

- **Root URL**: adresa, na ktorej sa nachádza klientská aplikácia midPoint. Od tejto adresy sa ďalšie adresy dokážu konfigurovať relatívne.
- **Home URL**: adresa koncového bodu klientskej aplikácie, kde prebieha autentifikácia pomocou OpenID Connect.
- **Valid redirect URIs**: adresa koncového bodu klientskej aplikácie, na ktorú sa má po úspešnej autentifikácií presmerovať naspäť (* znamená akákoľvek).




Obr. 7.3: Konfigurácia OpenID Connect klienta midpoint-oidc.

- **Client authentication**: príznak, či sa musí klientska aplikácia autentifikovať. Ak áno, klientovi bude v menu: `Credentials` vytvorené heslo.
- **Authentication flow**: Typ získania overenia. Standard flow predstavuje autorizáciu pomocou autorizačného kódu (obrázok 4.2).

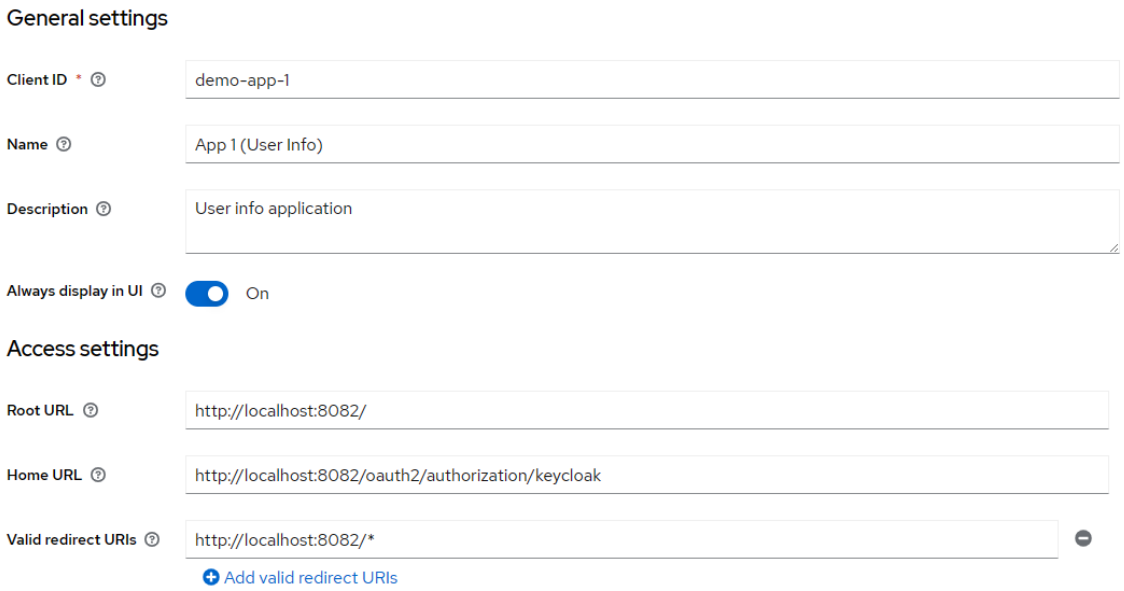



Obr. 7.4: Konfigurácia jednotného odhlásenia klienta midpoint-oidc.

- **Front channel logout**: príznak, ktorý povoľuje jednotné odhlásenie zo všetkých klientskych aplikácií, v ktorých je užívateľ prihlásený.
- **Front-channel logout URL**: adresa koncového bodu klientskej aplikácie, na ktorej sa užívatelia odhlasujú.

### 7.2.3 Klient demo-app-1

V mennom priestore bol vytvorený OpenID Connect klient s menom `demo-app-1`, ktorý predstavuje klientskú aplikáciu *demo-app-1*.




Obr. 7.5: Konfigurácia adries klienta demo-app-1.

Konfigurácia OpenID Connect bola nastavená rovnaká ako na obrázku 7.3. Konfigurácia jednotného odhlásenia bola nastavená rovnaká ako na obrázku 7.4, len s použitou adresou: `http://localhost:8082/logout`.

Ďalej boli pre tohto klienta namapované atribúty: `oid`, `organization`, `roles` (možné vidieť na obrázku 7.1 v parametry `User Attributes`, ktoré `midPoint-KeyCloak` konektor číta z midPointu a propaguje do KeyCloak) na tvrdenia tokenu identity, ktoré sa budú posielať do *demo-app-1* pre zobrazenie grafického rozhrania.

### 7.2.4 Klient demo-app-2

V mennom priestore bol vytvorený OpenID Connect klient s menom `demo-app-2`, ktorý predstavuje klientskú aplikáciu *demo-app-2*.

Konfigurácia tohto klienta bola nastavená totožne s konfiguráciou `demo-app-1`, jediný rozdieľ je v adrese, kde namiesto `http://localhost:8082` bola použitá adresa `http://localhost:8083`.

Atribúty, ktoré boli namapované do tvrdení tokenu identity sú v prípade tohto klienta: `oid` a `vacDays`. Tieto dva atribúty sú pre *demo-app-2* dôležité pre midPoint API. Atribút `oid` predstavuje unikátny identifikátor XML dokumentu užívateľa, ktorý bude v aplikácií prihlásený.

## 7.3 Klientská aplikácia demo-app-1

V aplikácií *demo-app-1*, po inicializácií Spring Boot projektu, bola vytvorená adresárová štruktúra s koreňovým adresárom `keycloak1`. Závislosť `Spring Web` definuje MVC architektúru, takže boli vytvorené nasledujúce balíky: `Config`, `Controller`, `Handler`, `Service`. Balíky obsahujú jednotlivé Java triedy. Pre grafické rozhranie boli v adresárovej štruktúre na ceste `keycloak1/src/main/resources` vytvorené nasledujúce balíky: `static` a `templates`. V balíku `templates` sa nachádzajú grafické rozhrania v HTML kóde a v balíku `static/css` sa nachádzajú grafické štýly.

V balíku `Config` bola vytvorená trieda Java s názvom `SecurityConfig`, ktorá bola nastavená pomocou Spring anotácií na: `@Configuration`, `@EnableWebSecurity`, `@EnableMethodSecurity` a `@RequiredArgsConstructor`. Prvý jedináčik bol vytvorený ako objekt `SecurityFilterChain`, ktorý nastavuje reťaz zabezpečenia. Bol naprogramovaný, aby príjmal len autentifikované požiadavky pomocou OpenID Connect a následne manažoval a vytváral relácie (pokiaľ to je nutné). Ďalej tu boli nastavené predvolené adresy pre prihlásenie a odhlásenie. Každá požiadavka smerovaná na túto aplikáciu prejde reťazou zabezpečenia pre filtráciu nežiadúcich požiadavkov.

Druhý jedináčik bol vytvorený ako objekt `LogoutSuccessHandler`, ktorý príjma požiadavky o odhlásenie aktuálne prihláseného užívateľa. V jeho logike získava token identity aktuálne prihláseného užívateľa a následne posiela tento token na koncový bod odhlásenia, ktorý sa nachádza v nástroji KeyCloak. Po zaslaní tokenu identity bude užívateľ odhlásený zo všetkých relácií, ktoré sú v nástroji KeyCloak pod týmto užívateľom aktívne (zabezpečenie jednotného odhlásenia). Adresa koncového bodu odhlásenia bola nastavená na: `http://localhost:8082/logout`.

V balíku `Handler` bola vytvorená trieda Java s názvom `SuccessfulAuthHandler`, ktorá po úspešnej autentifikácií užívateľa zabezpečí uloženie tokenu identity, tokenu obnovy a prístupového tokenu. Po úspešnom uložení tokenov užívateľa bude užívateľ presmerovaný na domovské grafické rozhranie (s tokenami).

V balíku `Service` bola vytvorená trieda Java s názvom `UserInfoService`. Táto trieda je anotovaná ako `@Service` a teda tiež ako jedináčik, v ktorom bola naprogramovaná funkcia `fetchUserInfo(String accessToken)`, ktorá pri zavolaní príjma prístupový token. Tento token následne posiela na koncový bod `userinfo`, ktorý sa nachádza v nástroji KeyCloak. Po zaslaní prístupového tokenu na tento koncový bod budú z nástroja KeyCloak obdržané detailné informácie o identite užívateľa vo forme tvrdení, ktoré sa namapujú do `Map<String, Object>` a vrátia na výstupe funkcie.

V balíku `Controller` boli vytvorené dva kontroléri. Kontrolér s názvom `RootController`, ktorý zabezpečuje presmerovanie koreňového koncového bodu na koncový bod domovského grafického rozhrania. Kontrolér s názvom `HomeController`, ktorý príjma prístupový token, ktorý pomocou spomenutej funkcie `fetchUserInfo` vymieňa prístupový token za detailné informácie o identite a tie ďalej posiela do grafického rozhrania (`Thymeleaf` predlohy) s názvom `home.html`, ktorá sa nachádza v balíku `templates`. Ak pri získavaní detailných informácií o identite nastane chyba, presmeruje užívateľa do grafického rozhrania chyby s návzom `error.html`.

Grafické rozhranie `home.html` obsahuje mapovanie atribútov z kontroléru `HomeController`, ktoré sú užívateľovi zobrazené vo webovom prehliadači. Toto grafické rozhranie taktiež obsahuje tlačidlo pre presmerovanie do klienta `account-console`, ktoré presmeruje užívateľa do zobrazenia všetkých aplikácií, ktoré má k dispozícií.

## 7.4 Klientská aplikácia demo-app-2

Klientská aplikácia *demo-app-2* má rovnaké rozloženie adresárovej štruktúry a značná časť kódu bola skopírovaná z *demo-app-1*. Hlavný rozdiel je v kontroléri `HomeController`, v ktorom sa zmenili atribúty, ktoré sa užívateľovi vypíšu v grafickom rozhraní. Táto aplikácia aktívne pracuje s tvrdením `oid` a `vacDays`. V grafickom rozhraní `home.html` pribudol POST formulár, ktorý po vybraní začínajúceho dátumu a koncového dátumu dovolenky pošle POST požiadavok na nový koncový bod definovaný v kontroléri `SubmitVacationController`.

V adresárovej štruktúre na adrese `keycloak1/src/main/resources/static` pribudol nový balík s názvom `js`, ktorý obsahuje skript JavaScript s názvom `script.js`. Tento skript zabezpečuje počiatočné nastavenie kalendárov, pre výber dátumu od (HTML element s identifikátorom `startDate`) a dátumu do (HTML element s identifikátorom `endDate`). Počiatočný dátum `startDate` bol nastavený na `aktuálny_dátum + 7 dní`, aby užívateľ nemohol žiadať o dovolenku skôr, ako sedem dní vopred. Počiatočný dátum `endDate` bol nastavený na ten istý dátum ako `startDate`. Minimálna hodnota dátumu `endDate` bola nastavená tak, aby užívateľ nemohol žiadať pred dátumom `startDate`. Ďalej sa v grafickom rozhraní zobrazí počet dní dovolenky, ktoré má užívateľ k dispozícií a na základe toho skript reaguje. Maximálna hodnota dátumu `endDate` je nastavená tak, aby od dátumu `startDate` neprekročila počet dní dovolenky. Vždy keď sa zmení dátum `startDate`, adekvátne sa upraví minimálna a maximálna hodnota dátumu `endDate`.

Následne, ak je zvolený dátum validný, pošle sa POST požiadavok do kontroléru `SubmitVacationController`. Tento kontrolér sa stára o zaslanie požiadavky na koncový bod midPoint API. Metóda požiadavku, ktorá sa posiela na midPoint API je vždy typu `PATCH`, ktorý identifikuje aktualizáciu dát nad midPoint objektom. Tu sa vytvorí JSON objekt, ktorý v sebe obsahuje hodnoty objketu `itemDelta` definovanej v nástroji midPoint, ktorá zabezpečí zmenu požadovaných atribútov midPoint objektu. Po vytvorení JSON objektu, sa tento objekt pripojí k `PATCH` požiadavke. Požiadavok sa autentifikuje pomocou technického účtu `vacation_system` v nástroji midPoint, ktorý má priradené minimálne opravánie pomocou roly `REST_API` a pripojí sa jeho heslo. Ďalej je tento požiadavok poslaný na koncový bod midPoint API: `http://localhost:8080/midpoint/ws/rest/users/`, ku ktorému sa na koniec adresy pripojí tvrdenie `oid` identifikujúce, pre ktorého užívateľa sa rola `VACATION` žiada (hodnota `oid` je vždy identifikátor midPoint objektu užívateľa, ktorý je aktuálne prihlásený do aplikácie).

# Kapitola 8

# Testovanie

Testovanie sa skladá z niekoľkých testovacích prípadov. Každý testovací prípad je typu integračný test, ktorý testuje integráciu medzi jednotlivými aplikáciami z aplikačnej domény. Niektoré testovacie prípady budú obsahovať aj systémový pod-test pre overenie správnej funkcionality propagácie dát medzi jednotlivými aplikáciami. Každý testovací prípad bude obsahovať cieľ (čo je predmetom testovania) a taktiež prerekvizity potrebné pre začatie testovania.

## 8.1 Integračný test nástrojov KeyCloak a midPoint

**Prerekvizity:**

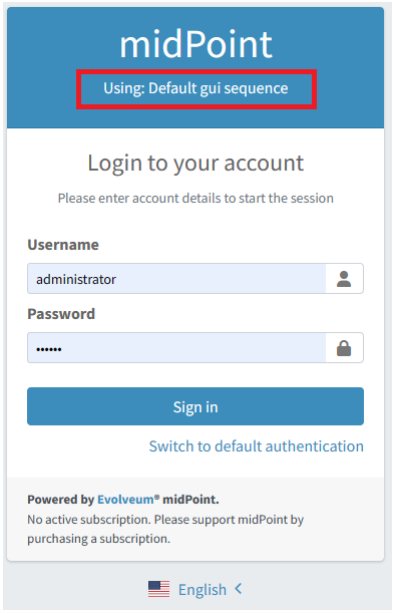
- Nástroj midPoint je spustený a počúva na adrese `http://localhost:8080`.
- V nástroji midPoint sú importované počiatočné objekty užívateľov, rolí a organizácií.
- Nástroj midPoint je pripojený pomocou `midPoint-KeyCloak` konektoru do nástroja KeyCloak.
- Nástroj KeyCloak je spustený a počúva na adrese `http://localhost:8081`.
- V nástroji KeyCloak je:
  - vytvorený menný priestor `midpoint`.
  - vytvorený užívateľ `midpoint-admin` s oprávneniami pre správu `midpoint`.

**Cieľ:**

Cieľom testovania je propagácia užívateľov z nástroja midPoint do nástroja KeyCloak menného priestoru `midpoint`. Medzi týmito nástrojmi by mal vzniknúť jednotný zoznam užívateľov s atribútmi, potrebnými pre ďalšie testovanie Spring Boot aplikácií. Užívatelia by mali mať zhodné hodnoty atribútov.

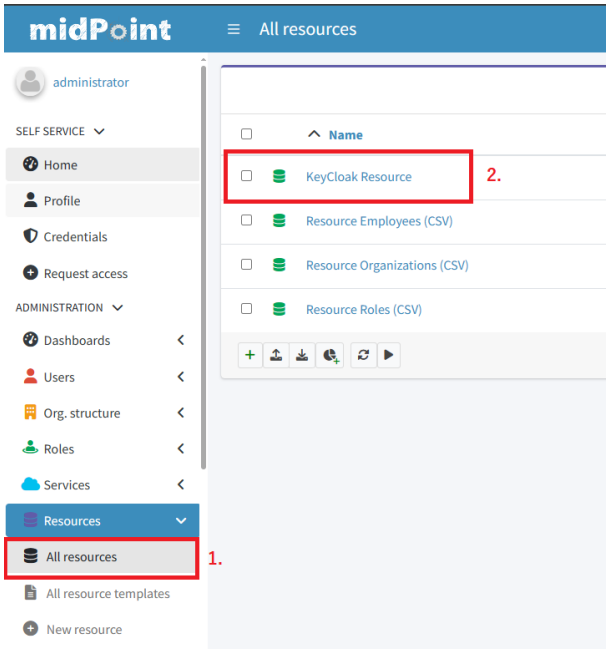
**Testovanie:**

1. Prihlásenie pod užívateľom `administrator` cez `http://localhost:8080/midpoint-/auth/gui-default` s heslom `5ecr3t`.



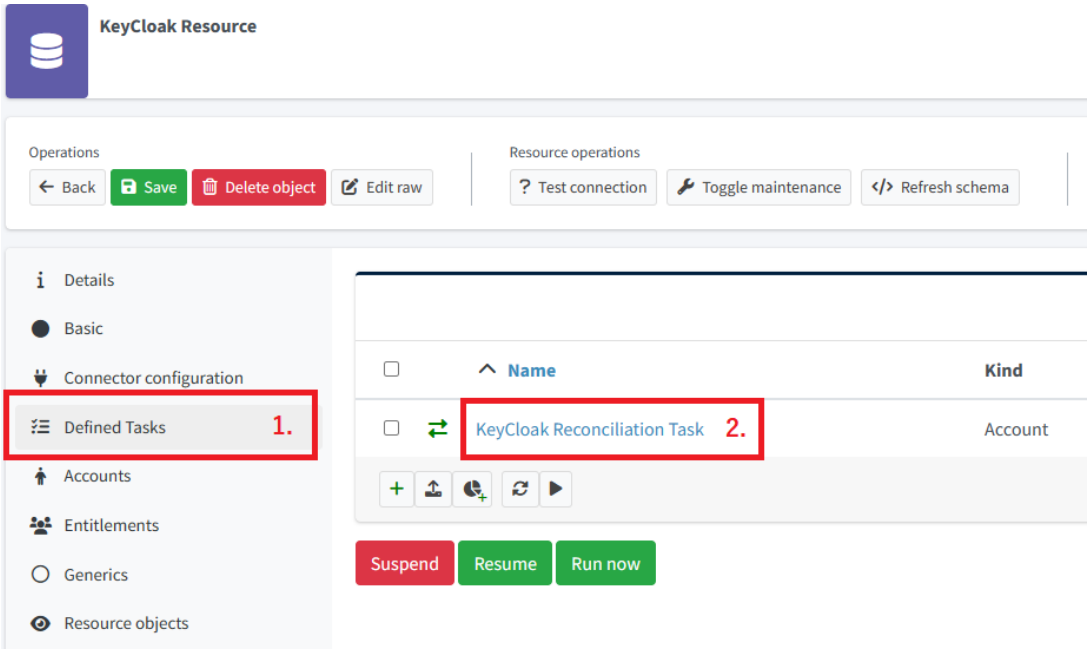

Obr. 8.1: Prihlasovací formulár midPoint

2. Prechod do zdroju `KeyCloak Resource`.



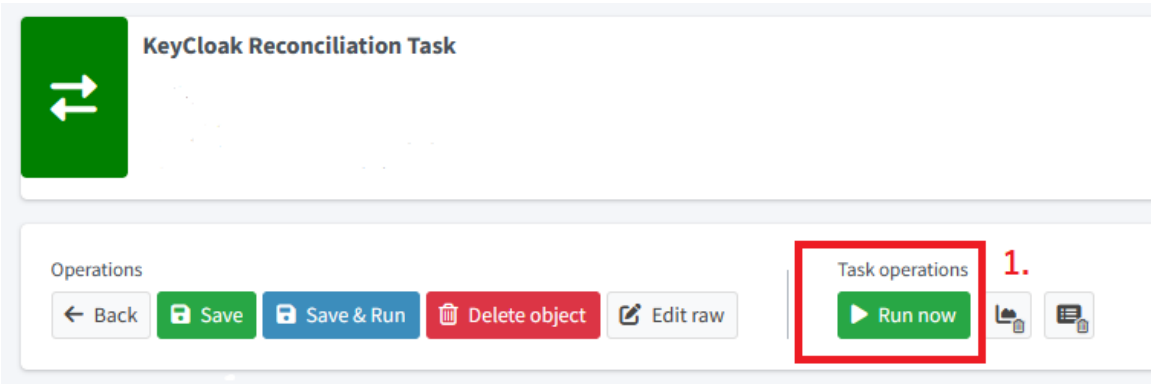

Obr. 8.2: Prechod do zdroju KeyCloak Resource

3. Prechod na preddefinovanú úlohu prepočtu, ktorá prepočíta všetkých prepojených užívateľov s nástrojom KeyCloak (implicitne všetci užívatelia v nástroji midPoint, typu `Employee`).



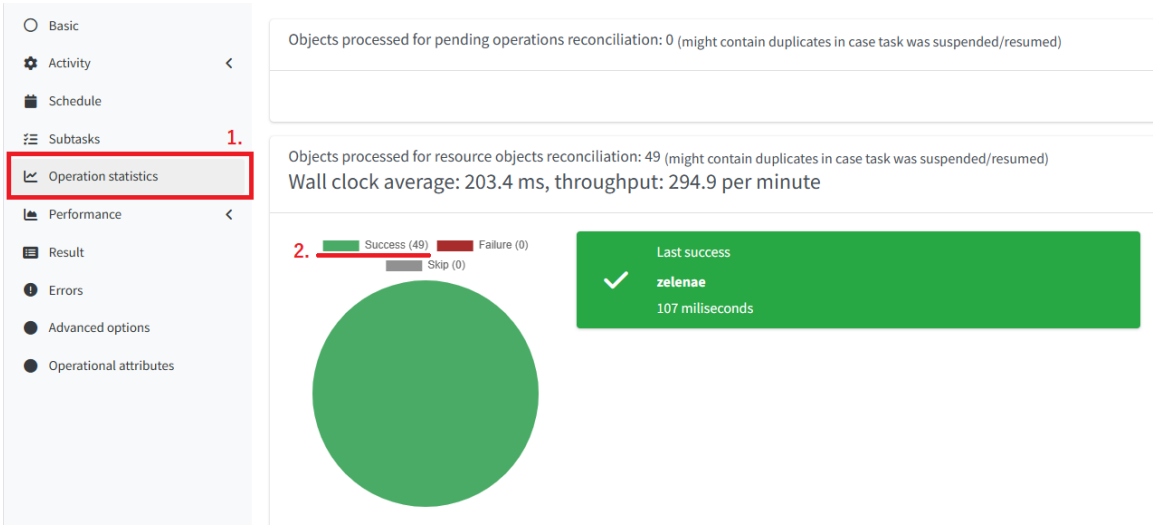

Obr. 8.3: Prechod do úlohy prepočtu

4. Spustenie úlohy prepočtu.



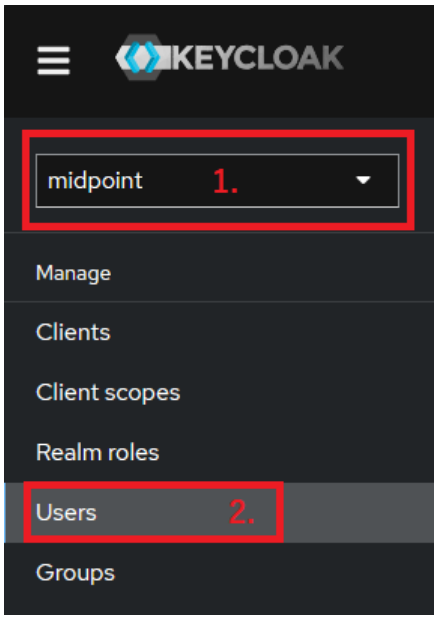

Obr. 8.4: Spustenie úlohy prepočtu

5. Zobrazenie štatistiky po dokončení úlohy prepočtu. Implicitne sa v nástroji midPoint nachádza 49 zamestnancov.



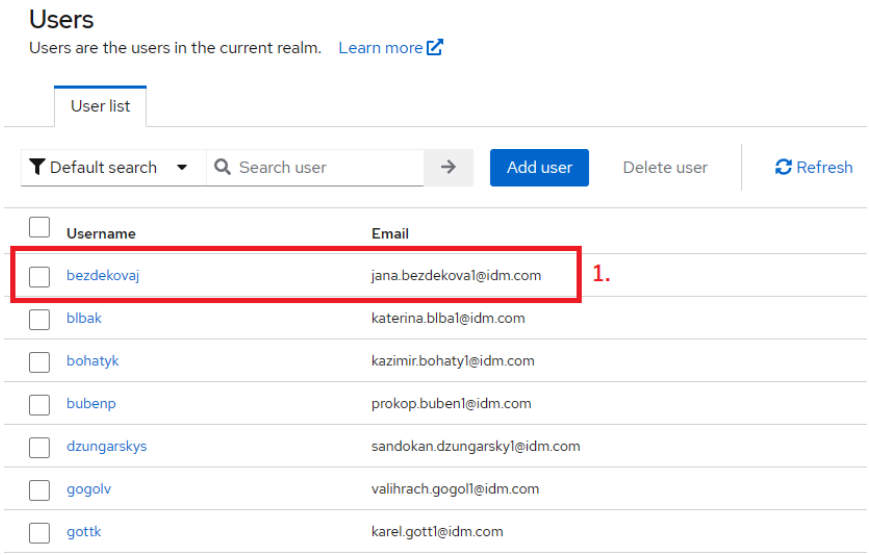

Obr. 8.5: Zobrazenie štatistiky úlohy prepočtu

6. Prihlásenie do KeyCloak pod užívateľom `admin` cez `http://localhost:8081` s heslom `admin`. Následne sa vyberie menný priestor `midpoint` a prejde sa do záložky `Users`.



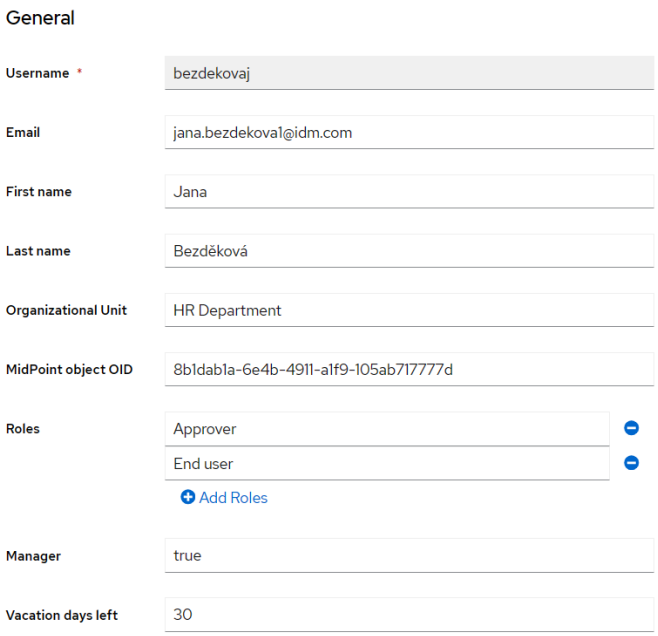

Obr. 8.6: Prechod k užívateľom v mennom priestore midpoint nástroja KeyCloak

7. Zobrazenie užívateľov v mennom priestore `midpoint`. Propagácia prebehla úspešne, pretože všetky účty v tomto zobrazení existujú. Následne sa vyberie prvý užívateľ a otvoria sa jeho detaily.

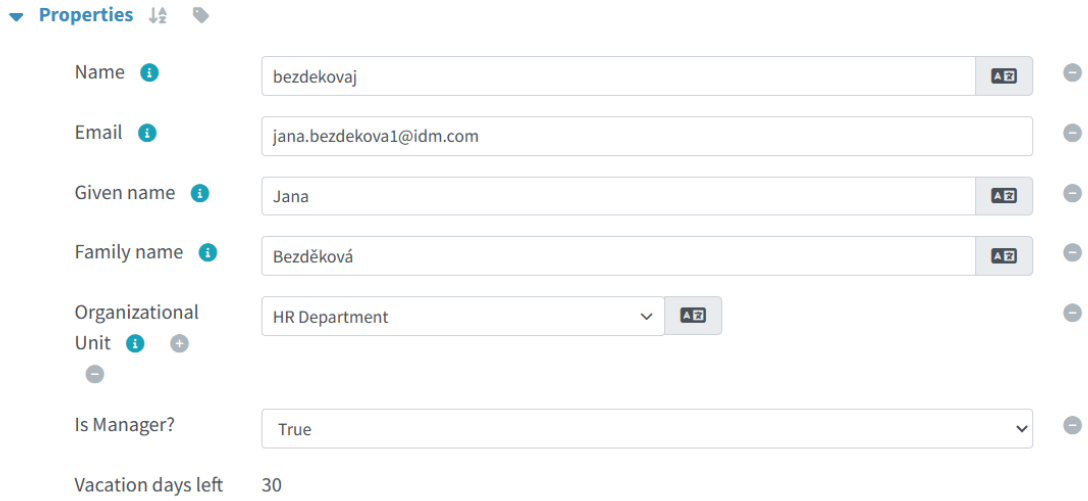



Obr. 8.7: Zobrazenie užívateľov v mennom priestore midpoint nástroja KeyCloak

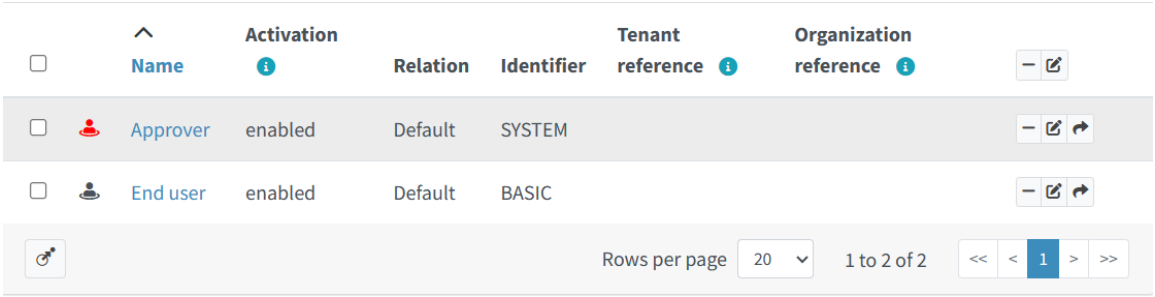
8. Po zobrazení detailu vybraného užívateľa v nástroji KeyCloak sa tieto detaily porovnajú s nástrojom midPoint. V nástroji midPoint sa v hlavnom menu vyberie záložka `Users -> All users`, nájde sa užívateľ `bezdekovaj` a porovnajú sa detaily.




Obr. 8.8: Detaily užívateľa bezdekovaj v mennom priestore midpoint nástroja KeyCloak




Obr. 8.9: Detaily užívateľa bezdekovaj v nástroji midPoint




Obr. 8.10: Detaily priradených rolí užívateľa bezdekovaj v nástroji midPoint

### Výsledok testovania:

Užívatelia z nástroja midPoint sa úspešne vypropagovali do menného priestoru `midpoint` nástroja KeyCloak a atribúty jednotlivých užívateľov sú medzi nástrojmi jednotné. Tento test zároveň potvrdil funkčnú konfiguráciu `midPoint-KeyCloak` konektoru.

## 8.2 Integračný a systémový test demo-app-1

### Prerekvizity:

- Nástroj KeyCloak je spustený a počúva na adrese `http://localhost:8081`.
- V nástroji KeyCloak sú vypropagovaný užívatelia s príslušnými atribútmi.
- Aplikácia *demo-app-1* je spustená a počúva na adrese `http://localhost:8082`.

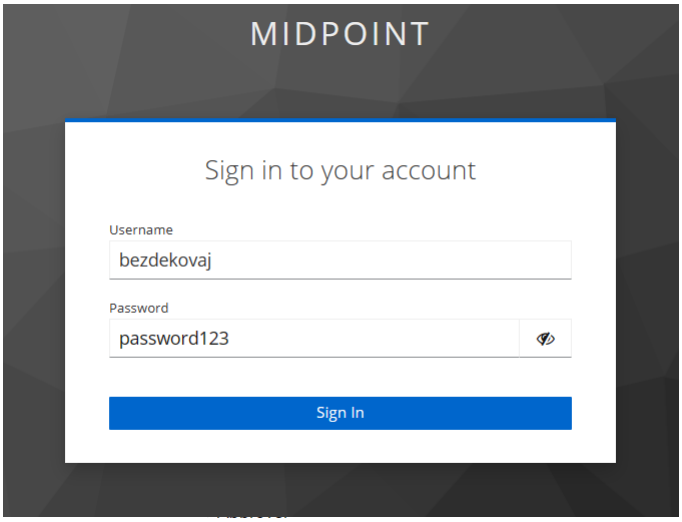
### Cieľ:

Cieľom testovania je overiť funkcionalitu autentifikácie do aplikácie *demo-app-1* pomocou prihlasovacieho formulára nástroju KeyCloak. Po úspešnej autentifikácií by mal byť webový prehliadač presmerovaný naspäť do aplikácie *demo-app-1*, kde bude užívateľovi zobrazené grafické rozhranie s jeho atribútmi. Funkčným zobrazením sa potvrdí funkcionalita konzumovania tokenov implementovaná v *demo-app-1*.

### Testovanie:

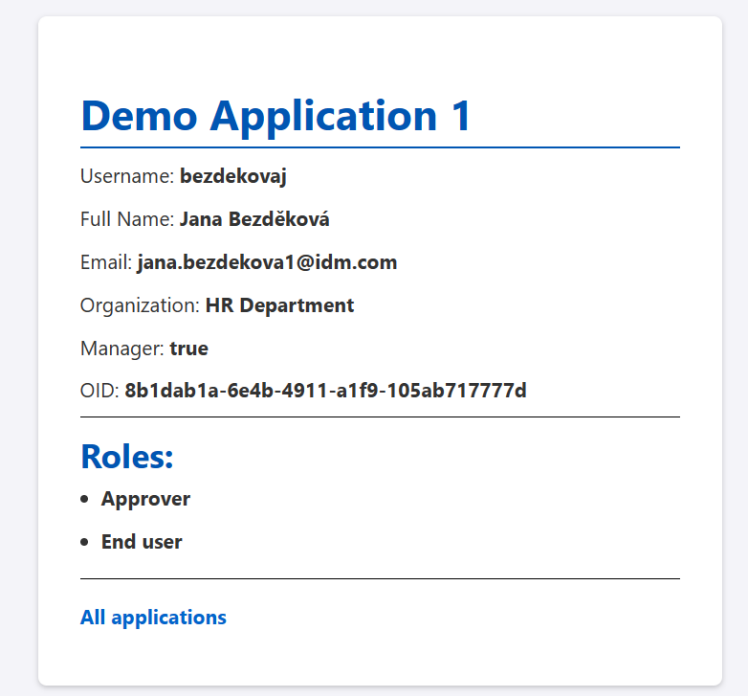
Testovací užívateľ bude užívateľ `bezdekovaj`, ktorému bolo implicitne vygenerované heslo `password123`.

1. Prihlásenie užívateľa `bezdekovaj` do aplikácie *demo-app-1* na adrese `http://localhost:8082`. Užívateľ by mal byť presmerovaný do prihlasovacieho formulára v nástroji KeyCloak na adrese `http://localhost:8081/realms/midpoint/protocol/openid-connect/*`.



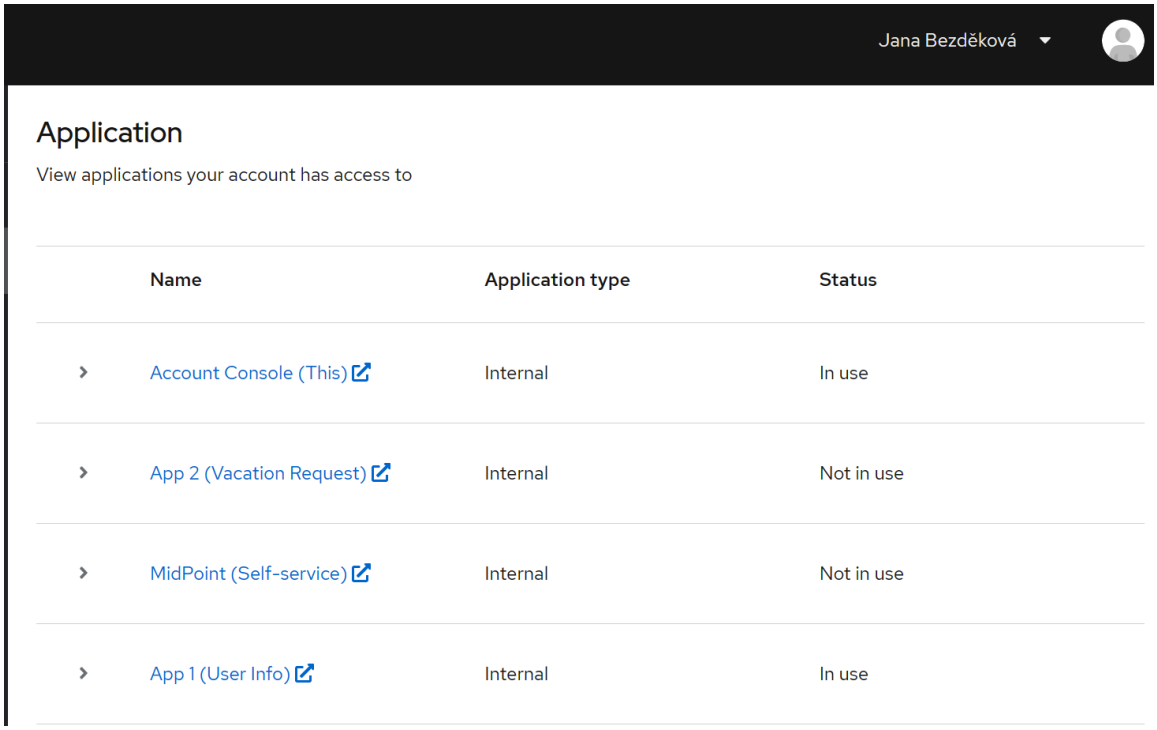

Obr. 8.11: Presmerovanie užívateľa z demo-app-1 do prihlasovacieho formulára v nástroji KeyCloak

2. Po úspešnom prihlásení užívateľa by mal webový prehliadač presmerovať užívateľa naspäť do aplikácie *demo-app-1*, kde sa mu zobrazí grafické rozhranie aplikácie. V grafickom rozhraní by mali byť vypísané hodnoty jednotlivých atribútov, ktoré boli získané pomocou konzumácie tokenu identity v implementácií aplikácie *demo-app-1*.




Obr. 8.12: Grafické rozhranie aplikácie demo-app-1

3. Užívateľ klikne na tlačidlo `All applications`. Keďže sa jedná o úplne cudziu aplikáciu, užívateľ by za normálnych okolností musel absolvovať prihlásenie znovu, ale vďaka tomu, že tieto aplikácie patria do jedného menného priestoru užívateľ by mal byť presmerovaný do tejto aplikácie a mali by sa mu zobraziť všetky jeho dostupné aplikácie bez prihlásenia a bez povšimnutia (zafungovanie SSO).




Obr. 8.13: Grafické rozhranie aplikácie account-console/all-applications

### Výsledok testovania:

Integračný test medzi aplikáciou *demo-app-1* a nástrojom KeyCloak prebehol úspešne. Webový prehliadač presmeroval užívateľa do prihlasovacieho formulára v nástroji KeyCloak a po úspešnej autentifikácií sa potvrdil aj systémový test aplikácie *demo-app-1*, ktorá skonzumovala token identity, vďaka ktorému boli užívateľovi zobrazné detaily o jeho identite. Taktiež bola vytvorená relácia medzi aplikáciami menného priestoru `midpoint` a užívateľ sa dokázal bez ďalšieho prihlasovania presmerovať do aplikácie `account-console/all-applications` bez povšimnutia (zafungovanie SSO).

## 8.3 Integračný a systémový test demo-app-2

### Prerekvizity:

- Nástroj midPoint je spustený a počúva na adrese `http://localhost:8080`.
- Nástroj KeyCloak je spustený a počúva na adrese `http://localhost:8081`.
- V nástroji KeyCloak sú vypropagovaný užívatelia s príslušnými atribútmi a je zaručená konzistencia dát užívateľov medzi nástrojmi midPoint a KeyCloak.
- Aplikácia *demo-app-2* je spustená a počúva na adrese `http://localhost:8082`.
- V aplikácií *demo-app-2* je už prihlásený užívateľ `bohatyk`, ktorý sa rovnako ako v predchádzajúcom testovaní autentifikoval pomocou prihlasovacieho formulára v nástroji KeyCloak.

- V nástroji midPoint je už prihlásený užívateľ `bezdekovaj`, ktorý je zároveň manažérom užívateľa `bohatyk` a bude čakať na vytvorenie schvaľovacieho procesu o priradenie dovolenky.

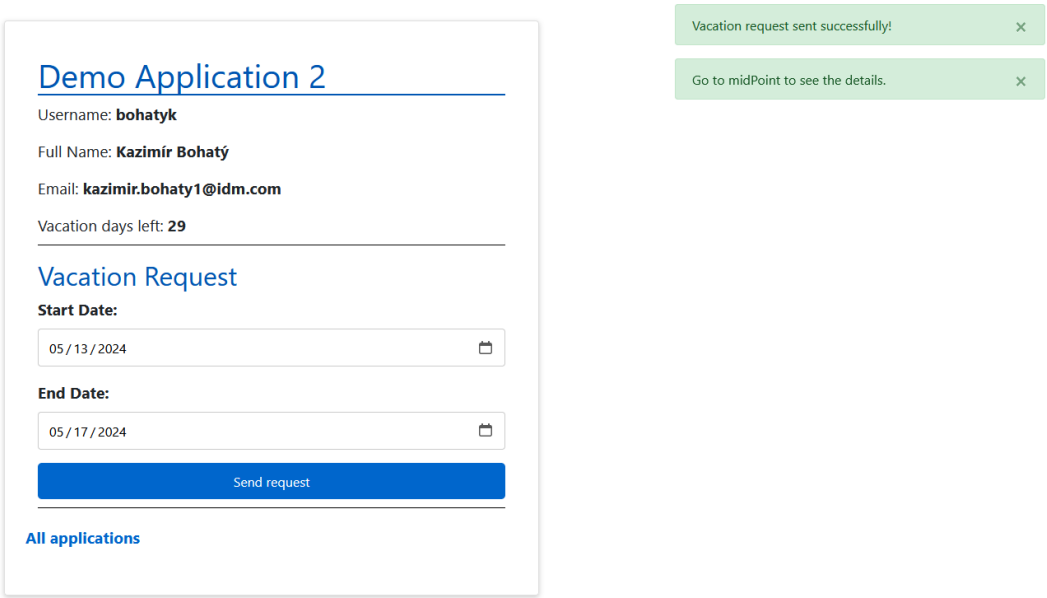
### Cieľ:

Cieľom testovania je overiť funkcionalitu aplikácie *demo-app-2*, zároveň overiť integráciu medzi nástrojom KeyCloak a nástrojom midPoint. Prihlásený užívateľ by mal požiadať o dovolenku pomocou grafického rozhrania v aplikácií *demo-app-2*, po ktorej sa ukáže informácia o prebehnutom proces (či bola žiadosť o dovolenku poslaná na manažéra, alebo zlyhala). Manažér užívateľa by mal čakať v nástroji midPoint v záložke `Cases`, kde by sa mal (po úspešnom požiadaní) vytvoriť schvaľovací proces, pre priradenie dovolenky užívateľovi `bohatyk`. Manažér by mal následne potvrdiť schvaľovací proces a užívateľovi `bohatyk` by mali byť adekvátne odčítané voľné dni dovolenky a priradená rola `VACATION`. Užívateľ `bohatyk` by mal kliknúť na tlačidlo `All applications` v grafickom rozhraní aplikácie *demo-app-2*, po ktorom by mal byť presmerovaný bez povšimnutia do aplikácie `account-console/all-applications` a z tade by sa mal prekliknúť do nástroja midPoint pomocou tlačidla `MidPoint (Self-service)`. Následne by mal byť užívateľ presmerovaný bez povšimnutia a nutnej autentifikácie do svojho profilu v nástroji midPoint, v ktorom by mal medzi svojimi rolami uvidieť priradenú rolu `VACATION`.

### Testovanie:

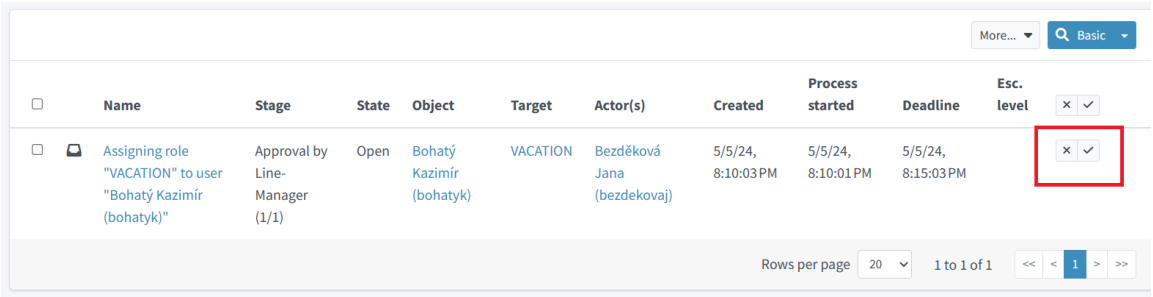
Ani jeden z užívateľov nemá aktívnu reláciu v rámci menného priestoru `midpoint`.

1. Užívateľ `bohatyk` sa úspešne prihlásil do grafického rozhrania aplikácie *demo-app-2*, v ktorej mu bol vypísaný atribút počtu voľných dní dovolenky (29). Užívateľ vybral štartovací dátum dovolenky na `13.05.2024` (vrátane) a koncový dátum `17.05.2024`, čo dokopy tvorí 5 dní a zaklikol tlačidlo `Send request`. Grafické rozhranie zobrazilo notifikáciu o úspešnom poslaní žiadosti.



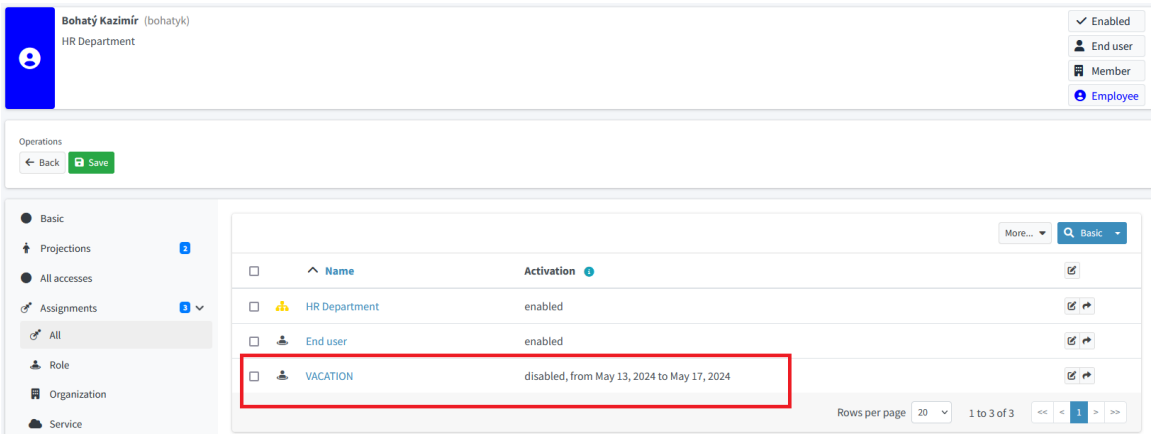

Obr. 8.14: Úspešná žiadosť o dovolenku

2. Užívateľ `bezdekovaj` obdržal žiadosť o dovolenku v nástroji midPoint v záložke `Cases`, na ktorej je vidieť, že sa žiada pre užívateľa `bohatyk`, že sa žiada rola `VACATION` a čas kedy bol požiadavok vytvorený. Po rozkliknutý schvaľovacieho procesu je možné vidieť detaily dátumov o ktoré užívateľ zažiadal. Následne tento schvaľovací proces potvrdí a užívateľovi `bohatyk` sa priradí rola `VACATION`.



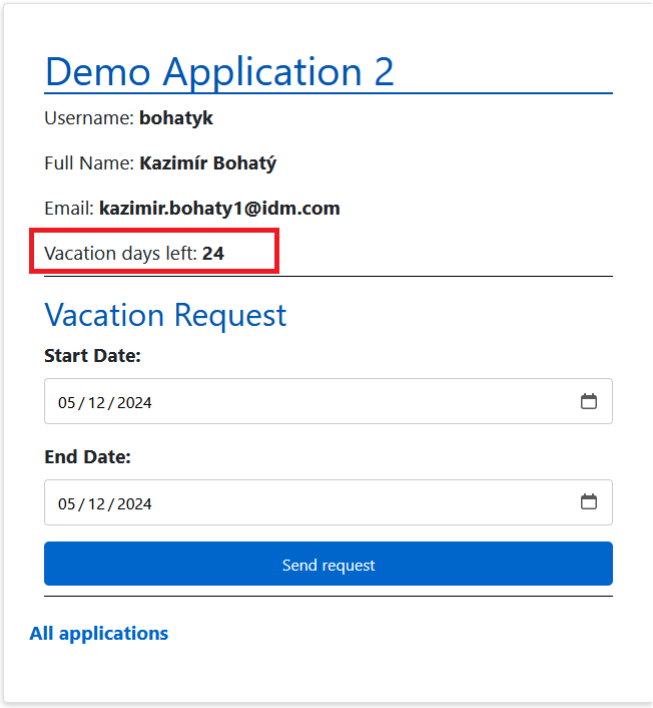

Obr. 8.15: Schvaľovací proces žiadania dovolenky pre užívateľa bohatyk

3. Užívateľ `bohatyk` sa medzi časom presmeroval do nástroja midPoint, kde si otvoril svoje priradené roly. Vidí, že mu bola rola `VACATION` priradená a zatiaľ je neaktívna. Aktívna nastane až v štartovací deň.




Obr. 8.16: Overenie schválenia roly

4. Užívateľ `bohatyk` sa presunul naspäť do aplikácie *demo-app-2*, kde uvidel, že mu bolo odčítaných 5 dní z voľných dní dovolenky.




Obr. 8.17: Odčítanie dní dovolenky

### Výsledok testovania:

Integračný test aj systémový test prebehol úspešne. Užívateľ `bohatyk` zažiadal o dovolenku, kde zafungovala implementácia *demo-app-2*, ktorá úspešne odoslala požiadavok na midPoint API a následne bol vytvorený schvaľovací proces, ktorý bol priradený manažérovi `bezdekovaj`. Manažér schválil žiadosť, a až následne na to, bola užívateľovi `bohatyk` rola `VACATION` priradená. Až po priradení roly boli užívateľovi `bohatyk` odčítané voľné dni dovolenky a keď sa užívateľ vrátil do grafického rozhrania aplikácie *demo-app-2* uvidel propagáciu zmeny jeho voľných dní dovolenky z nástroja midPoint do nástroja KeyCloak a nakoniec aj do aplikácie *demo-app-2*. Užívateľ sa po celý čas úspešne presmerovával medzi aplikáciami a vďaka SSO sa musel prihlásiť iba raz a to na začiatku pred začatím testovania.

## 8.4 Zhodnotenie výsledkov

Systémové testovanie aplikácií preukázalo očakávané správanie a správne fungovanie výmeny prihlasovacích údajov užívateľa za token identity, ktorý bol úspešne skonzumovaný a poskytol údaje o identite do grafických rozhraní demonštračných aplikácií. Integračné testy potvrdili správnu komunikáciu medzi jednotlivými aplikáciami a nástrojmi, vďaka ktorej boli úspešne vytvorené relácie a jednotné prihlásenie do všetkých aplikácií bez povšimnutia pomocou protokolou OpenID Connect a OAuth 2.0. Tým bola zabezpečená taktiež centrálna identita, ktorej informácie boli jednotné naprieč jednotlivými aplikáciami a nástrojmi.

# Kapitola 9

# Záver

Cieľom práce bolo naštudovať a porovnať vybrané nástroje pre správu identít v informačných systémoch, demonštrovať ich funkcionalitu v centralizovanej správe identít pomocou návrhu a implementácie dvoch jednoduchých aplikácií. Pre hlbšie pochopenie aktuálnych trendov a praktík v manažmente identít, boli vysvetlené dôležité pojmy a protokoly, ktoré popisujú výmenu informácií o identitách v centralizovanej správe identít. Implementačná časť tejto práce mala predstaviť generické prostredie oddelenej správy identít a oddelenej správy prístupov týchto identít, ktoré je vysoko prispôsobiteľné a ľahko rozšíriteľné. Hlavnou cieľovou skupinou, ktorá by mohla takéto prostredie v budúcnosti prispôsobiť a nasadiť, je určite spoločnosť s veľkým počtom zamestnancov, externistov, dodávateľov a rozsiahlou organizačnou štruktúrou, ktorá potrebuje riadiť životné cykly svojich zamestnancov automatizovane a riadiť ich prístupy bezpečne a užívateľsky prívetivo.

Pre mňa najzaujímavejšia znalosť, ktorú som sa počas tejto práce naučil bola teória tokenov a ich komunikačný tok, ktorý zabezpečuje bezpečnú a rýchlu výmenu prihlasovacích údajov za zašifrovaný a kompaktný kus kódu, ktorý dokáže konzumovať veľké množstvo informačných systémov a tým vytvoriť príjemné prostredie bez veľkého množstva autentifikačných procesov medzi užívateľom a webovými aplikáciami.

Čo sa týka konfigurácie jednotlivých nástrojov, bol nástroj midPoint veľkou výzvou. Nástroj je stále v aktívnom vývoji a občas sa stalo, že očakávaná funkcionalita nemala správanie, ktoré by sa za normálnych okolností očakávalo. Najväčším sklamaním pre mňa bol fakt, že nástroj neponúka sekvenciu autorizácie pre API pomocou konzumovania prístupových tokenov, táto funkcionalita je aktuálne v rozvoji.

Nástroj KeyCloak bol na druhú stranu veľkým prekvapením, pretože to na čo sa nástroj zameriava je naozaj dobre vyladené. Z tohto nástroja som mal pocit vysokého výkonu a v jeho grafickom prostredí sa pracovalo veľmi pohodlne.

Do budúcna určite stojí za vyskúšanie obohatiť demo-app-2 o viac bezpečnú komunikáciu s midPoint API, samozrejme keď bude nástroj podporovať konzumáciu prístupového tokenu a bude adekvátne odpovedať na HTTP požiadavky. Ďalší rozvoj by určite mohol byť kontajnerizácia jednotlivých aplikácií do Docker Desktop s použitím HTTPS. Vďaka prispôsobivosti nástroja midPoint určite stojí za zváženie vytvorenie ďalších užitočných aplikácií, ktoré by rozšírili arzenál tohto nástroja a uľahčili personalistom prácu pomocou samoobsluhy, alebo automatizácie ďalších procesov zamestnancov a skĺbiť ich s modulmi nástroja midPoint, ktoré v tejto práci nestihli byť otestované, ako napríklad notifikácie, reporty, auditné záznamy a certifikácia oprávnení.

# Literatúra


[1] *Centralized vs Decentralized Identity Management* [online]. Ping Identity [cit. 2024-04-05]. Dostupné z: https://www.pingidentity.com/en/resources/identity-fundamentals/identity-and-access-management/centralized-decentralized-identity-management.html.

[2] *The importance of centralized identity management* [online]. Okta [cit. 2024-04-05]. Dostupné z: https://www.okta.com/identity-101/the-importance-of-centralized-identity-management.

[3] *SAP Cloud Identity Services - Identity Provisioning* [online]. SAP [cit. 2024-03-27]. Dostupné z: https://help.sap.com/docs/identity-provisioning/identity-provisioning/what-is-identity-provisioning.

[4] *What Is Access Control? | Microsoft Security* [online]. Microsoft [cit. 2024-03-25]. Dostupné z: https://www.microsoft.com/en-au/security/business/security-101/what-is-access-control.

[5] *What is an identity provider (IdP)?* [online]. Cloudflare [cit. 2024-03-31]. Dostupné z: https://www.cloudflare.com/learning/access-management/what-is-an-identity-provider.

[6] *What is Authorization?* [online]. Okta [cit. 2024-03-25]. Dostupné z: https://auth0.com/intro-to-iam/what-is-authorization.

[7] *What is identity and access management (IAM)?* [online]. Cloudflare [cit. 2024-03-25]. Dostupné z: https://www.cloudflare.com/learning/access-management/what-is-identity-and-access-management.

[8] *What is identity lifecycle management with Microsoft entra ID? - microsoft entra ID governance* [online]. Microsoft [cit. 2024-04-07]. Dostupné z: https://learn.microsoft.com/en-us/entra/id-governance/what-is-identity-lifecycle-management#what-is-a-digital-identity.

[9] Identity management; concepts, technologies, and systems. *Reference & Research Book News*. Portland: Ringgold, Inc. 2011, zv. 26, č. 2. ISSN 0887-3763.

[10] *Access Token: Definition, Architecture, Usage & More* [online]. Okta, 14. Feb 2023 [cit. 2024-04-16]. Dostupné z: https://www.okta.com/identity-101/access-token.

[11] *What is the difference between authentication and authorization?* [online]. SailPoint Technologies, Inc., Mar 2023 [cit. 2024-02-29]. Dostupné z: https://www.sailpoint.com/identity-library/difference-between-authentication-and-authorization.

[12] *What Is Token-Based Authentication?* [online]. Okta, 28. Feb 2024 [cit. 2024-04-15]. Dostupné z: https://www.okta.com/identity-101/what-is-token-based-authentication.

[13] Alaca, F. a Oorschot, P. C. Comparative Analysis and Framework Evaluating Web Single Sign-on Systems. *ACM computing surveys*. Baltimore: Association for Computing Machinery. 2020, zv. 53, č. 5, s. 1–34. ISSN 0360-0300.

[14] Bertocci, V. *JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens* [RFC 9068]. RFC Editor, oct 2021. DOI: 10.17487/RFC9068. Dostupné z: https://www.rfc-editor.org/info/rfc9068.

[15] Campbell, B., Mortimore, C. a Jones, M. B. *Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants* [RFC 7522]. RFC Editor, may 2015. DOI: 10.17487/RFC7522. Dostupné z: https://www.rfc-editor.org/info/rfc7522.

[16] Chiarelli, A. *ID Token and Access Token: What's the Difference?* [online]. auth0, 28. Oct 2021 [cit. 2024-04-16]. Dostupné z: https://auth0.com/blog/id-token-access-token-what-is-the-difference.

[17] Ethelbert, O., Moghaddam, F. F., Wieder, P. a Yahyapour, R. A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications. *ArXiv.org*. Ithaca: Cornell University Library, arXiv.org. 2017. ISSN 2331-8422.

[18] Ferdous, M. S. a Poet, R. Managing dynamic identity federations using Security Assertion Markup Language. *Journal of theoretical and applied electronic commerce research*. CURICO: Univ Talca, Fac Ingenieria. 2015, zv. 10, č. 2, s. 53–76. ISSN 0718-1876.

[19] Ferraiolo, D. F. a Kuhn, D. R. Role-Based Access Controls. *ArXiv.org*. Ithaca: Cornell University Library, arXiv.org. 2009. ISSN 2331-8422.

[20] Grassi, P. A., Garcia, M. E. a Fenton, J. L. Digital identity guidelines. *NIST special publication*. 2017, zv. 800, s. iv.

[21] GREENBERG, E. A. a LONG, C. O. What are cookies? *Nursing (Jenkintown, Pa.)*. Philadelphia: Lippincott Williams & Wilkins, Inc. 2003, zv. 33, č. 6, s. 76–76. ISSN 0360-4039.

[22] Gupta, D. *Cookie-based vs. Cookieless Authentication: What's the Future?* [online]. LoginRadius Inc. [cit. 2024-04-15]. Dostupné z: https://www.loginradius.com/blog/engineering/cookie-based-vs-cookieless-authentication.

[23] Hardt, D. *The OAuth 2.0 Authorization Framework* [RFC 6749]. RFC Editor, oct 2012. DOI: 10.17487/RFC6749. Dostupné z: https://www.rfc-editor.org/info/rfc6749.

[24] Hendrickson, L. *Centralized vs. Decentralized Identity Management* [online]. Identity, 4. Mar 2024 [cit. 2024-04-05]. Dostupné z: https://www.identity.com/centralized-vs-decentralized-identity-management.

[25] Jones, M. B., Bradley, J. a Sakimura, N. *JSON Web Token (JWT)* [RFC 7519]. RFC Editor, May 2015. DOI: 10.17487/RFC7519. Dostupné z: https://www.rfc-editor.org/info/rfc7519.

[26] Jones, M. B. a Hardt, D. *The OAuth 2.0 Authorization Framework: Bearer Token Usage* [RFC 6750]. RFC Editor, oct 2012. DOI: 10.17487/RFC6750. Dostupné z: https://www.rfc-editor.org/info/rfc6750.

[27] Ma, X., Li, R., Lu, Z., Lu, J. a Dong, M. Specifying and enforcing the principle of least privilege in role-based access control. *Concurrency and computation*. Chichester, UK: John Wiley & Sons, Ltd. 2011, zv. 23, č. 12, s. 1313–1331. ISSN 1532-0626.

[28] Magnusson, A. *Token-based Authentication: Everything You Need to Know* [online]. StrongDM, 08. Apr 2024 [cit. 2024-04-15]. Dostupné z: https://www.strongdm.com/blog/token-based-authentication.

[29] Mohammed, I. A. CLOUD IDENTITY AND ACCESS MANAGEMENT - A MODEL PROPOSAL. *SSRN Electronic Journal*. Október 2019, zv. 6, s. 1–8.

[30] Nielsen, H., Mogul, J., Masinter, L. M., Fielding, R. T., Gettys, J. et al. *Hypertext Transfer Protocol – HTTP/1.1* [RFC 2616]. RFC Editor, jun 1999. DOI: 10.17487/RFC2616. Dostupné z: https://www.rfc-editor.org/info/rfc2616.

[31] O'Gorman, L. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*. New York, NY: IEEE. 2003, zv. 91, č. 12, s. 2021. ISSN 0018-9219.

[32] Pandey, P. a Nisha, T. N. Challenges in Single Sign-On. *Journal of Physics: Conference Series*. Bristol: IOP Publishing. 2021, zv. 1964, č. 4, s. 42016. ISSN 1742-6588.

[33] Peyrott, S. *What is single sign-on authentication (SSO) and how does it work?* [online]. Auth0, Mar 2023 [cit. 2024-04-07]. Dostupné z: https://auth0.com/blog/what-is-and-how-does-single-sign-on-work.

[34] Rouse, M. *Service Provider* [online]. Techopedia, 25. Jan 2017 [cit. 2024-03-31]. Dostupné z: https://www.techopedia.com/definition/22021/service-provider.

[35] Sakimura, N., Bradley, J., Jones, M., Medeiros, B. de a Mortimore, C. *OpenID Connect Core 1.0* [online]. openid.net, 25. Feb 2014 [cit. 2024-04-20]. Dostupné z: https://openid.net/specs/openid-connect-core-1_0-final.html.

[36] Sandhu, R. a Samarati, P. Authentication, access control, and audit. *ACM computing surveys*. Baltimore: ACM. 1996, zv. 28, č. 1, s. 241–243. ISSN 0360-0300.

[37] Sandhu, R. a Samarati, P. Access control: principle and practice. *IEEE communications magazine*. IEEE. 1994, zv. 32, č. 9, s. 40–48. ISSN 0163-6804.

[38] Schwarz, L. *What Is Authorization? Definition and Examples* [online]. Oracle NetSuite, 29. Apr 2022 [cit. 2024-03-25]. Dostupné z: https://www.netsuite.com/portal/resource/articles/erp/authorization.shtml.

[39] Sharma, A., Sharma, S. a Dave, M. Identity and access management- a comprehensive study. In: *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*. IEEE, 2015, s. 1481–1485. DOI: 10.1109/ICGCIoT.2015.7380701. ISBN 9781467379106.

[40] Shoemaker, P. *What is distributed Ledger Technology (DLT)?* [online]. Identity, 08. Mar 2024 [cit. 2024-04-05]. Dostupné z: https://www.identity.com/distributed-ledger-technology-dlt-guide.

[41] Sood, S. K. Cookie-Based Virtual Password Authentication Protocol. *Information security journal.* Taylor & Francis Group. 2011, zv. 20, č. 2, s. 100–111. ISSN 1939-3555.

[42] Witharana, H. *Identity provisioning* [online]. Medium, 20. Dec 2019 [cit. 2024-03-27]. Dostupné z: https://medium.com/identity-beyond-borders/identity-provisioning-dd1a8814573e.