Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Návrh a implementace aplikace pro propojení rozdílných API databází

Bc. Jiří Novotný

Diplomová práce
2024

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jiří Novotný**
Osobní číslo:	**I20211**
Studijní program:	**N0613A140007 Informační technologie**
Téma práce:	**Návrh a implementace aplikace pro propojení rozdílných API databází**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem diplomové práce je implementace propojení databází za účelem synchronizace dat mezi nimi pomocí REST API. Díky tomu bude možné data automaticky synchronizovat ze zdrojové databáze do ostatních databází bez nutnosti "point to point" propojení každé databáze zvlášť na zdrojovou databázi.

V teoretické části práce bude provedena analýza již existujících řešení, popsání technologie Active Directory, protokolu LDAP, rozhraní API a možnosti virtualizace.

Praktická část práce bude obsahovat návrh a implementaci aplikace pro propojení databází. Autentizace a autorizace k databázím a k webovému rozhraní propojení budou řešena pomocí protokolu LDAP.

Rozsah pracovní zprávy: **cca 60 stran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

CARTER, G. LDAP system administration. Cambridge: O´Reilly, 2003. ISBN 9781565924918.
MASSE, Mark. REST API Design Rulebook. Cambridge: O'Reilly Media, 2011. ISBN 9781449310509.
WALLS, Craig. Spring in Action. 5. New York: Manning, 2018. ISBN 9781617294945.

Vedoucí diplomové práce: **Ing. Jan Merta, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **8. listopadu 2023**
Termín odevzdání diplomové práce: **17. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

prof. Ing. Antonín Kavička, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 30. listopadu 2023

Prohlašuji:

Práci s názvem Návrh a implementace aplikace pro propojení rozdílných API databází jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Praze dne 13. 05. 2024

Bc. Jiří Novotný

PODĚKOVÁNÍ

Chtěl bych poděkovat Ing. Janu Mertovi, Ph. D. za rady a čas, který mi věnoval během vypracování této diplomové práce. Dále bych chtěl poděkovat rodině a všem, kteří mě podporovali při tvorbě této práce a během celého studia.

ANOTACE

Cílem diplomové práce je implementace propojení databází za účelem jejich automatické synchronizace pomocí API.

Teoretická část diplomové práce seznamuje s rozhraním API a s již existujícími řešeními integrace mezi databázemi. Dále teoretická část diplomové práce seznamuje s možností virtualizace, technologií Active Directory a protokolem LDAP.

Praktickou část diplomové práce tvoří návrh a implementace aplikace pro propojení databází s využití LDAP pro autentizace a autorizace. Pro propojení jsou využity API a volání mezi nimi. API jsou zabezpečeny pomocí JWT tokenů, SSL certifikátů a přenášená data jsou zašifrována.

KLÍČOVÁ SLOVA

synchronizace dat, API, synchronizační bod, zabezpečení API, AD

TITLE

Design and implementation of an application for integrating different API databases.

ANNOTATION

The aim of the master thesis is to implement database integration for their automatic synchronization using APIs.

The theoretical part introduces API interfaces and existing solutions for database integration. Next explores virtualization options, Active Directory technology, and the LDAP protocol.

The practical part consists of designing and implementing an application for database integration using LDAP for authentication and authorization. APIs are utilized for the integration, secured with JWT tokens and SSL certificates, and encrypted data transmission.

KEYWORDS

data synchronization, API, synchronization point, API security, Active Directory

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

AD	Active Directory
LDAP	Lightweight Directory Access Protocol
NOS	Network Operating System
SID	Configuring Security Identifier
NetBIOS	Network Basic Input Output System
WINS	Windows Internet Naming Service
DNS	Domain Name System
SAM	Security Account Manager
DC	Distinguished Name
ESE	Extensible Storage Engine
GC	Global Catalog
MMC	Microsoft Management Console
WMI	Windows Management Instrumentation
GPO	Group Policy Object
ADFS	Active Directory Federation Service
RODC	Read-Only Domain Controller
API	Application Programming Interface
PDC	Programme Delivery Control
BDC	Business Development Company
OU	Organizational Unit
ADAM	Active Directory Application Mode
AD LDS	Active Directory Lightweight Directory Services
AD CS	Active Directory Certificate Services
AD RMS	Active Directory Rights Management Services
SSO	Single sign-on
IRM	Information Rights Management
LDIF	LDAP Data Interchange Format
DUA	Directory User Agent
DSA	Directory Server Agent

RPC	Remote procedure call
2FA	Two-factor authentication
UML	Unified Modeling Language
DTO	Data Transfer Object

TERMINOLOGIE

Proprietární – označuje software nebo technologii, která je vlastněna a kontrolována určitým subjektem a není distribuována s otevřeným zdrojovým kódem.

Middleware – software, který funguje jako prostředník mezi různými aplikacemi nebo systémy, usnadňující jejich komunikaci a integraci.

ÚVOD

V současnosti existují ve firemních sítích na sebe provázané systémy s potřebou vzájemné automatické synchronizace dat. Mnoho z nich v sobě však nemá nativní synchronizaci dat vůči dalším systémům implementovanou. Tento stav neumožňuje využít celý potenciál systémů a tím do určité míry potenciál dotčených systémů znehodnocuje.

Cestou k dosažení vyšší efektivity při práci s daty je vytvořit automatizaci pro synchronizaci dat. Pokud jeden systém obsahuje podkladová data pro ostatní systémy, je nutné rozhodnout, zda integrovat každý systém zvlášť metodou point to point, nebo vytvořit integrační bod pro všechny dotčené systémy. Z hlediska dodržování pravidel best practice se jeví, že lepší je použít metodu spočívající ve vytvoření integračního bodu.

Vytvořením integračního bodu se především eliminuje velká část problémů se synchronizací, nevhodným napojením systémů, neznalostí, případným rozbitím systémů a dalších implementačních obtíží. S ohledem na očekávanou potřebu synchronizace dat z podkladového systému do více a více aplikací, je vytvoření integračního bodu efektivnější cestou než implementovat napojení každé aplikace zvlášť. A to i pro firmy s menším počtem systémů, u kterých se vytvoření integračního bodu nemusí jevit efektivní.

Ve firemních sítích je pro autentizaci a autorizaci standardem využití Active Directory a k němu příslušného protokolu LDAP. Autentizace a autorizace v rámci integračního bodu je tedy také třeba řídit tímto způsobem.

Cíle diplomové práce vedou k návrhu řešení efektivního propojení aplikací pro synchronizaci dat mezi jednotlivými ukázkovými systémy.

Teoretická část bude analyzovat již používaná řešení, popisovat technologie Active Directory a příslušný protokol LDAP, rozhraní API a možnosti virtualizace důležité pro vývoj i pro případné nasazení synchronizace do produkčního prostředí. Výsledky analýzy budou použity pro implementaci praktické části diplomové práce.

Praktická část bude navrhovat a implementovat integrační bod pro propojení systémů. Vzhledem k předpokladu, že integraci budou řešit firmy využívající Active Directory, bude autentizace a autorizace uživatelů řešena pomocí Active Directory a protokolu LDAP.

1. Active Directory

Active Directory (AD) je síťový operační systém (NOS) společnosti Microsoft, postavený na systémech Windows 2000, Windows Server 2003, Windows Server 2008 a dalších verzí Windows Server. Umožňuje správcům efektivně spravovat celopodnikové informace z centrálního úložiště, které lze globálně distribuovat. Jakmile jsou informace o uživatelích, skupinách, počítačích, tiskárnách, aplikacích a službách přidány do Active Directory, mohou být dle potřeby zpřístupněny pro použití lidem v celém podniku. Struktura informací může odpovídat struktuře dané organizace a uživatelé se mohou dotazovat Active Directory na umístění tiskárny nebo e-mailovou adresu kolegy. Pomocí organizačních jednotek lze delegovat kontrolu a správu dat dle aktuální potřeby. Velké společnosti mívají značné množství dat, zaměstnanců a počítačů. Import dat do Active Directory a jejich správa může působit komplikovaně. Pro tyto případy má Microsoft několik velmi robustních, ale snadno použitelných rozhraní API (Application Programming Interfaces), které pomáhají usnadnit programovou správu dat.

Pro lepší přehled o využití AD a funkcích, které AD nabízí, je potřeba popsat jeho historii a vývoj.

Zdroje [1], [2], [3]

1.1. Historie Active Directory

První implementace řízení uživatelů byla již ve Windows NT 3.0. Tento systém byl pojmenován Network operating system (NOS). O NOS, více v kapitole LDAP.

První plnohodnotné AD bylo implementováno ve Windows 2000.

Zdroje [1], [2]

1.1.1. Windows NT 3.0 vs Windows 2000

Windows NT a Active Directory poskytovaly klientům adresářové služby. Ačkoli oba sdílely některé společné koncepty, například Security Identifiers (SID) k identifikaci objektů zabezpečení, velmi se lišily z hlediska funkce, škálovatelnosti a funkčnosti.

Zdroje [2], [4], [5]

Windows NT	**Active Directory**
Používá se replikace s jedním hlavním serverem, od hlavního řadiče primární domény (PDC) po podřízené záložní řadiče domény (BDC).	Multimaster replikace se používá mezi všemi řadiči domény.
Systémová pravidla lze použít lokálně na počítačích nebo nastavit na úrovni domény.	Pravidla skupin lze spravovat centrálně a aplikovat je na uživatele v rámci doménové struktury na základě kritérií domény, lokality nebo organizační jednotky (OU).
Doména má pravidla, replikace a je hranicí zabezpečení.	Doména má pravidla a hranici replikace. Les je bezpečnostní hranicí.
Pro překlad názvů se používají NetBIOS a WINS.	DNS se používá pro překlad jmen. Pro aplikace nebo starší klienty může být vyžadována služba WINS.
Objekt je nejmenší jednotka replikace.	Atribut je nejmenší jednotka replikace. V systému Windows Server 2003 Active Directory a výše se některé atributy replikují na základě hodnoty (například atribut člena skupinových objektů).
Maximální doporučená velikost databáze pro Security Accounts Manager (SAM) je 40 MB.	Doporučená maximální velikost databáze pro Active Directory je 16 TB.
Maximální efektivní počet uživatelů je 40 000 (při doporučeném maximu 40 MB).	Maximální počet objektů na les se pohybuje v desítkách milionů. Microsoft testoval na 1 miliardě uživatelů.
Čtyři modely domén (single, single-master, multimaster, complete-trust) jsou vyžadovány k vyřešení problémů s administrátorskými hranicemi jednotlivých domén a uživatelskými limity.	Je implementován model úplné důvěry, nejsou vyžadovány žádné doménové modely. Jednosměrné vztahy důvěryhodnosti s externími doménami, doménovými strukturami a sférami UNIX Kerberos lze implementovat ručně.
Schéma není rozšiřitelné.	Schéma je plně rozšiřitelné.
K datům lze přistupovat pouze prostřednictvím rozhraní Microsoft API.	K datům lze přistupovat prostřednictvím rozhraní Microsoft API nebo prostřednictvím LDAP. Jde o standardní protokol používaný adresáři, aplikacemi a klienty, kteří chtějí přistupovat k datům adresáře. Umožňuje přístup k datům a jejich správu napříč platformami.

Zdroj [2]

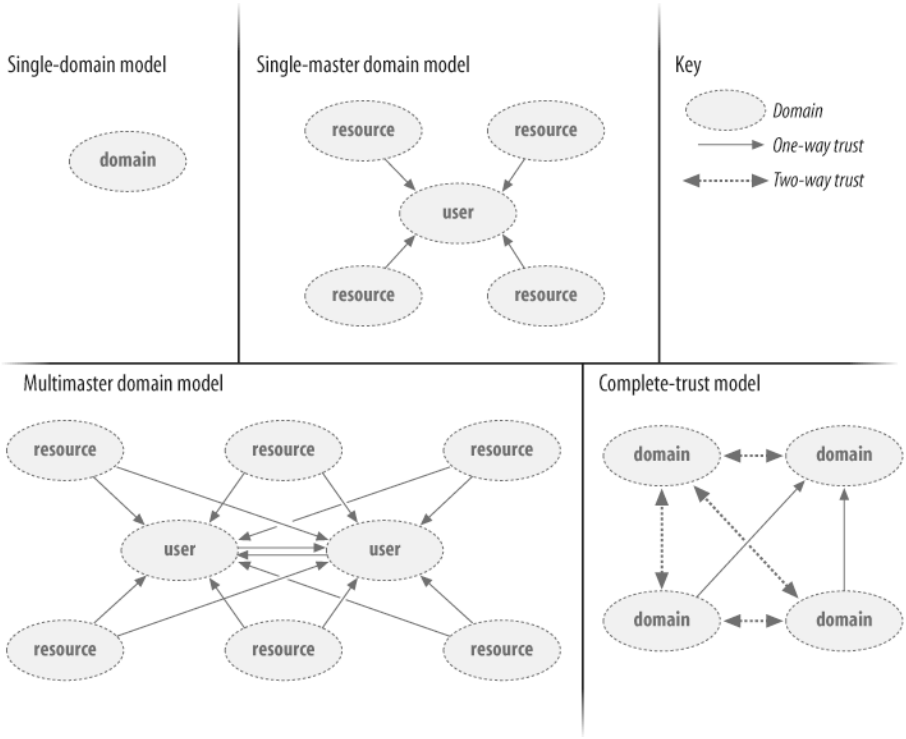
Primární řadiče a záložní řadiče domény Windows NT byly nahrazeny řadiči domény Active Directory. V rámci služby Active Directory je možné povýšit členské servery na řadiče domény (DC) a snížit úroveň řadičů domény na běžné členské servery, to vše bez nutnosti přeinstalace operačního systému. Toto u Windows NT nelze.

Organizační jednotky jsou důležitou změnou ve službě Active Directory. V systému Windows NT byla správa delegována na základě jednotlivých domén. Služba Active Directory umožňuje správcům definovat hranice správy od celé doménové struktury, domény nebo organizační jednotky až po jednotlivé objekty a atributy. To může výrazně snížit počet požadovaných domén a nabízí mnohem větší flexibilitu při výběru správy.

Navrhovaná maximální velikost databáze SAM (Windows NT Security Accounts Manager) byla 40 MB, což je ekvivalentní přibližně 40 000 objektů, v závislosti na podílu účtů počítačů, uživatelů a skupin, které byli v doméně. Mnoho společností kvůli velkému počtu skupin překročilo 75 MB pro SAM pro jednu doménu. Toto pravidlo nebylo pevně dané a bylo možné jej porušit, pokud byly brány v potaz eventuální problémy. Služba Active Directory je založena na databázi Extensible Storage Engine (ESE) používané serverem Exchange a byla vyvinuta pro uložení milionů objektů s maximální velikostí databáze 16 TB. To by mělo být většinou dostačující, počet objektů má pouze doporučený maximální limit. Tato nová databáze obsahuje všechny třídy objektů, nejen uživatele, skupiny a počítače SAM předchozí verze. S postupným vývojem aplikací s podporou Active Directory bylo do schématu přidáváno více tříd objektů a do adresáře více objektů.

Pro administrátory Windows NT mohlo být výrazné zvýšení škálovatelnosti tou nejdůležitější změnou. Velmi rychle a snadno bylo dosaženo doporučení maximálně 40 MB SAM v doméně NT, což administrátory přinutilo rozdělit doménu. Nakonec spravovali více domén, což bylo nežádoucí a nekomfortní. Žádná z domén nebyla organizována do stromu domén nebo podobné struktury. Mezi doménami nebyly žádné automatické důvěryhodnosti. Správci NT museli nastavovat manuální důvěryhodnosti mezi doménami. Důvěryhodnost musela být iniciována v obou doménách, aby se nastavil jediný jednosměrný vztah důvěryhodnosti. Přidání další domény znamenalo správu většího počtu důvěryhodností. Existují čtyři modely domén, které lze použít jako šablony pro návrh systému Windows NT: model jedné domény, model domény s jednou hlavní doménou, model domény s více hlavními servery a model domény s úplnou důvěryhodností.

Zdroje [1], [2], [5], [6]



Obrázek 1: Doménové modely [2]

1.1.2. Windows 2000 vs Windows Server 2003

První verze Active Directory dostupná s Windows 2000 byla velmi stabilní a bohatá na funkce. Stále však měla prostor pro zlepšení, především v oblasti správy a výkonu. V systému Windows Server 2003 společnost Microsoft mnohé z problémů vyřešila. Pro využití funkcí bylo třeba upgradovat řadiče domény na systém Windows Server 2003 a podle potřeby zvýšit úroveň funkčnosti domény a doménové struktury.

Rozdíl mezi Windows 2000 Active Directory a Windows Server 2003 Active Directory byl ve vývoji stávajících funkcionalit, nikoli ve velkých revolučních změnách.

Některé z nových funkcí byly dostupné již po povýšení prvního řadiče domény Windows Server 2003 do existující domény Windows 2000 Active Directory.

Zdroje [1], [2], [3], [6], [7]

Vlastnosti	**Popis**
Aplikační oddíly	Lze vytvořit vlastní oddíly pro ukládání dat odděleně od výchozích oddílů a lze nakonfigurovat, které řadiče domény (DC) v doménové struktuře je replikují.
Globální katalog (GC) není vyžadován pro přihlášení (tj. ukládání do mezipaměti univerzální skupiny)	Pod Windows 2000 musel DC kontaktovat GC, aby určil členství v univerzální skupině a následně umožnil uživatelům přihlásit se. Tato funkce umožňuje řadičům domény ukládat do mezipaměti členství v univerzální skupině, takže nemusí být nutné kontaktován GC kvůli přihlášení.
Vylepšení konzole Microsoft Management Console (MMC) a nové nástroje příkazového řádku	Nová konzole Active Directory Users and Computers umožňuje ukládat dotazy, přetahovat a upravovat více uživatelů najednou a mnohem efektivnější procházení velkým množstvím objektů. Kromě toho je se serverem nainstalováno několik nových nástrojů příkazového řádku (dsadd, dsmod, dsrm, dsquery, dsget a dsmove), které umožňují větší flexibilitu při správě Active Directory.
Instalace z média	Správci mohou vytvářet nové řadiče domény pro již existující doménu instalací ze zálohy existujícího řadiče domény, která je umístěna na CD nebo DVD.
Filtrování WMI pro GPO	Filtr WMI, je dotaz, který může využívat jakékoli informace WMI o klientovi, lze použít na objekt GPO. Dotaz bude spuštěn proti každému cílovému klientovi. Pokud je dotaz úspěšný, objekt GPO bude pokračovat ve zpracování, jinak se zpracování zastaví. Tato funkce vyžaduje, aby klienti měli Windows XP nebo lepší.

Zdroje [2]

Dále Windows Server 2003 přináší nové funkce dostupné v doménách. Doménu lze změnit na funkční úroveň Windows Server 2003, pokud všechny řadiče domény v doméně používají Windows Server 2003.

Zdroje [1], [7]

Vlastnosti	**Popis**
Přejmenování řadiče domény	Měl-li být řadič domény přejmenován v systému Windows 2000, bylo třeba řadič domény snížit, přejmenovat a znovu povýšit. S doménou Windows Server 2003 přejmenování řadiče domény vyžaduje pouze jeden restart.
Časové razítko přihlášení replikováno	V systému Windows 2000 atribut lastLogon obsahoval časové razítko posledního přihlášení uživatele. Tento atribut nebyl replikován mezi řadiči domény. V systému Windows Server 2003 lastLogonTimeStamp je atribut průběžně aktualizován, přibližně každých 10 dní.
Kvóty	Uživatelé a počítače mající přístup pro zápis do AD, mohou způsobit útok Denial of Service (DOS) vytvářením objektů, dokud se disk DC nezaplní. Tomuto typu útoku lze zabránit pomocí kvót. Nastavením kvóty lze omezit počet objektů, které může objekt zabezpečení vytvořit v oddílu, kontejneru nebo organizační jednotce. Windows Server 2003 DC může vynutit kvóty, i když nejsou na funkční úrovni domény Windows Server 2003. Pro správnou aplikaci kvót musí všechny DC používat Windows Server 2003.

Zdroje [2]

1.1.3. Windows Server 2003 vs Windows Server 2003 R2

Vydání systému Windows Server 2008 se opakovaně prodlužovalo. Společnost Microsoft vydala prozatímní aktualizaci systému Windows Server 2003 – Windows Server 2003 R2. R2 obsahuje Windows Server 2003 SP1 a také řadu volitelných doplňkových komponent Active Directory. Některé z těchto nových volitelných součástí, například Active Directory Application Mode (ADAM), byly k dispozici na stránkách Microsoftu. Microsoft je pro zpřístupnění širší veřejnosti přidal na CD R2. Někteří uživatelé zpochybňovali závazek společnosti Microsoft k softwaru, který je dostupný pouze z jeho webových stránek. Začleněním komponent do Core OS byly rozptýleny jakékoli pochybnosti o pozici podpory Microsoftu.

Service Pack 1 nabídl značné množství vylepšení pro Windows Server 2003. Také Windows XP Service Pack 2 obsahoval mnoho změn týkajících se zabezpečení, opravoval problémy v aplikaci Internet Explorer a nabídl nové funkce brány firewall.

Zdroje [1], [2], [7]

Vlastnosti	**Popis**
Připomenutí zálohování adresářové služby	Speciální zprávy zaznamenané do protokolu událostí adresářové služby, pokud nejsou zálohovány oddíly adresáře.
Další zabezpečení replikace a menší chybovost	Metadata replikace pro řadiče domény odebrané z domény se již nezachovávají. To zvyšuje zabezpečení adresářů a eliminuje chybové zprávy replikace související s odstraněnými řadiči domény.
Vylepšení instalace z média pro instalaci serverů DNS	Nová možnost zahrnout oddíly adresáře aplikace do zálohovacího média eliminuje požadavek na síťovou replikaci oddílů adresáře aplikací DomainDNSZone a ForestDNSZones před uvedením serveru DNS do provozu.
Aktualizované nástroje	Novější verze DcDiag, NTDSUtil, IADSTools.DLL, AdPrep a dalších nástrojů, které pomáhají při správě, aktualizacích a odstraňování problémů.
Podpora virtuálních serverů	Oficiální podpora pro spouštění doménových řadičů v rámci Microsoft Virtual Server 2005. Byla přidána další logika k ochraně

	před poškozením adresáře v důsledku nesprávných postupů zálohování a obnovy.
Rozšířené úložiště smazaných objektů	Životnost smazaných objektů v nových lesích se zvýšila z 60 na 180 dní. Stávající lesy se neupravují. V důsledku chyby regrese měly nové doménové struktury Windows Server 2003 R2 životnost 60 dní. Následně došlo k opravě v systémech Windows Server 2003 SP2 a Windows Server 2008 na 180 dní.
Důvěrné atributy	Možnost označit atributy jako důvěrné. Nelze je číst bez udělení dodatečných oprávnění. Ve výchozím nastavení mohou jakýkoli atribut označený jako důvěrný číst pouze důvěryhodní uživatelé s plným přístupem k objektu, to však může být delegováno granulárním způsobem.
Active Directory Federated Services (ADFS)	Technologie založená na standardech umožňující distribuovanou identifikaci, ověřování a autorizaci napříč hranicemi organizace a platformy.

Zdroje [2]

1.1.4. Windows Server 2003 R2 vs Windows Server 2008

Windows Server 2008 je firmami stále používán. Zavedl podstatná a v některých případech komplikovaná vylepšení služby Active Directory. Nejpodstatnějšími funkcemi jsou zavedení Server Core a podpora pro provozování Active Directory na Server Core spolu se zavedením řadičů domény pouze pro čtení (RODC). Rozdíly mezi základními službami Active Directory v systémech Windows Server 2003 R2 a Windows Server 2008 jsou opět ve vývoji stávajících funkcionalit, nikoli ve velkých revolučních změnách.

Zdroje [1], [2], [8]

Vlastnosti	Popis
Řadiče domény pouze pro čtení (RODC)	RODC neumožňují místní zápisy. Ve výchozím nastavení neukládají hesla. Tato funkce přidává velkou míru zabezpečení řadičům domény v místech s nedostatečným fyzickým zabezpečením.
Jemné zásady pro hesla	Zásady hesel lze nyní definovat pro jednotlivé uživatele nebo skupiny.
Oddělení administrativních rolí	Uživatelům, kteří nejsou správci domény, lze bezpečně delegovat správu správy řadičů domény jen pro čtení, bez poskytnutí přístupu ke službě Active Directory.
DNS pouze pro čtení	RODC mohou hostovat dynamické zóny DNS a odkazovat aktualizace na zapisovatelné řadiče domény.
Nová infrastruktura auditování a protokolování	Auditování přístupu a změn Active Directory, stejně jako různé další akce, byly zcela přepracovány.
Statistika posledního přihlášení	Klienti Windows Vista a Windows Server 2008 mohou ukládat a zobrazovat podrobné informace o úspěšnosti a selhání posledního přihlášení přímo na uživatelských objektech v adresáři.
Snímky databáze Active Directory	Snímky databáze Active Directory lze pořídit a připojit jako základ pro obnovu po havárii a dalších operacích obnovy objektů.
Pravidla Starter Group	Lze definovat šablony pravidel skupiny, na kterých mohou správci založit nová pravidla.

Zdroje [2]

1.2. Nejdůležitější funkce Active Directory

Hlavní službou nabízenou Active Directory je služba Domain Service, nazývaná „AD DS“. Je to služba, která ukládá informace o adresáři a spravuje interakci uživatele s doménou. Když se uživatel přihlásí do systému nebo se pokusí připojit k serveru v síti, služba AD DS provede úlohu ověření uživatelského přístupu. Technicky je přístup uživatelů k jakémukoli síťovému prostředku řízen službou AD DS. Produkty Exchange Server a SharePoint Server společnosti Microsoft spoléhají při poskytování přístupu k síťovým prostředkům na službu AD DS. Server hostující službu AD DS se nazývá řadič domény (DC).

Zdroje [1], [2], [9]

1.2.1. Struktura Active Directory

Active Directory firemní strukturu ukládá v logických celcích:

- **Doména**: Doména označuje sadu objektů (uživatelů, periferií, zařízení), které sdílejí společnou databázi AD. Objekty jsou rozpoznány podle názvu DNS (abc_xyz.cz).
- **Strom**: Jedna nebo více domén se spojí a vytvoří strom. Strom má společný kořenový název DNS. Například oddelenia.firma.cz, oddelenib.firma.cz a oddelenic.firma.cz.
- **Lesní struktura**: Jeden nebo více stromů sdílejících společné schéma, konfiguraci adresáře nebo globální katalog je považován za doménovou strukturu. Nemá však souvislý jmenný prostor jako strom. Lesy fungují jako bezpečnostní hranice pro podnikové sítě.
- **Organizační jednotky**: Objekty v rámci domény spadají pod organizační jednotky. Ty zjednodušují správu a správu zásad. Správci mohou vytvářet funkční, geografické nebo obchodní struktury a následně aplikovat skupinové zásady na všechny dotčené organizační jednotky. Organizační jednotky hrají klíčovou roli při delegování kontroly nad síťovými zdroji na různé správce.

Zdroje [2], [9]

1.2.2. Další důležité funkce Active Directory

Active Directory prošlo od první verze ve Windows NT výraznými změnami. Přibyly důležité funkce, například:

- **Active Directory Lightweight Directory Services (AD LDS)**: Odlehčená verze Domain Services eliminuje složité a pokročilé funkce z adresářových služeb tím, že omezuje používání doménových řadičů, doménových struktur nebo domén. Tyto služby se obvykle používají v malých kancelářských sítích.
- **Active Directory Certificate Services (AD CS)**: AD CS označuje služby používané k vydávání a správě digitálních certifikátů, které se často vyskytují v nastavení zabezpečení softwaru. Takové bezpečnostní systémy se spoléhají na technologie veřejného klíče. Digitální certifikáty poskytované službou AD CS lze použít k šifrování a podepisování digitálních dokumentů. Tyto certifikáty lze dále použít k ověřování počítačů, uživatelů nebo zařízení v síti.
- **Active Directory Federation Services (AD FS)**: Služba AD FS umožňuje jednotlivcům používat funkci jednotného přihlášení (SSO) pro přístup k aplikacím a systémům mimo podnikové prostředí. Je to podobné jako webová funkce, kterou mohou dodavatelé použít k přihlášení do osobní sítě a zároveň získat oprávnění pro přístup do privátní sítě klienta.
- **Active Directory Rights Management Services (AD RMS)**: Služba AD RMS je používána organizacemi jako bezpečnostní strategie k zabezpečení a ochraně dokumentů pomocí zásad správy práv k informacím (IRM). Služba AD RMS umožňuje jednotlivcům a správcům definovat oprávnění pro přístup k dokumentům, sešitům a prezentacím.

Zdroje [9], [10], [11], [12], [13]

1.3. Alternativy Active Directory

Active Directory, má své alternativy. Tyto alternativy jsou využitelné například, pokud administrátor nechce mít mezi svými servery Windows Server, pokud chce nainstalovat pouze Active Directory a ne celý Windows Server, pokud chce vyžívat pouze open source software nebo pokud chce AD a LDAP virtualizovat.

1.3.1. Apache Directory

Apache Directory je open-source software navržený Apache Software Foundation. Apache Directory je adresářový server na bázi Java. Adresář byl schválen Open Group a databázemi založenými na Eclipse v roce 2006. Díky integraci se serverem Kerberos může podporovat další kódy.

Mezi pokročilé funkce patří prohlížeč schémat, editor DSML (Directory Services Markup Language), editor/prohlížeč LDAP a editor formátu LDIF (LDAP Data Interchange Format) pro directory server založený na Eclipse. Dále lze začlenit nové funkce pomocí upgradu zásuvných modulů založených na Eclipse.

Zdroje [9], [14]

1.3.2. OpenLDAP

OpenLDAP je open-source nástroj navržený v rámci projektu OpenLDAP. Jde o administrační nástroj používaný pro řízení databází LDAP. Obvykle plní funkci klienta Windows LDAP, který umožňuje procházet, vyhledávat, vytvářet, měnit a odstraňovat prvky umístěné na serveru LDAP.

Mezi další funkce patří procházení schémat, správa hesel, export a import LDIF a další.

Zdroje [9], [15]

1.3.3. FreeIPA

FreeIPA je open-source nástroj pro identitu a ověřování vyvinutý společností Red Hat. Nabízí sadu služeb, například správu identit, auditní služby a zásady přizpůsobené pro počítačové sítě Linux a Unix. Současná verze projektu RHEL 6.2 (Red Hat Enterprise Linux 6) si klade za cíl začlenit část funkcí, které AD nabízí.

Mezi jeho klíčové vlastnosti patří:

- Řešení pro správu bezpečnostních informací, které kombinuje Linux (Fedora), 389 Directory Server, MIT Kerberos a další.
- Rozšiřitelná rozhraní pro správu, včetně CLI, WEB UI a Python SDK.

Zdroje [9], [16]

1.3.4. Jxplorer

Jxplorer je multiplatformní nástroj, který lze spustit na Windows, Linuxu a mnoha dalších operačních systémech. Poskytuje rozhraní LDAP a DSML, která umožňují sledovat a spravovat adresář LDAP bez použití tradičního příkazového řádku. Jedná se o nástroj založený na Javě, který je poměrně přizpůsobivý a snadno konfigurovatelný. Je k dispozici v bezplatné verzi a v placené verzi pro firmy.

Zdroje [9], [17]

2. LDAP

Adresářové služby bezpečně spravují uživatele a jejich přístupová práva ke zdrojům IT v rámci organizace pomocí protokolů. LDAP je jedním ze základních protokolů používaných pro adresářové služby. Přestože adresářové služby mohou používat také další protokoly jako Kerberos, SAML, RADIUS, SMB, Oauth atd., většina z nich stále používá protokol LDAP.

Zdroje [18], [19]

2.1. Historie Active Directory

LDAP byl vyvinut v roce 1993 Timem Howesem a jeho kolegy z University of Michigan. Byl vytvořen s nízkou režií, jako odlehčená verze protokolů adresářových služeb X.500 používaných v té době, například protokol DAP (directory access protocol).

X.500 měl velké nároky na šířku pásma, byl tedy namáhavý pro systémy i pro síť. V důsledku toho se na počátku 90. let většina osobních počítačů nemohla připojit k adresářové službě X.500. V té době bylo použití X.500 omezeno na specifické systémy, jako jsou sálové počítače, minipočítače nebo mikropočítače.

LDAP vyřešil velké nároky na šířku pásma tím, že umožnil autentizaci a autorizaci uživatelů k IT zdrojům a zároveň snížil režii, využití šířky pásma a požadavky na koncové body. Díky těmto úpravám se LDAP stal autentizačním protokolem internetových adresářových služeb.

LDAP verze 3 byl navržen a přijat jako internetový standard pro adresářové služby koncem devadesátých let. Dnes je to nejnovější a nejrozšířenější verze LDAP. Po tomto milníku zahájil Kurt Zeilenga vydáním OpenLDAP 1.0 v roce 1998 projekt OpenLDAP. Verze obsahovala pokročilé bezpečnostní funkce, aktualizovanou podporu platformy a opravy chyb. Jednalo se také o první zcela open-source sadu klientských a serverových aplikací odvozených z LDAP v3.3.

OpenLDAP se těšil popularitě administrátorů. Mimo jiné umožnil správcům IT úpravy vyhovující potřebám konkrétních organizací. Howard Chu a tým Symas od roku 1999 nadále řídí vývoj OpenLDAP. V roce 1999 vydal Microsoft Active Directory, který používal LDAP a Kerberos při vytváření proprietárních rozšíření.

LDAP od té doby významně ovlivnil vývoj adresářových služeb a stal se nezbytnou součástí moderní IT infrastruktury.

Zdroje [2], [18], [19]

2.2. Fungování Lightweight Directory Access Protocol

LDAP specifikuje metodu adresářového úložiště, která umožňuje přidávat, mazat a upravovat záznamy. Umožňuje také prohledávat záznamy pro usnadnění autentizace a autorizace uživatelů ke zdrojům.

Tři hlavní funkce LDAP:

- **Aktualizace**: Zahrnuje přidání, odstranění nebo úpravu informací o adresáři.
- **Dotaz**: Zahrnuje vyhledávání a porovnávání informací v adresáři.
- **Autentizace**: Mezi hlavní autentizační funkce patří propojení a zrušení propojení.

Zdroje [18], [19]

2.3. Autentizace a autorizace LDAP

Protokol LDAP autentizuje a autorizuje uživatele k jejich prostředkům. Protokol ověřuje uživatele pomocí operace vazby, která uživateli umožňuje komunikovat s adresářem LDAP. Pokud vstupní přihlašovací údaje uživatele odpovídají údajům uvedeným pro uživatele v databázi, autorizuje protokol autentizovaného uživatele ke zdrojům, které potřebuje, a ke kterým má práva přístupu.

Zdroje [18], [19]

2.3.1. Autentizace LDAP

Autentizace LDAP závisí na operaci vazby klient/server. Tato operace umožňuje klientovi připojenému pomocí LDAP, také nazývanému adresářový uživatelský agent (DUA), komunikovat s adresářovým serverem, nazývaným také adresářový systémový agent (DSA), v zabezpečené šifrované relaci.

Při ověřování proti serveru LDAP, pro přístup k databázi, je uživatel vyzván k zadání svého uživatelského jména a hesla. Pokud se hodnoty zadané uživatelem do klienta shodují s hodnotami v databázi LDAP, server LDAP udělí uživateli přístup ke zdroji IT.

Zdroje [18], [19]

2.3.2. Autorizace LDAP

Jakmile je uživatel úspěšně ověřen, musí být autorizován pro přístup k požadovaným zdrojům. I když různé instance LDAP mohou tento proces strukturovat a kódovat mírně odlišně, je autorizace v podstatě dosaženo přiřazením oprávnění skupinám a rolím k adresáři.

Zdroje [18], [19]

2.4. Využití Lightweight Directory Access Protocol

LDAP se využívá například k ukládání uživatelských dat na jednom centrálním a dostupném místě nebo přidružuje uživatelům zdroje, ke kterým mají povolen přístup.

Autentizuje a autorizuje uživatele při přístupu:

- **Technické aplikace**: Jenkins, Docker, OpenVPN, sada Atlassian a mnoho dalších se nejlépe ověřují pomocí LDAP.
- **Infrastruktura serveru**: Servery se systémem Linux, místní i cloudové (například AWS) využívají LDAP k ověřování uživatelů.
- **Souborové servery**: Souborové servery QNAP, Synology a FreeNAS podporují LDAP.
- **Síťové vybavení**: Ačkoli může dojít k překrytí s protokolem RADIUS, některé organizace používají LDAP pro přístup k síti přes VPN, přístupové body WiFi a další síťová zařízení.

Zdroje [18], [19]

2.5. Alternativy Lightweight Directory Access Protocol

Active Directory a Lightweight Directory Access Protocol jsou propojeny. LDAP je součástí AD, tak i alternativy pro LDAP jsou stejné jako alternativy k AD.

3. API

API je zkratka pro Application Programming Interface. V kontextu API se slovo Aplikace vztahuje na jakýkoli software s odlišnou funkcí. Rozhraní lze chápat jako smlouvu o poskytování služeb mezi dvěmi aplikacemi. Tato smlouva definuje, jak spolu aplikace komunikují pomocí požadavků a odpovědí. Dokumentace API obsahuje informace, jak mají vývojáři strukturovat požadavky a odpovědi.

Zdroje [20], [21]

3.1. Historie API

První myšlenky na vytvoření API se zrodily již v roce 1951. Od té doby se představy o API výrazně změnily.

Zdroje [22], [23]

3.1.1. První představy o API

V 50. letech 20. století bylo API chápáno jako potenciální metoda pro usnadnění komunikace mezi dvěma počítači. Termín byl poprvé zmíněn v roce 1951 v knize napsané Mauricem Wilkesem a Davidem Wheelerem nazvané „Příprava programů pro elektronický digitální počítač“. Nastínili zde několik klíčových počítačových termínů, včetně ranné verze API. V této fázi začínalo existovat rozhraní API, které se však omezovalo na jednoduchá rozhraní příkazového řádku, která umožňovala programátorům komunikovat s počítači.

Zdroje [22], [23]

3.1.2. Vývoj API

V průběhu 60. let 20. století začaly počítače získávat na popularitě a organizace začaly experimentovat s jejich využitím. Termín API byl v této době chápán jako interakce jedné aplikace se zbytkem počítačového systému. Zavedením konzistentního aplikačního rozhraní (obvykle volání podprogramu Fortran) se programátoři mohli osvobodit od grafického zobrazovacího zařízení. Rozhraní API umožnilo komunikaci mezi sálovými počítači a jinými systémy, například terminály a tiskárnami.

Zdroje [22], [23]

3.1.3. Pokračování vývoje API

V 80. letech 20. století se počítačové sítě začaly stávat samozřejmostí a programátoři potřebovali přistupovat ke knihovnám umístěným na lokálních i na vzdálenějších počítačích. Rozhraní API umožňovalo volání vzdálených procedur (RPC), která obecně podporovala Javu. Rozhraní API hrálo klíčovou roli při zajišťování kompatibility mezi platformami.

Zdroje [22], [23]

3.1.4. API začíná být populární

V 90. letech 20. století vznikl internet a rozhraní API představovalo způsob vyměňování dat mezi aplikacemi pomocí standardní sady protokolů. Aplikace již nemusely sdílet zprávy pouze ve své síti. API umožňovalo aplikacím sdílet zprávy s aplikacemi v jiných sítích přes internet.

Zdroje [22], [23]

3.1.5. Nové milénium a růst popularity API

S rostoucí popularitou online aplikací začaly organizace přesouvat vše do cloudu. Salesforce, eBay a Amazon byly průkopníky v poskytování nových služeb. Pomocí protokolu HTTP poskytovaly přístup ke strojově čitelným datům ve formátu JSON nebo XML prostřednictvím webových rozhraní API. Inovativní startupy i velké podniky brzy implementovaly nabídky jako služby, které využívaly cloud a jeho model API-first. V době, kdy Apple v roce 2007 uvedl na trh iPhone, již použití API způsobilo revoluci ve způsobu výstavby infrastruktury společností.

Zdroje [22], [23]

3.1.6. Vývoj API do Cloudu

V polovině roku 2010 podpořili Kubernetes přechod k distribuovaným systémům, které se skládaly z volně propojených mikroslužeb. Každá z nich implementuje své vlastní API. Rozhraní API umožnilo společnostem rozšířit dosah svých webových aplikací všem uživatelům. Vzhledem k tomu, že rozhraní API bylo založeno na standardizovaných protokolech, umožnilo vývojářům vytvářet aplikace ve více prostředích a s více službami.

Zdroje [22], [23]

3.2. Alternativy API

V závislosti na požadavcích systémů, aplikační architektuře a use case lze použít alternativy pro API.

3.2.1. Přímá integrace

Při přímé integraci systémy sdílí knihovny, kód nebo moduly za účelem vytvoření propojení. Přímá integrace se často používá v monolitické architektuře, kde jsou komponenty složitě propojeny. Přímá integrace umožňuje bezproblémovou komunikaci, ale obvykle vede k závislosti, problémům při aktualizaci a změnám v dotčených systémech.

Zdroj [24]

3.2.2. Webhooks

Webhook nazývaný také webové zpětné volání nebo HTTP push API je způsob, jak může aplikace poskytovat jiným aplikacím informace v reálném čase. Na rozdíl od typických API webhooky dodávají data do jiných aplikací okamžitě.

Zdroj [24]

3.2.3. Fronta zpráv

Fronta zpráv je forma middlewaru používaného při vývoji softwaru k umožnění komunikace mezi službami, programy a odlišnými komponentami, například operačními systémy a komunikačními protokoly. Fronta zpráv je podobná SMS zprávám. Stejně jako posílání SMS zpráv dává uživateli mobilního telefon možnost číst zprávy postupně, dle časových možností. Fronta zpráv umožňuje hladký tok informací a zpracování dat při nízkém zatížení cílového systému.

Zdroj [25]

3.2.4. Replikování databází

Při replikování databází si systémy data nepředávají mezi sebou. Data jsou napřímo replikována z jedné databáze do druhé. Replikování databází je možné využít, je-li vyžadována okamžitá synchronizace a přístup k datům.

3.2.5. Integrace pomocí sdílených souborů

Data jsou aktualizována pomocí souborů ve sdíleném souborovém systému. Tento způsob je nejjednodušší, má však i své nevýhody, například nekonzistence dat, omezená škálovatelnost při vysokém využití zdrojů nebo omezená funkcionalita při napojování zdrojů.

3.3. Zabezpečení API

Pomocí API se přenášejí data, je tedy třeba API dostatečně zabezpečit. Způsobů zabezpečení se nabízí celá řada.

3.3.1. Autentizace a autorizace

Nejzákladnější obranou proti neoprávněnému přístupu je implementace ověřovacího systému. Ten vyžaduje od uživatelů, před každým přístupem k datům, poskytnutí platných loginů. Autorizace určuje úroveň přístupu uživatele ke konkrétním informacím. Zároveň je možné implementovat dvoufaktorovou autentizaci (2FA). Při tomto způsobu zabezpečení útočníkovi nestačí získat pouze login a heslo, musí získat i druhou ověřovací metodu (například potvrzení na mobilním telefonu).

Zdroje [26], [27]

3.3.2. Šifrování SSL/TLS

Veškerá komunikace mezi API a klienty by měla být zabezpečena prostřednictvím připojení SSL nebo šifrovacího protokolu TLS, jako je HTTPS. Tím je zajištěno, že všechna data jsou zašifrována a chráněna před škodlivými třetími stranami. Šifrování pomocí SSL/TLS je řešeno certifikáty.

Zdroj [26]

3.3.3. Omezení počtu přístupů

Dalším způsobem zabezpečení API před automatickými útoky je implementace omezení počtu přístupů k danému API. Díky tomuto opatření bude klientovi zajištěno získání požadované informace v potřebném čase. Omezení počtu přístupů je jedna z možných ochran proti takzvaným DDoS útokům. Při DDoS útoku se snaží útočník zahltit systém velkým počtem dotazů a následně systém dostat do stavu pro běžného klienta nefunkčního.

Zdroj [26]

3.3.4. Auditní záznamy o přístupu k API

Nedílnou součástí ochrany jakéhokoli systému je sledování chování uživatelů. Sledování, k čemu uživatelé přistupují a jak s daty zacházejí. Je nutné zaznamenávat každý požadavek API a uchovávat auditní protokoly o aktivitě uživatele, aby byla zajištěna bezpečnost dat a dodržování předpisů. Pomocí auditních záznamů je možné identifikovat prostup útočníka přes ochranu a identifikovat potřebu problém aktivně řešit.

Zdroje [26], [28]

3.3.5. Omezení přístupu k API

Zabezpečení omezení přístupu k API úzce souvisí s metodou autentizace a autorizace implementovanou na daném systému. U tohoto způsobu zabezpečení je uživateli nebo skupině uživatelů povolen přístup pouze k určitým API nebo metodám. Pokud není zvolen správný způsob autentizace a autorizace, není možné s omezením přístupu k API efektivně nebo vůbec pracovat.

Zdroj [26]

3.3.6. Monitorování přístupu k API

Monitorování přístupu k API je důležité pro sledování chování uživatelů a pro upozornění administrátora na zjištění podezřelé aktivity (například více násobný neúspěšný pokus o přístup k určitému API).

Zdroj [26]

3.3.7. Šifrování dat

Veškerá data uložená a přenášená, je třeba šifrovat. Zajistí-li si útočník přístup k šifrovaným datům, šifrování mu znemožní neoprávněně získaná data použít.

Zdroj [26]

3.3.8. Web Application Firewall (WAF)

WAF je bezpečnostní software umístěný mezi API a internetem. Přes WAF prochází veškerý provoz z internetu do sítě. WAF filtruje škodlivý provoz předtím, než se dostane až k serveru. WAF je účinný způsob, jak chránit server před útoky (například před DDoS útoky).

Zdroje [26], [29]

3.3.9. Brána rozhraní API

Brány rozhraní API (gateway) pomáhají chránit API před útoky. Gateway hlídají a zkoumají veškerý provoz mezi klientem a API. Pokud je provoz vyhodnocen jako škodlivý nebo podezřelý, je nahlášen nebo zablokován a dotaz se na API vůbec nedostane.

Zdroj [26]

4. Databáze

Databáze je způsob organizace informací umožňující uživatelům rychle procházet daty, zjišťovat trendy a provádět další akce.

Počítačové databáze byly poprvé představeny světu v 60. letech 20. století a od té doby se staly základem pro produkty, analýzy, obchodní procesy a další. Mnoho služeb používaných každý den online v bankovnictví, sociálních médiích, nakupování nebo e-mailech, je postaveno na databázích.

Jednotlivé typy databází se liší způsobem práce s daty a jsou vhodné ke zpracování rozdílných typů dat.

Zdroje [30], [31]

4.1. Historie databází

Historie databází sahá do 60. let 20. století a je vzájemně propojena s historií počítačů.

Zdroje [30], [32]

4.1.1. Začátky databází

Počítačové databáze začaly vznikat v 60. letech 20. století, kdy se stalo používání počítačů pro soukromé organizace cenově efektivní možností, jak data ukládat a jak s nimi pracovat. S poklesem cen bylo snazší přesunout úložiště dat a databáze do počítačů.

Zdroje [30], [32]

4.1.2. Nástup relační databáze

Na začátku 70. let 20. století EF Codd publikoval důležitý dokument, kterým představil relační databáze. Revoluční pohled na problematiku databází změnil i přístup odborné veřejnosti k práci s databázemi. V novém modelu je schéma databáze nebo logická organizace odpojena od fyzického úložiště informací, což se stalo standardním principem databázových systémů.

Zdroje [30], [32]

4.1.3. Nástup jazyka SQL

Structured Query Language neboli SQL se stal standardním dotazovacím jazykem pro databáze, který v roce 1986 vybral Americký národní institut pro standardy a v roce 1987 Mezinárodní organizace pro standardizaci.

Zdroje [30], [32]

4.1.4. Další vývoj databází

Devadesátá léta sehrála klíčovou roli v rozvoji databází a databázového softwaru. Podobně jako v 60. letech vedl širší rozvoj ICT k dalšímu rozvoji tohoto odvětví. Většina společností prodávala složité databázové produkty za vysoké ceny.

Zdroje [30], [32]

4.1.5. Postupný vývoj databází

Přestože internetový průmysl zaznamenal na počátku 21. století pokles, databázové aplikace nadále rostly. Byly vyvinuty nové interaktivní aplikace pro PDA.

Zdroje [30], [32]

4.2. Alternativy databází

Alternativou k ukládání dat mimo databází je ukládání dat v souborech. Pokud jsou data ukládána v souborech, nedá se s nimi následně pracovat tak komplexně, jako s daty uloženými v databázích. Například dotazování a filtrování dat, indexace dat a optimalizace, zálohování a obnovení dat.

5. Virtualizace

Virtualizace je vytvoření jednoho nebo více virtuálních systémů na jednom fyzickém počítači. Tyto virtuální systémy mohou přistupovat ke všem částem fyzického stroje včetně procesoru, paměti nebo úložiště. Každý virtuální stroj běží na svém vlastním operačním systému a může fungovat zcela samostatně.

Zdroj [33]

5.1. Typy virtualizace

Existuje celá řada typů virtualizace a každý typ je vhodný a určen k jinému použití.

Vybrané typy virtualizace:

- **Virtualizace serveru** – správce vytvoří jeden nebo více virtuálních strojů na jednom fyzickém hostitelském zařízení. To umožňuje každému virtuálnímu stroji provozovat vlastní operační systém a aplikace.
- **Virtualizace počítače** – správce vytvoří prostředí virtuálních počítačů na centrálním serveru, ke kterému mohou uživatelé přistupovat vzdáleně.
- **Virtualizace úložiště** –zahrnuje abstrahování fyzických úložných zařízení a sdružování jejich kapacity do jediného virtuálního úložného prostředku.
- **Virtualizace sítě** – správce vytvoří síť nebo sítě, které fungují nezávisle na fyzické síťové infrastruktuře hostitele.
- **Virtualizace aplikací** –zahrnuje vytváření virtuálních instancí aplikací, které mohou běžet nezávisle na hostitelském operačním systému.

Zdroj [34]

5.2. Virtualizace pomocí technologie Docker

Docker používá takzvanou kontejnerovou virtualizaci. Kontejner je izolovaný, není závislý na hostitelském systému a lze jej jednoduše přenášet mezi různými hostitelskými systémy.

Vybrané rozdíly mezi klasickou virtualizací a virtualizací pomocí Dockeru:

- **Využití zdrojů** – kontejnery v Dockeru využívají zdroje hostitelského systému efektivně. Sdílejí se knihovny a HW zdroje jsou kontejneru přiřazeny až ve chvíli, kdy je kontejner potřebuje.
- **Izolace** – virtualizované kontejnery jsou mezi sebou zcela oddělené a díky tomu jsou mezi sebou zcela izolované. Kontejnery si vzájemně sdílejí pouze jádro hostitelského systému.
- **Velikost a potřebný výkon kontejneru** – klasická virtualizace bývá běžně velmi náročná na systémové požadavky, protože si alokuje veškeré potřebné HW zdroje ihned po spuštění. Kontejnery v Dockeru si však alokují HW zdroje až ve chvíli, kdy je potřebují využít.

Kontejnerová virtualizace není vždy nejlepší cesta jakou zvolit pro virtualizaci. Avšak pro vývoj a nasazení vybraných druhů aplikací, jako jsou například mikrosložby, moderní webové aplikace Cloud-native aplikace, se jedná o správnou cestu.

Zdroj [35]

6. Nástroje pro integraci systémů

Nástroje pro integraci systémů jsou k dispozici, každý takový nástroj musí obsahovat podporu všech systémů, které firma potřebuje integrovat. Vzhledem k velké rozmanitosti cílových aplikací nelze podporu zaručit. Při použití zvoleného nástroje se firma může dostat do situace, kdy bude muset například volit mezi podporou integrace a funkčností systému.

Vybrané systémy pro integraci:

- Zapier
- Make
- Effortless API Connect

7. Návrh aplikace

Diplomová práce v praktické části obsahuje tři ukázkové systémy, které mezi sebou vyžadují automatizovat integraci. Dále obsahuje integrační bod, který zajišťuje automatizování integrace a webové rozhraní pro základní funkčnosti.

Pro správnou implementaci aplikace je důležitý návrh aplikace. Během této fáze lze stanovit například funkčnost systému, aktéry, bližší určení rozpočtu, datovou náročnost nebo databázové entity.

7.1. Funkční a nefunkční požadavky

Stanovení funkčních a nefunkčních požadavků je jeden z prvotních kroků provedené analýzy softwaru. Stanovení požadavků je důležité proto, aby finální software splňoval všechna očekávání zákazníka.

7.1.1. Funkční požadavky

Funkční požadavky specifikují, co je od systému očekáváno. V rámci diplomové práce byly stanoveny následující funkční požadavky:

- Integrační bod bude mít webové rozhraní, které bude obsahovat základní funkčnosti.
- Administrátor systému bude moci realizovat správu uživatelů.
- Administrátor bude moci ručně vyvolat integraci mezi systémy.
- Uživatelé si budou moci ve webovém rozhraní zkontrolovat stav systémů.

7.1.2. Nefunkční požadavky

Nefunkční požadavky specifikují, jaké funkce jsou požadovány pro daný systém. V rámci diplomové práce byly stanoveny následující nefunkční požadavky:

- Ukázkové systémy i integrační bod budou využívat API.
- Ukázkové systémy i integrační bod budou využívat zabezpečenou komunikaci pomocí TLS.
- Synchronizovaná data budou při přenášení mezi API zašifrována asymetrickou šifrou.
- Projekt bude napsán v jazyce Java a bude využívat framework Java Spring 3.

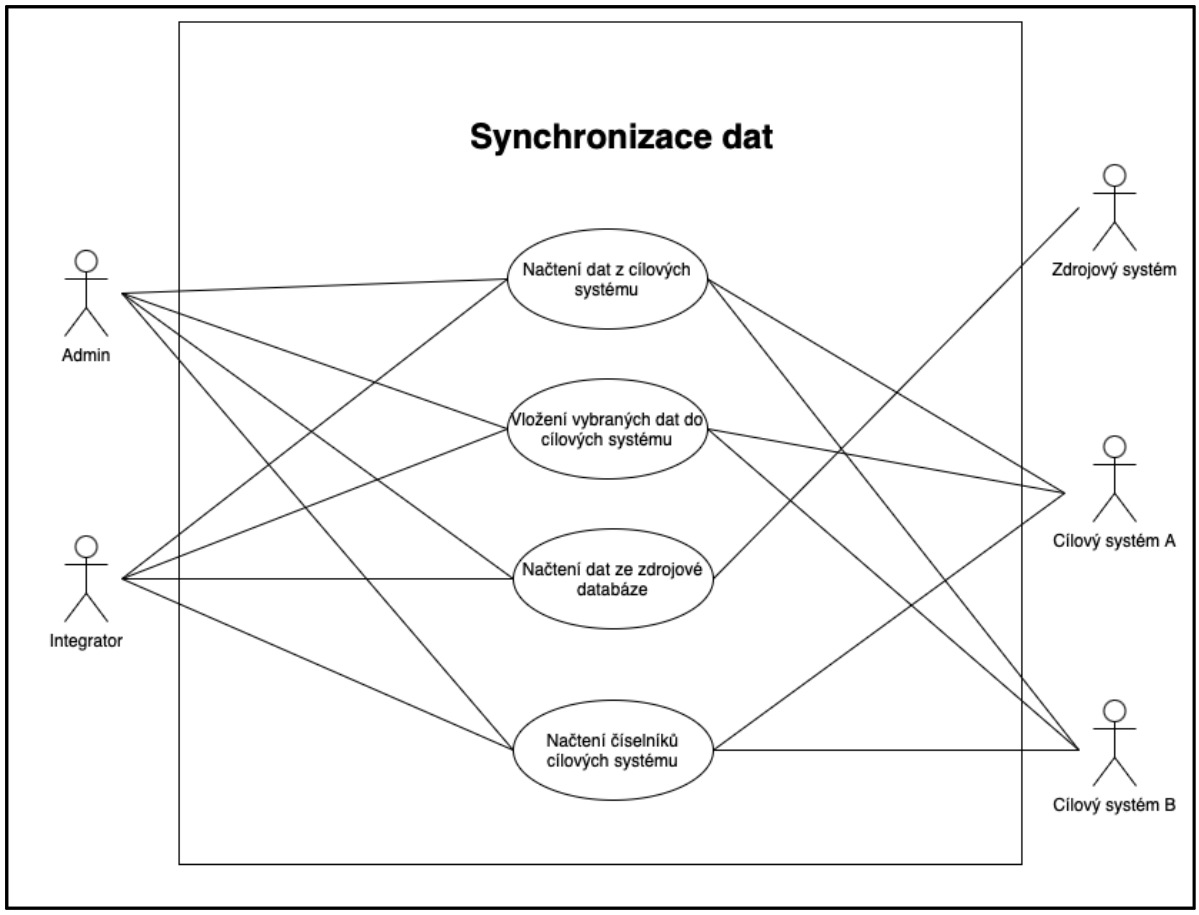
7.2. Případy použití

Dalším důležitým bodem při návrhu aplikace je stanovení případů požití neboli use case. Případy použití umožňují například stanovit rozsah projektu a požadavky, nastínit, jak bude uživatel komunikovat se systémem nebo vizualizovat architekturu. V modelu jsou graficky znázorněny tři hlavní části, tj. systém, aktéři a případy použití pro synchronizaci dat.

Uvedený případ použití popisuje, kteří aktéři mohou synchronizaci dat spustit a jaké systémy budou ovlivněny.

Jednotlivé synchronizace probíhají automaticky v integračním bodě, synchronizaci může spustit také ručně uživatel s rolí administrátora z webového prostředí. Každý z kroků synchronizace ovlivní pouze daný systém nebo systémy.

Zdroj [36]

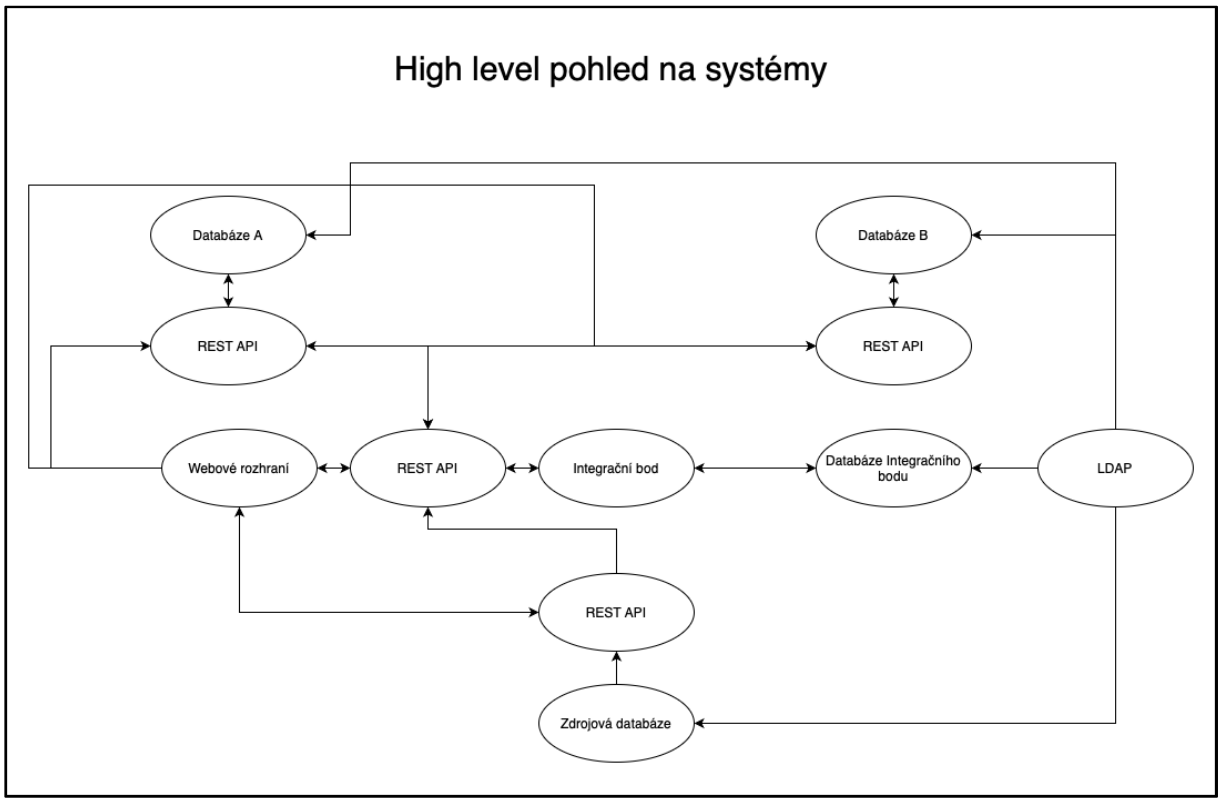


Obrázek 2: Model případu použití pro synchronizaci dat [Zdroj: vlastní tvorba]

7.3. High level pohled na systémy

Účelem high level pohledu na systémy je na první pohled zjistit, jak jsou systémy mezi sebou propojeny, pomocí jakých komponent komunikují a jaké komponenty lze v systémech očekávat.

Z diagramu lze vyčíst, že každá databáze má vlastní REST API, které je napojené na REST API integračního bodu. Integrační bod má vlastní databázi a webové rozhraní. Pro demonstraci funkčnosti integračního bodu je na webové rozhraní integračního bodu napojeno také REST API zdrojového systému a cílových systému. Všechny databáze jsou dále napojeny na LDAP.



Obrázek 3: High level pohled na systémy [Zdroj: vlastní tvorba]

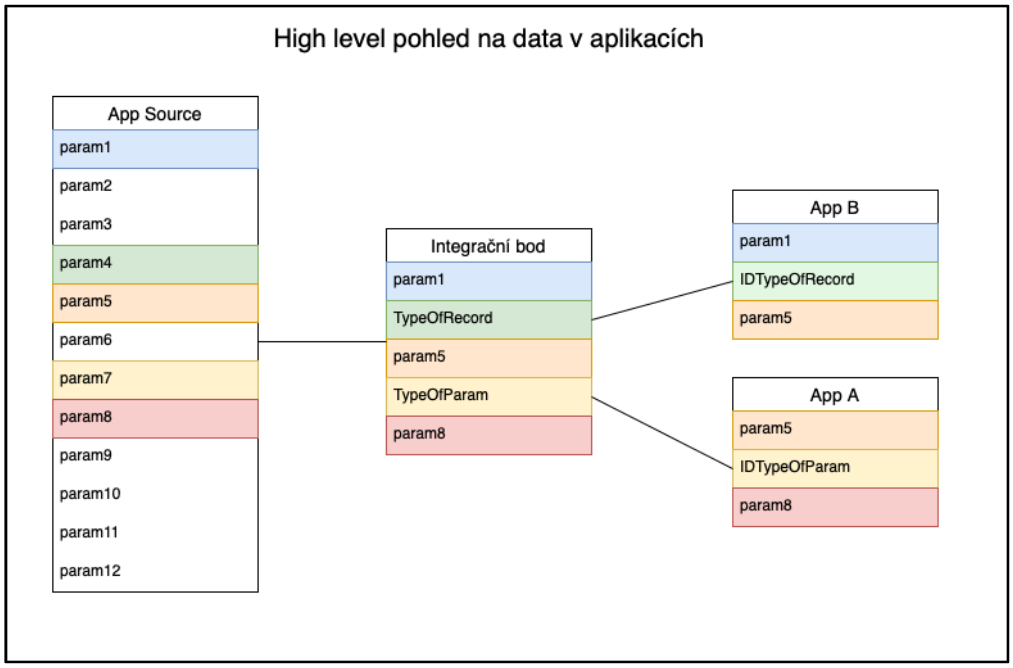
7.4. High level pohled na data v aplikacích

Účelem high level pohledu na data v aplikacích je na první pohled zjistit, jak jsou data mezi jednotlivými aplikace propojena a jakou mají mezi sebou závislost.

Z diagramu lze vyčíst, že zdrojová aplikace obsahuje 12 parametrů, které poskytuje integračnímu bodu. Integrační bod si uchovává pouze 5 hodnot, které následně poskytuje cílovým aplikacím. Vzhledem k tomu, že je potřeba vytvořit univerzální integraci mezi aplikacemi, tak v některých případech cílovým aplikacím nejsou poskytována celá data, ale pouze ID z navázaných číselníků v cílových aplikacích. Pro lepší rozeznání vazeb jsou barevně zvýrazněna data, která k sobě patří.

V tomto ukázkovém případě je param5 jedinečný pro všechny aplikace a je sdílený napříč všemi aplikacemi. Param4 se v integračním bodě transformuje na TypeOfRecord, který je jedinečný pro cílovou aplikaci App B. Obdobně se param7 v integračním bodě transformuje na TypeOfParam, který je jedinečný pro cílovou aplikaci App A. Pro hodnoty TypeOfRecord a TypeOfParam platí, že do cílových aplikací se odesílají pouze ID přiřazené z číselníků, které integrační bod získá z cílových aplikacích. Param1 a param8 nemají žádnou následnou funkci, vycházejí z ukázkového příkladu.

Vzhledem k tomu, že se jedná pouze o ukázková data, tak i jejich pojmenování je pouze ukázkové. Při integraci reálných aplikací by parametry byly pojmenovány dle konkrétních aplikací.

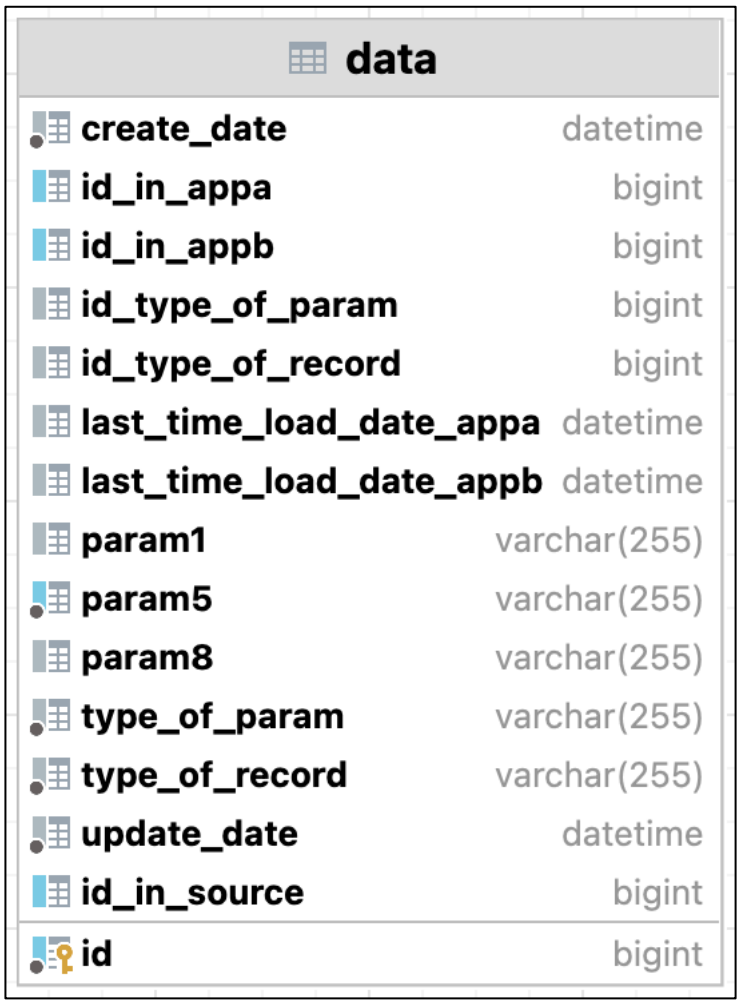


Obrázek 4: High level pohled na data v aplikacích [Zdroj: vlastní tvorba]

7.5. Databázový model integračního bodu

Databázový model je pohled na databázi, který definuje, jak jsou data organizována v rámci databáze, jak jsou data uložena a jak lze k datům přistupovat.

V databázovém modelu integračního bodu je kromě přenášených dat a ID ze zdrojové databáze také ID záznamů a ID, která jsou v cílových systémech přiřazena záznamům. Pro zamezení zbytečného nahrávání dat do cílových aplikací jsou u jednotlivých záznamů uvedena časová razítka, kdy byl záznam vytvořen, kdy byl naposledy aktualizován a kdy byl nahrán do cílové aplikace.



Obrázek 5: Databázový model pro data v integračním bodě [Zdroj: vlastní tvorba]

8. Implementace aplikace

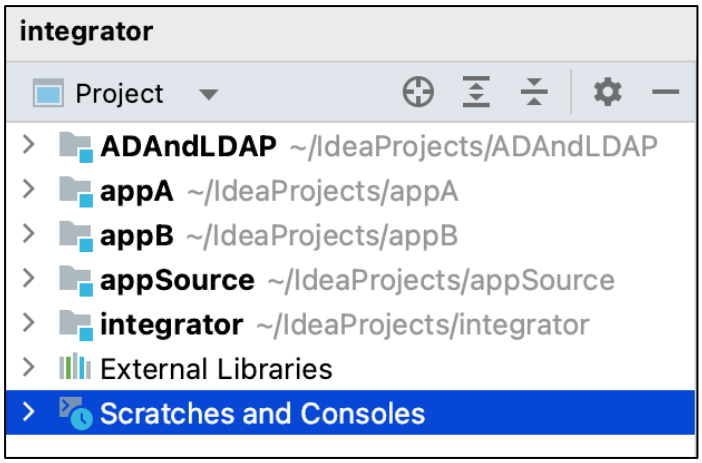
V diplomové práci jsou obsaženy tři virtualizované databáze. Nad každou databází je vlastní aplikace, která obsahuje API pro komunikaci. Všechny tři databáze spojuje integrační bod zajišťující automatickou integraci těchto systémů. Integrační bod obsahuje webové rozhraní pro monitorování systémů a základní operace s uživateli. V diplomové práci je implementován openLDAP pro správu uživatelů. Aplikace dále využívají několik architektonických a návrhových vzorů například: Model view controller, Microservice dessing pattern, Signle page app, Dependency injection, Singleton, Prototype, Builder, Proxy.

8.1. Použité jazyky

Pro tvorbu backendů byl použit jazyk Java s frameworkem Java Spring 3. Pro tvorbu frontendu byl použit JavaScript s knihovnou React.

8.2. Adresářová struktura projektu

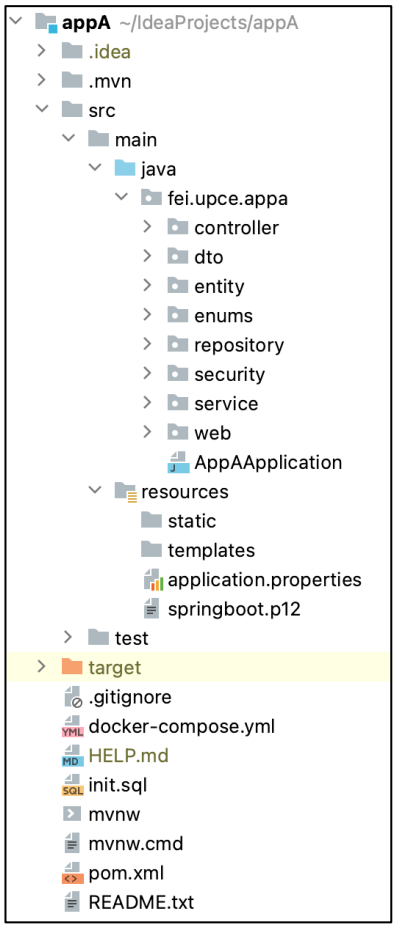
Každý ze systémů má vlastní repozitář, který je verzován na webu github.com.



Obrázek 6: Adresářová struktura projektů [Zdroj: vlastní tvorba]

8.2.1. Adresářová struktura zdrojové a cílových aplikací

V rámci zdrojové aplikace a cílových aplikací je rozdělení projektu takřka totožné.



Obrázek 7: Adresářová struktura App A [Zdroj: vlastní tvorba]

V projektu jsou obsaženy tyto důležité složky a soubory:

- **Složka controller** – obsahuje třídy zodpovědné za zpracování příchozích HTTP požadavků a poskytování odpovídajících odpovědí. Tyto třídy implementují metody s anotacemi @RequestMapping nebo jejími odvozeninami, jako jsou @GetMapping, @PostMapping, @PutMapping, @DeleteMapping atd. Každá metoda odpovídá konkrétnímu API, URL cestě a definuje logiku, která se má provést při přístupu k danému API.
- **Složka dto** – slouží k definici jednoduchých objektů (data transfer objects) pro přenos dat mezi vrstvami aplikace nebo mezi ostatními aplikacemi, jako jsou HTTP požadavky. Tato vrstva je použita k zapouzdření dat, které mají být přeneseny a k poskytnutí jednotného rozhraní pro přenos dat bez nutnosti složitějších objektů.
- **Složka entity** – obsahuje třídy, které reprezentují objekty modelu aplikace. Tyto třídy mapují tabulky v databázi nebo jiné třídy, které jsou použity.

- **Složka repository** – obsahuje rozhraní a implementaci pro práci s databází. Díky těmto třídám je možné provádět CRUD operace (vytvoření, čtení, aktualizace a mazání) nad databází.
- **Složka security** – obsahuje třídy pro zabezpečení aplikace a pro zabezpečení přenášených dat.
- **Složka service** – obsahuje třídy, které implementují aplikační logiku a poskytují rozhraní pro manipulaci s daty a provádění operací nad nimi.
- **Složka web** – obsahuje implementaci Swaggeru. Použití Swaggeru je způsob, jak jednoduše a rychle popsat všechny API a DTO pro vývojáře.
- **Složka WorkWithApiPush** – obsahuje třídy pro nahrání auditních dat do integrátoru.
- **Soubor AppAplication** – obsahuje konfiguraci a inicializaci aplikace a definuje základní chování a konfiguraci aplikace.
- **Soubor application.properties** – obsahuje konfigurační informace pro aplikaci, jako je připojení k databázi, nastavení zabezpečení pomocí JWT tokenu, SSL konfigurace, nastavení pro odesílání e-mailů nebo klíče pro symetrické a asymetrické šifrování.
- **Soubor springboot.p12** – obsahuje certifikát využívaný pro SSL/TLS šifrování.

8.2.2. OpenLDAP

V rámci projektu ADAndLDAP jsou pouze dva soubory, a to soubory pro práci s dockerem.

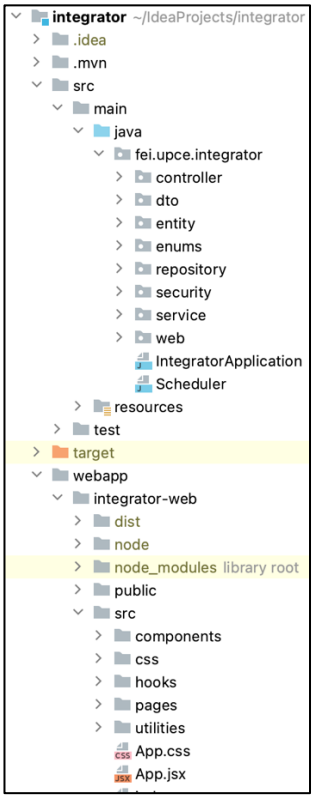
Obrázek 8: Adresářová struktura OpenLDAP [Zdroj: vlastní tvorba]

Soubory obsažené v projektu:

- **Soubor docker-compose.yml** – obsahuje definice docker kontejneru, včetně nastavení kontejnerů, sítí, svazků a dalších parametrů kontejneru.
- **Soubor init.ldif** – obsahuje základní záznamy používané při inicializaci LDAP, jako například uživatelské účty, skupiny, oprávnění a další atributy.

8.2.3. Adresářová struktura integračního bodu

V rámci integračního bodu jsou stejné složky, jako ve zdrojové a cílových aplikacích a také například vnitřní fungování integrace, spolupráce s LDAP a pro webového rozhraní.



Obrázek 9: Adresářová struktura integračního bodu [Zdroj: vlastní tvorba]

V projektu jsou obsaženy tyto důležité složky:

- **Složka components** – obsahuje opakovaně používané komponenty, jako jsou navigační panel, patička nebo nastavení zabezpečené routy.
- **Složka css** – obsahuje kaskádové styly pro jednotlivé stránky.
- **Složka pages** – obsahuje jednotlivé stránky.
- **Složka utilities** – obsahuje pomocné funkce, jako jsou volání API, kontrola JWT tokenu a jiné.
- **Soubor Scheduler** – obsahuje metodu pro automatickou synchronizaci dat v určeném časovém intervalu.

8.3. Virtualizace

Všechny databáze a openLDAP jsou virtualizovány pomocí Dockeru. Nespornou výhodou Dockeru je udržování image jednotlivých systémů od výrobců, a tudíž je předejito možným bezpečnostním hrozbám z důvodu neaktuálnosti systémů. Pro správnou funkčnost praktické části diplomové práce bylo potřeba vytvořit vlastní síť, do které byly přidány systémy. Díky tomu jsou všechny databáze ve stejné síti jako openLDAP.

Příkaz pro vytvoření sítě:

docker network create private-network

Všechny virtualizované systémy jsou nastaveny pomocí souboru docker-compose.yml. Spuštění image tak, aby respektoval docker-compose.yml anebo změny v něm je potřeba provést pomocí příkazu v příkazové řádce daného systému.

Příkaz pro spuštění kontejneru:

docker-compose up

```yaml
version: '3'

services:

  mysql-source:
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_USER: user
      MYSQL_PASSWORD: user
      MYSQL_DATABASE: source
      MYSQL_ROOT_HOST: '%'
    ports:
      - "3306:3306"
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
      - mysql-source:/var/lib/mysql
    command: ['--init-file', '/docker-entrypoint-initdb.d/init.sql', '--default-authentication-plugin=mysql_native_password']
    networks:
      - private-network

volumes:
  mysql-source:
    driver: local

networks:
  private-network:
    external: true
    name: private-network
```

Obrázek 10: Docker compose pro databáze [Zdroj: vlastní tvorba]

```yaml
version: '3'

services:
  openldap:
    image: osixia/openldap:latest
    command: --copy-service
    environment:
      LDAP_ORGANISATION: "Fei.upce"
      LDAP_DOMAIN: "fei.upce"
      LDAP_ADMIN_PASSWORD: adminpassword
    networks:
      - private-network
    ports:
      - "389:389"
      - "636:636"
    volumes:
      - ./init.ldif:/container/service/slapd/assets/config/bootstrap/ldif/50-bootstrap.ldif

  phpldapadmin:
    image: osixia/phpldapadmin:latest
    environment:
      PHPLDAPADMIN_LDAP_HOSTS: "openldap"
      PHPLDAPADMIN_HTTPS: "false"
    networks:
      - private-network
    ports:
      - "8086:80"
    depends_on:
      - openldap
```

Obrázek 11: Docker compose pro OpenLDAP [Zdroj: vlastní tvorba]

Pomocí atributu volumes lze přidat soubory do image dockeru a správným příkazem lze mít již při prvním startu image požadované položky v systému.

8.4. Autentizace a autorizace uživatelů

V rámci diplomové práce byla úvodní myšlenka napojit všechny systémy na LDAP a pomocí něj zajišťovat autentizaci a autorizaci.

Během tvorby praktické části diplomové práce však bylo zjištěno několik závažných objektivních důvodů, proč nelze realizovat napojení na LDAP.

Tyto důvody se týkaly napojení SQL databází na LDAP:

- **PostgreSQL** – databáze má oficiální image pro docker, avšak v image chybí plugin pg_ldap, který slouží k namapování LDAP pro databázi. Tento plugin nebylo možné následně do docker image doinstalovat.
- **MySQL** – databáze MySQL lze napojit na LDAP, avšak pouze v enterprise edici. Enterprise edice nemá oficiální docker image a licence stojí kolem 5400 dolarů na rok.
- **Oracle** – není oficiální docker image pouze pro databázi, ale jen pro celý systém. Pro využívání kontejneru celého systému je potřeba placená licence.
- **SQLite** – databáze nemá oficiální docker image.
- **MarianDB** – databáze má oficiální image pro docker, avšak stejně jako v případě PostgreSQL, chybí plugin pam_ldap. Ani v případě MarianDB nebylo možné plugin doinstalovat.
- **Microsoft SQL** – databáze nemá docker image pro ARM procesory.

Jako alternativní řešení byla v rámci diplomové práce implementována autentizace a autorizace pomocí JWT tokenů. Každá databáze má implementovány své JWT tokeny a pro správu všech uživatelů je použita webová stránka v rámci integračního bodu. Z integračního bodu jsou spravováni uživatelé v openLDAP. Po spuštění systému je správa LDAP dostupná na stránce *http://localhost:8086/*.

```
@Service
public class AddUserServiceImpl implements AddUserService {
    1 usage
    @Value("${url.provide.ldap}")
    private String provideUrl;
    1 usage
    @Value("${sec.config-auth.ldap}")
    private String securityAuthentication;
    1 usage
    @Value("${sec.principal.ldap}")
    private String securityPrincipal;
    1 usage
    @Value("${sec.credential.ldap}")
    private String securityCredential;

    1 usage
    @Override
    public void addUserToLDAP(SignUpDTO request) {
        String userDN = "uid=" + request.getFirstName().toLowerCase().charAt(0) + request.g
        Hashtable<String, String> env = new Hashtable<>();
        env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
        env.put(Context.PROVIDER_URL, provideUrl);
        env.put(Context.SECURITY_AUTHENTICATION, securityAuthentication);
        env.put(Context.SECURITY_PRINCIPAL, securityPrincipal);
        env.put(Context.SECURITY_CREDENTIALS, securityCredential);

        DirContext context = null;

        try {
            context = new InitialDirContext(env);
```

Obrázek 12: Ukázka metody pro přidání uživatele do LDAP [Zdroj: vlastní tvorba]

Pokud by bylo možné napojit aplikace a databáze na LDAP, tak by autentizace a autorizace probíhala vůči LDAP a uživatelským objektům v LDAP. Následně by byl řízen přístup pomocí rolí v LDAP.

8.5. Způsob komunikace mezi aplikacemi

Zdrojová aplikace, cílové aplikace, integrační bod i LDAP mezi sebou komunikují výhradně pomocí API. Dále integrační bod obsahuje API pro komunikaci s webovým rozhraním. Tyto API nenesou stejný standard jako API pro synchronizaci, protože mají pouze demonstrační účel. V reálném použití by těchto API nebylo potřeba, protože by tyto funkce nesly aplikace a Active Directory. Kontrola duplicitních záznamu při zadávání z webového rozhraní je řešena ve webovém rozhraní.

```java
@RestController
@RequestMapping("/api/v1/auth")
@RequiredArgsConstructor
public class AuthController {
    1 usage
    private final AuthService authService;

    no usages
    @PostMapping("/signin")
    public ResponseEntity<?> signin(@RequestBody SignInDTO request) {
        return ResponseEntity.ok(authService.signin(request));
    }
```

Obrázek 13: API pro získání JWT tokenu z integračního bodu [Zdroj: vlastní tvorba]

8.6. Metody zabezpečení

V diplomové práci jsou implementovány bezpečnostní opatření, jako je využití JWT tokenů, SSL certifikáty, limitování počtu přístupu na API, šifrování dat při přenosu mezi jednotlivými API, auditování událostí v jednotlivých aplikacích, výstrahy pomocí emailu v případě neočekávané události nebo zabezpečení CORS.

8.6.1. Zabezpečení komunikace mezi systémy

Přístup na jednotlivé API endpointy jsou chráněné vyžadováním autentizace a autorizace pomocí JWT tokenů. API endpointy jsou dále přístupné pouze pomocí protokolu https s SSL certifikátem a limitováním počtu provolání jednotlivých API. Samotný přenos dat je šifrován asymetrickým šifrováním pomocí veřejného a soukromého klíče každé aplikace a integračního bodu. Zabezpečení CORS (Cross-Origin Resource Sharing) je vyžadováno na všechny API endpointy.

- **JWT tokeny** – vybraná API mají limitaci přístupu v podobě autentizace, autorizace uživatele a role daného uživatel, který může provolat dané API. Využívá se preautorizace, jako první po provolání API dojde k ověření platnosti JWT tokenu, uživatele a jeho role.

```java
@RestController
@RequestMapping("/api/v1/generic")
public class GenericController {
    no usages
    @GetMapping("/ping")
    @PreAuthorize("hasRole('READER') or hasRole('FULL') or hasRole('ADMIN')")
    public ResponseEntity<?> ping() { return ResponseEntity.ok().body(new StatusDTO(Status.running)); }
}
```

Obrázek 14: API integračního bodu pro kontrolu, zda systém funguje [Zdroj: vlastní tvorba]

```
if (StringUtils.isNotEmpty(userEmail) && SecurityContextHolder.getContext().getAuthentication() == null) {
    UserDetails userDetails = userService.userDetailsService().loadUserByUsername(userEmail);
    if (jwtService.isTokenValid(jwt, userDetails)) {
        SecurityContext context = SecurityContextHolder.createEmptyContext();
        UsernamePasswordAuthenticationToken authToken = new UsernamePasswordAuthenticationToken(
                userDetails,  credentials: null, userDetails.getAuthorities());
        authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
        context.setAuthentication(authToken);
        SecurityContextHolder.setContext(context);
    }
}
```

Obrázek 15: Autentizace a autorizace uživatele v integračním bodě [Zdroj: vlastní tvorba]

- **Protokol HTTPS** – všechny URL adresy v rámci diplomové práce jsou zabezpečeny self sign SSL certifikáty. Díky tomu, že se jedná o certifikáty mnou podepsané, tak se pro aplikace i integrační bod jeví certifikát jako nedůvěryhodný.

```
# SSL Configuration
server.ssl.enabled=true
server.ssl.key-alias=springboot
server.ssl.key-store=classpath:springboot.p12
server.ssl.key-store-password=password
```

Obrázek 16: Konfigurace SSL pro integrační bod [Zdroj: vlastní tvorba]

Pokud by byl certifikát vydán autorizační autoritou nebo by byl certifikát importován do systému nebo by byl v rámci firmy implementován pro systémy důvěryhodný certifikát, tak by nebylo potřeba ochranu obcházet při provolávání API ze všech backendů.

```
@Override
public void run(String... args) throws Exception {
    SSLContext sslContext = SSLContext.getInstance( protocol: "TLS");
    sslContext.init( km: null, new TrustManager[]{new X509TrustManager() {
        public X509Certificate[] getAcceptedIssuers() { return null; }

        public void checkClientTrusted(X509Certificate[] certs, String authType) {
        }

        public void checkServerTrusted(X509Certificate[] certs, String authType) {
        }
    }}, new java.security.SecureRandom());

    HttpsURLConnection.setDefaultSSLSocketFactory(sslContext.getSocketFactory());
    HttpsURLConnection.setDefaultHostnameVerifier((hostname, session) -> true);
}
```

Obrázek 17: Nastavení důvěryhodnosti certifikátů [Zdroj: vlastní tvorba]

Každý certifikát má pouze omezenou dobu platnosti, a pokud jeho platnost vyprší, tak je možné vygenerovat nový v rámci operačního systému.

Příkaz pro vygenerování nového self sign certifikátu pomocí příkazové řádky:

keytool -genkeypair -alias springboot -keyalg RSA -keysize 4096 -storetype PKCS12 -keystore springboot.p12 -validity 110 -storepass password

- **Limitování počtu přístupu k API** – vybraná API mají omezený počet přístupů za sekundu za účelem předejití zahlcení aplikací požadavky. Tato funkce je vypnuta pro technického uživatele, který slouží pouze k synchronizaci v rámci integrační služby.

```
@SneakyThrows
@GetMapping("")
@PreAuthorize("hasRole('READER') or hasRole('FULL') or hasRole('ADMIN')")
public ResponseEntity<?> showAllSourceData() {
    String userEmail = userService.getCurrentUserEmail();
    if (userService.isTechUser(userEmail) || rateLimiter.tryAcquire()) {
        List<Optional<SourceData>> sourceDataList = sourceDataService.getSourceData();
        return ResponseEntity.ok(encryptValuesOnly(sourceDataList, integratorPublicKeyBase64));
```

Obrázek 18: Omezení počtu přístupů k API [Zdroj: vlastní tvorba]

- **Šifrování přenášených dat** – data přenášená v rámci integračního procesu jsou zašifrována asymetrickým RSA šifrováním. Každá aplikace a integrační bod mají vlastní pár klíčů. Veřejný, který je dostupný ostatním systémům a soukromý, který je dostupný pouze cílové aplikaci. Při odesílání dat jsou data zašifrována pomocí veřejného klíče a po přijetí jsou data rozšifrována soukromým klíčem. Veřejným klíčem není možné data rozšifrovat.

Zdroj [37]

```
private String encryptData(long data, String publicKeyBase64) throws Exception {
    byte[] publicKeyBytes = Base64.getDecoder().decode(publicKeyBase64);
    PublicKey publicKey = KeyFactory.getInstance( algorithm: "RSA").generatePublic(new X509EncodedKeySpec(publicKeyBytes));

    Cipher cipher = Cipher.getInstance( transformation: "RSA");
    cipher.init(Cipher.ENCRYPT_MODE, publicKey);

    ByteBuffer buffer = ByteBuffer.allocate(Long.BYTES);
    buffer.putLong(data);
    byte[] encryptedBytes = cipher.doFinal(buffer.array());
    return Base64.getEncoder().encodeToString(encryptedBytes);
}
```

Obrázek 19: Zašifrování dat v integračním bodě [Zdroj: vlastní tvorba]

```java
private SourceDataDTO decryptSourceData(String decryptSourceData, String privateKeyBase64) throws Exception{
    SourceDataDTO sourceDataDTO;
    try {
        String[] segments = decryptSourceData.split( regex: ", ");
        sourceDataDTO = new SourceDataDTO();
        for (String segment : segments) {
            DecryptResult decrypt = decryptGenericService.convertEncryptToDecrypt(segment, privateKeyBase64);
            setParamValueSourceData(sourceDataDTO, decrypt.getParamName(), decrypt.getDecryptedBytes());
        }
    } catch (IllegalBlockSizeException | NoSuchPaddingException | NoSuchAlgorithmException |
             InvalidKeySpecException | BadPaddingException | InvalidKeyException ex) {
        throw new RuntimeException(ex);
    }
    return sourceDataDTO;
}
```

Obrázek 20: Dešifrování dat v integračním bodě [Zdroj: vlastní tvorba]

```
# asymmetric private key
security.receiver.private-key=MIIEvgIBADANBgkqhkiG9w
# asymmetric appb public key
security.receiver.appb-public-key=MIIBIjANBgkqhkiG9w
# asymmetric appb public key
security.receiver.appa-public-key=MIIBIjANBgkqhkiG9w
```

Obrázek 21: Definice proměnných pro asymetrické šifrování [Zdroj: vlastní tvorba]

```
"id:b1jZ322rlHBd4mWwq3e3g0fIf+ZxNaKTaSqoI9NIj8SYdZUNxwxFjOCLaKJqUvtYWWLE5csCiMeOwAaKwm
    +PEUSwHM9yhKGjvueAwMsQClvzIU7ot89pNAD1jVzc7H2VF/5TuSzhAgPtdXyszHwh
    +ZXG3UictUOVBpeHoU7eYtdoWHEK7ggpowN/Bp/p21+PoREEtTTp8mI0eRhzna+g9Y+Oy85DiA4Yx2d7FwrXWOY4cEJ/
    hkFnHWV0C31KCl5Txw0KOHETXidIn9DJukgXHq172FOgLAYFnIcsaEWigermckyM4hgA9zl03D
    +AXt3T6kHYDbx8NTjkjS8NB9u9dw==, param1:PoN9whcO0qQJRZeSWc5m+8g6VNGy9MzftWy4d7Gknw5EqEuYHx+c
    +J5aX3rQWE60SG/xcvqRUKkFCt0iwBIj+oRC4bmepg+N/daClDMPUkBXXQoA+xvwC6XNhY4uF+kGsAHcTyJRlw9d3wG9b1GR/
    flw8J3ZXl+WdCOLnYU7NYlywSr38unjSLQGmPuuZ2zkvrVYm3qrKz7znzyqD4OFjjm63+/+3PuJhPMYF/
    y2TuJte6YGrudLKtnO31BqHYv3ME7GXatpekKxx1F2/ToyTZlyJ1CqYV5U4fVL9J5r81cdsOEqsx04/
    w0A52v7jLktk84CPhOAApqv31jB7aYSpg==,
```

Obrázek 22: Ukázka zašifrovaných přenášených dat [Zdroj: vlastní tvorba]

8.6.2. Auditní a výstražný systém

Integrační bod obsahuje auditní systém pro zachovávání logů. Pokud nastane neočekávaná situace v aplikacích nebo v integračním bodě, log je nahrán na API integračního bodu a je poté uložen do databáze integračního bodu. Mimo logů zasílá každá aplikace a integrační bod email o neočekávané situaci na vybranou adresu.

- **Auditní systém** – auditní systém je umístěn v integračním bodě. Pokud by integrační bod nebyl funkční, logy by se neukládaly. Pokud by toto řešení bylo integrováno ve firemním prostředí, kde dochází ke zpracovávání velkého objemu dat, byla by z výkonnostních důvodů data ukládána například pomocí ELK (Elasticsearch, Logstash, Kibana), Apache Kafka nebo Splunk. Tyto systémy jsou pro zpracování velkého objemu navrženy a ve firemních systémech využívány.

```
@Transactional(noRollbackFor = Exception.class)
public void saveAudit(Audit audit) {
    Audit record = new Audit();
    record.setSource(audit.getSource());
    record.setInfo(audit.getInfo());
    record.setAddInfo(audit.getAddInfo());
    record.setDate(new Date());
    auditRepository.save(record);
}
```

Obrázek 23: Metoda pro uložení auditních logů v integračním bodě [Zdroj: vlastní tvorba]

- **Výstražný systém** – výstražný systém je řešen pomocí zasílání e-mailů. E-maily se zasílají pomocí schránky na webové stránce *https://www.centrum.cz*.

  Do každé metody pro zasílání e-mailů je implementováno vytvoření samostatného vlákna pro daný systém, a to z důvodu omezeného počtu zaslaných e-mailů za určitou časovou kvótu. Proto je potřeba zasílání e-mailů zpomalit. Pokud by byl výstražný systém implementován v rámci skutečného produkčního řešení, tak by se pro zasílání zpráv používal například mailtrap.

```java
private void startTaskExecutor() {
    scheduler.scheduleAtFixedRate(() -> {
        try {
            if (taskQueue.size() > 1) {
                TimeUnit.SECONDS.sleep( timeout: 3);
                while (!taskQueue.isEmpty()) {
                    Runnable task = taskQueue.poll();
                    task.run();
                }
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }, initialDelay: 0, period: 1, TimeUnit.SECONDS);
}

14 usages
public void sendEmail(String to, String subject, String text) {
    taskQueue.offer(() -> {
        SimpleMailMessage message = new SimpleMailMessage();
        message.setFrom("integratorservicefeiupce@centrum.cz");
        message.setTo(to);
        message.setSubject(subject);
        message.setText(text);
        javaMailSender.send(message);
    });
}
```

Obrázek 24: Metoda pro zasílání výstražných emailů [Zdroj: vlastní tvorba]

8.6.3. Zabezpečení webového rozhraní

Kromě zabezpečení backendu je třeba zabezpečit také frontend (webové rozhraní). Frontend je opět zabezpečen SSL certifikátem. Veškerá komunikace probíhá z adresy localhost. Je tedy třeba vypnout zabezpečení CORS pro danou adresu, port a dané požadavky. Zabezpečení frontendu je řešeno pomocí JWT tokenů.

- **Protokol https** – jedná se o zabezpečení pomocí self sign SSL certifikátu. Je tedy třeba opět obejít bezpečnostní opatření a povolit i nedůvěryhodné certifikáty.

```javascript
process.env.NODE_TLS_REJECT_UNAUTHORIZED = '0';
const APPA_BACKEND_URL = "https://localhost:8082/api";
```

Obrázek 25: Povolení nedůvěryhodných certifikátů ve webovém rozhraní [Zdroj: vlastní tvorba]

- **Zabezpečení CORS** – zabezpečení CORS je ochrana proti útoku typu CSRF (Cross-Site Request Forgery), kdy útočník vytvoří podobnou webovou stránku a uživateli ji podstrčí za účelem ukrást citlivá data. Vzhledem k tomu, že se veškerá komunikace volá z adresy localhost a liší se pouze port, na kterém daná aplikace a integrační bod funguje, je potřeba vypnout CORS na určitou adresu a metody. Nastavení bylo vyzkoušeno a otestováno ve webovém prohlížeči Chrome.

```
.cors(cors -> cors.configurationSource(request -> {
    CorsConfiguration configuration = new CorsConfiguration();
    configuration.setAllowedOrigins(List.of( e1: "https://localhost:8083"));
    configuration.setAllowedMethods(Arrays.asList("GET", "PUT", "POST", "DELETE", "PATCH"));
    configuration.setAllowCredentials(true);
    configuration.setAllowedHeaders(List.of( e1: "*"));
    configuration.setExposedHeaders(Arrays.asList("X-Auth-Token", "Authorization", "Access-Control-Allow-Origin",
    return configuration;
}))
```

Obrázek 26: Vypnutí ochrany CORS pro přístup z integračního bodu [Zdroj: vlastní tvorba]

- **Zabečení přístupu na webové stránky** – zabezpečení pro přístup na každou webovou stránku v rámci integračního bodu je řešeno pomocí JWT tokenu. Po úspěšném přihlášení se do lokálního úložiště uloží JWT token, který uživatel získá po úspěšné autentizaci. Při přístupu na další webovou stránku systém ověří, zda existuje JWT token v lokálním úložišti, zda je validní a jakou má uživatel roli. Validita tokenu se ověřuje z pohledu expirace a vydavatele JWT tokenu. Po odhlášení z webového rozhraní je JWT token smazán z lokálního úložiště uživatele.

```
const ProtectedRoute = ({roles :any[] = [], children}) => {
    if (!isAuthenticated()) {
        return <Navigate to={AppRoute.ROOT} replace/>
    }
    if (checkToken()) {
        return <Navigate to={AppRoute.ROOT} replace/>
    }
    const userRole = tokenRole();
    if (roles.length > 0 && !roles.includes(userRole)) {
        return <Navigate to={AppRoute.ROOT} replace/>;
    }
    return children;
};
```

Obrázek 27: Kontrola platnosti JWT tokenu a role uživatele [Zdroj: vlastní tvorba]

```
return (
    <div className="bookmarks-panel">
        <a href={"../#"+AppRoute.DASHBOARD}>Dashboard</a>
        <a href={"../#"+AppRoute.AUDIT}>Audit</a>
        <a href={"../#"+AppRoute.INTEGRATOR_RECORDS}>Integrator Records</a>
        {userRole === "ROLE_ADMIN" && <a href={"../#"+AppRoute.USERS}>Users</a>}
        {userRole === "ROLE_ADMIN" && <a href={"../#"+AppRoute.USER_MANAGEMENT}>UserManagement</a>}
        {userRole === "ROLE_ADMIN" && <a href={"../#"+AppRoute.DATA_MANAGEMENT}>DataManagement</a>}
        {userRole === "ROLE_ADMIN" && <a href={"../#"+AppRoute.SYNC}>Manual Sync</a>}
        <button onClick={handleLogout}>Logout</button>

    </div>
);
```

Obrázek 28: Omezení přístupu uživatele k záložkám ve webovém rozhraní [Zdroj: vlastní tvorba]

```
const App = () => {
    return <QueryClientProvider client={queryClient}>
        <HashRouter>
            <Routes>
                <Route path={AppRoute.ROOT} element={<Login />} />
                <Route path={AppRoute.DASHBOARD} element={<ProtectedRoute><Dashboard /></ProtectedRoute>} />
                <Route path={AppRoute.AUDIT} element={<ProtectedRoute><Audit /></ProtectedRoute>} />
                <Route path={AppRoute.INTEGRATOR_RECORDS} element={<ProtectedRoute><IntegratorRecords /></ProtectedRoute>} />
                <Route path={AppRoute.USERS} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><Users /></ProtectedRoute>} />
                <Route path={AppRoute.USER_DETAILS} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><UserDetails /></ProtectedRoute>} />
                <Route path={AppRoute.USER_MANAGEMENT} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><UserManagement /></ProtectedRoute>} />
                <Route path={AppRoute.CREATE_USER} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><CreateUser /></ProtectedRoute>} />
                <Route path={AppRoute.CHANGE_PASSWORD} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><ChangePassword /></ProtectedRoute>} />
                <Route path={AppRoute.DELETE_USER} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><DeleteUser /></ProtectedRoute>} />
                <Route path={AppRoute.SYNC} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><Sync /></ProtectedRoute>} />
                <Route path={AppRoute.DATA_MANAGEMENT} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><DataManagement /></ProtectedRoute>} />
                <Route path={AppRoute.APP_B_TREE} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><AppBTree /></ProtectedRoute>} />
                <Route path={AppRoute.APP_B_TYPE_OF_RECORD} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><AppBTypeOfRecord /></ProtectedRoute>} />
                <Route path={AppRoute.CREATE_TYPE_OF_RECORD} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><CreateTypeOfRecord /></ProtectedRoute>} />
                <Route path={AppRoute.APP_B_RECORDS} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><AppBRecords /></ProtectedRoute>} />
                <Route path={AppRoute.APP_A_TREE} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><AppATree /></ProtectedRoute>} />
                <Route path={AppRoute.APP_A_RECORDS} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><AppARecords /></ProtectedRoute>} />
                <Route path={AppRoute.APP_A_TYPE_OF_PARAM} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><AppATypeOfParam /></ProtectedRoute>} />
                <Route path={AppRoute.CREATE_TYPE_OF_PARAM} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><CreateTypeOfParam /></ProtectedRoute>} />
                <Route path={AppRoute.SOURCE_DATA} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><SourceData /></ProtectedRoute>} />
                <Route path={AppRoute.CREATE_SOURCE_DATA} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><SourceAddData /></ProtectedRoute>} />
                <Route path={AppRoute.SOURCE_MODIFY_RECORD} element={<ProtectedRoute roles={["ROLE_ADMIN"]}><ModifySourceData /></ProtectedRoute>} />
            </Routes>
        </HashRouter>
    </QueryClientProvider>
};
```

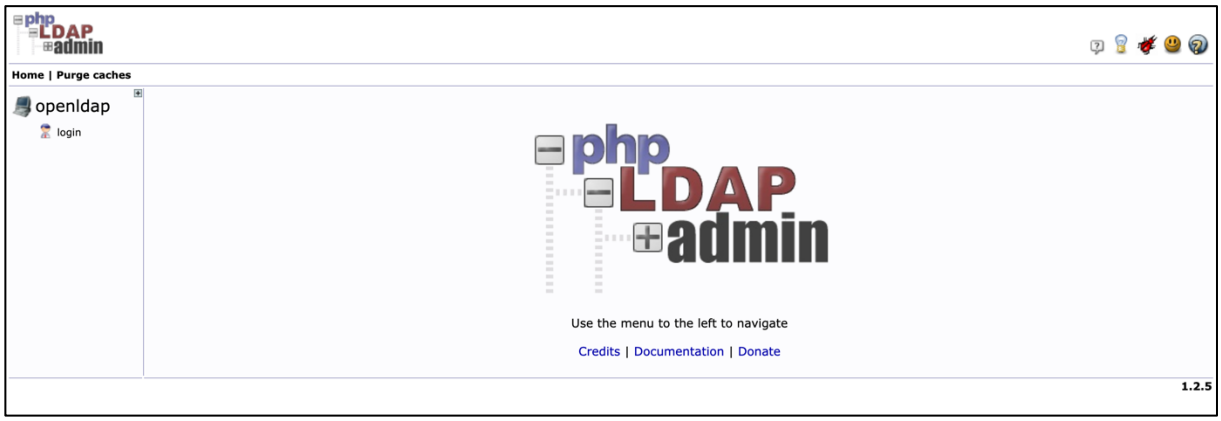
Obrázek 29: Zabezpečení webových stránek z pohledu validity a role JWT tokenu [Zdroj: vlastní tvorba]

8.7. Webové rozhraní systémů

Diplomová práce má dvě webová rozhraní, rozhraní openLDAP a rozhraní integračního bodu.

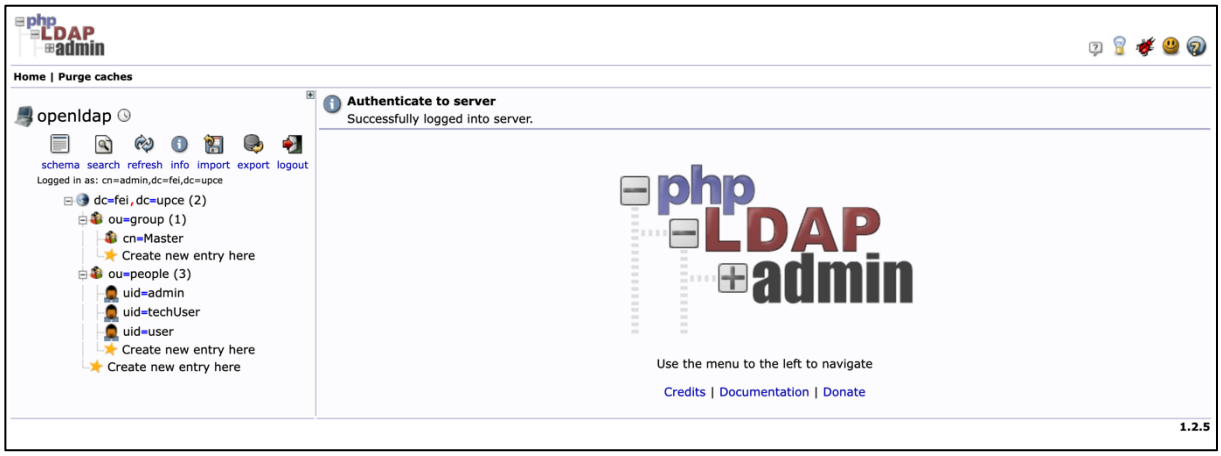
8.7.1. OpenLDAP

Rozhraní pro OpenLDAP je dostupné na adrese http://localhost:8086/.

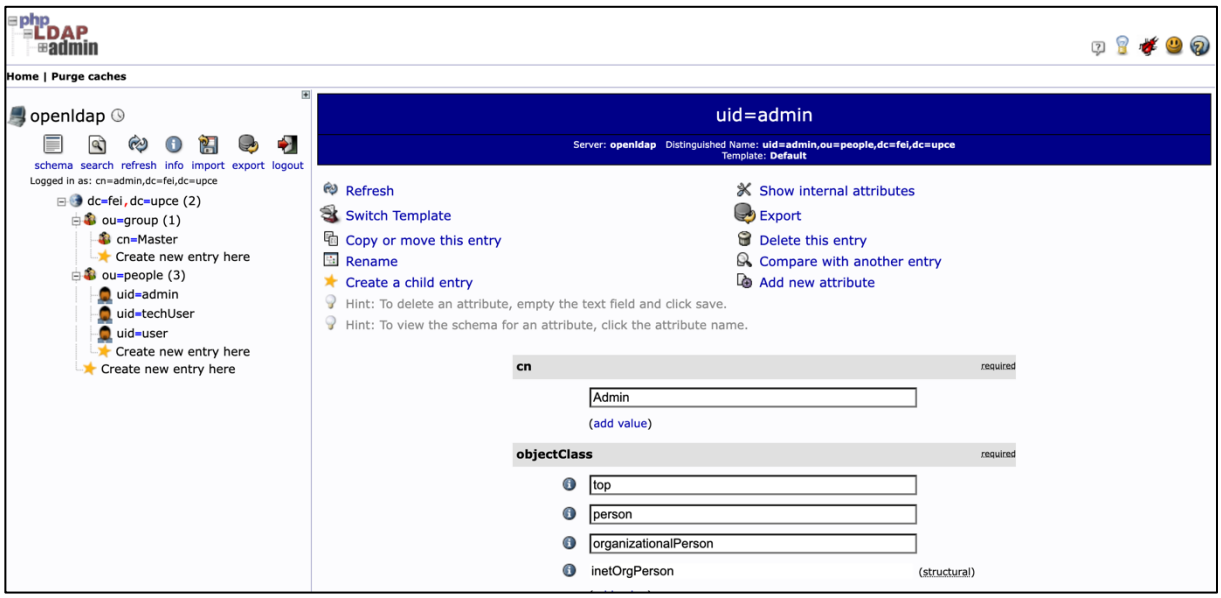


Obrázek 30: Úvodní webová stránka OpenLDAP [Zdroj: vlastní tvorba]

Na webové stránce je možná správa uživatelů z pohledu OpenLDAP. Dále je možné pracovat s rolemi jednotlivých skupin i uživatelů.



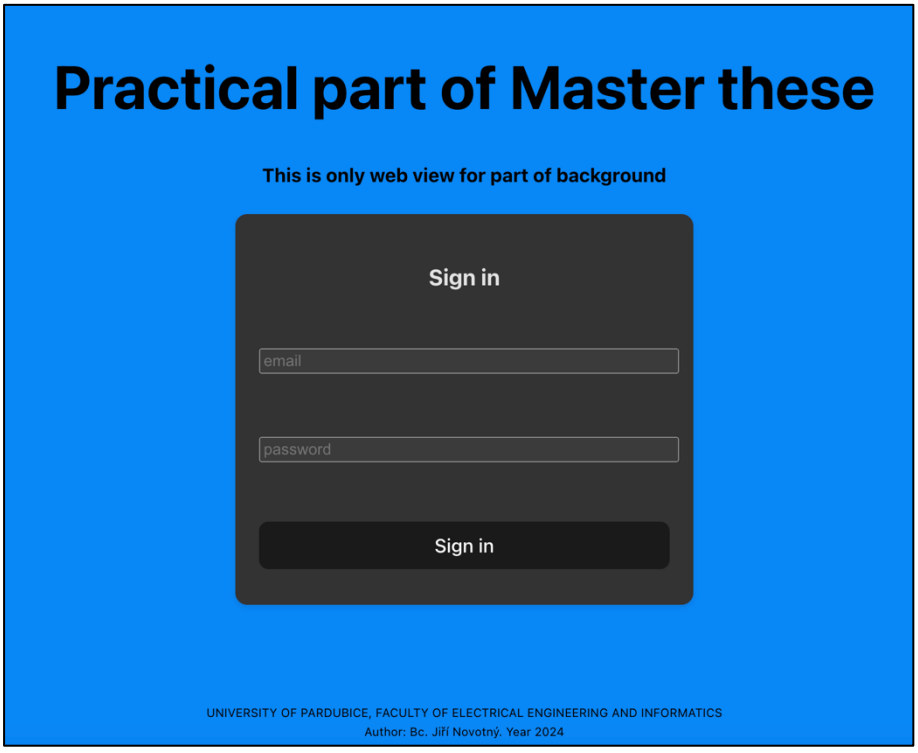
Obrázek 31: Webové rozhraní po přihlášení administrátora [Zdroj: vlastní tvorba]



Obrázek 32: Správa objektu administrátora [Zdroj: vlastní tvorba]

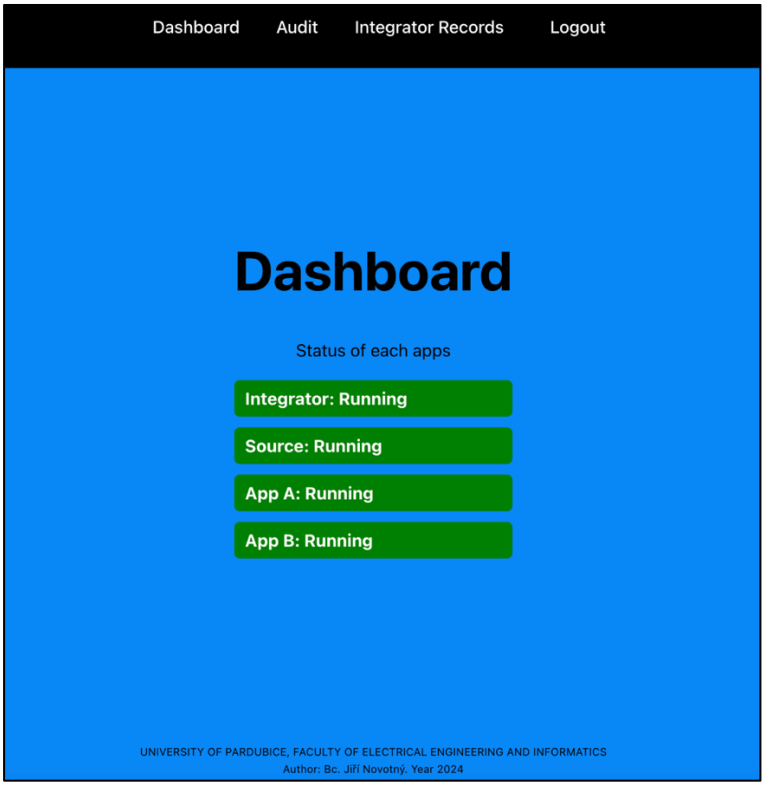
8.7.2. Integrační bod

Webové rozhraní integračního bodu je dostupné na adrese *https://localhost:8083/*. Pro komunikaci s webovým rozraním nejsou přenášená data zašifrována, protože webové rozhraní má pouze demonstrativní účel. Na adrese se po úspěšném přihlášení uživatele zobrazí dashboard o stavu aplikací a integračního bodu.

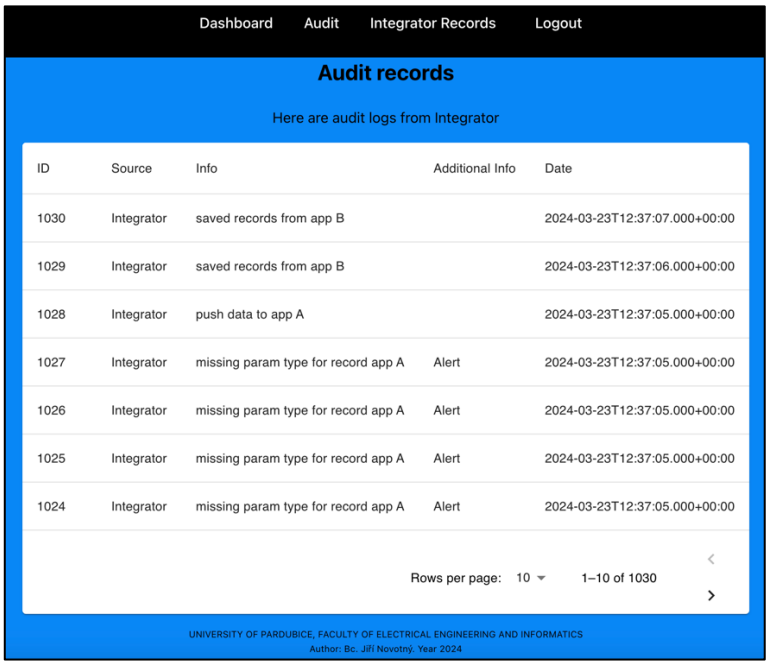


Obrázek 33: Úvodní stránka integračního bodu [Zdroj: vlastní tvorba]

Pokud je uživatel přihlášen jako běžný uživatel, má přístup pouze na webové stránky pro prohlížení stavu prostředí, auditních záznamů a záznamů v integračním bodě. V nabídce se uživateli zobrazí pouze uvedené položky. Způsob prezentace těchto záznamů je pro všechny stránky stejný.

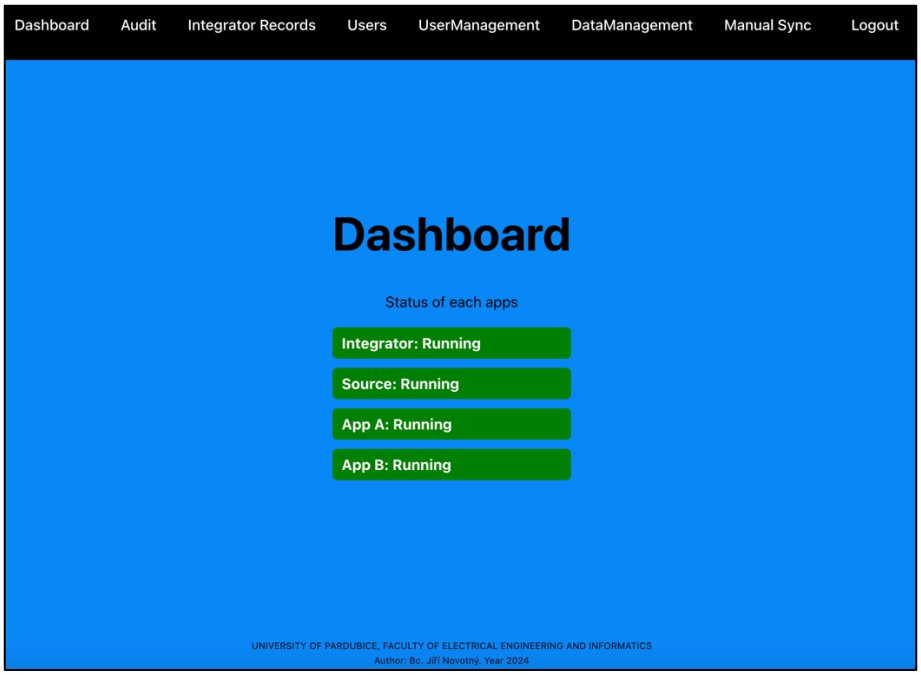


Obrázek 34: Webové rozhraní po úspěšném přihlášení běžného uživatele [Zdroj: vlastní tvorba]



Obrázek 35: Prohlížení auditních záznamů běžného uživatele [Zdroj: vlastní tvorba]

Pokud je uživatel přihlášen jako administrátor, má přístup na stránky pro úpravu rolí uživatelů. Styl je vizuálně stejný jako prohlížení záznamů, obsahuje navíc správu uživatelů a ruční synchronizaci systémů přes integrační bod.



Obrázek 36: Webové rozhraní po úspěšném přihlášení administrátora [Zdroj: vlastní tvorba]

Využití webového rozhraní diplomové práce je primárně demonstrováno v uživatelské příručce. Uživatelská příručka je přílohou diplomové práce.

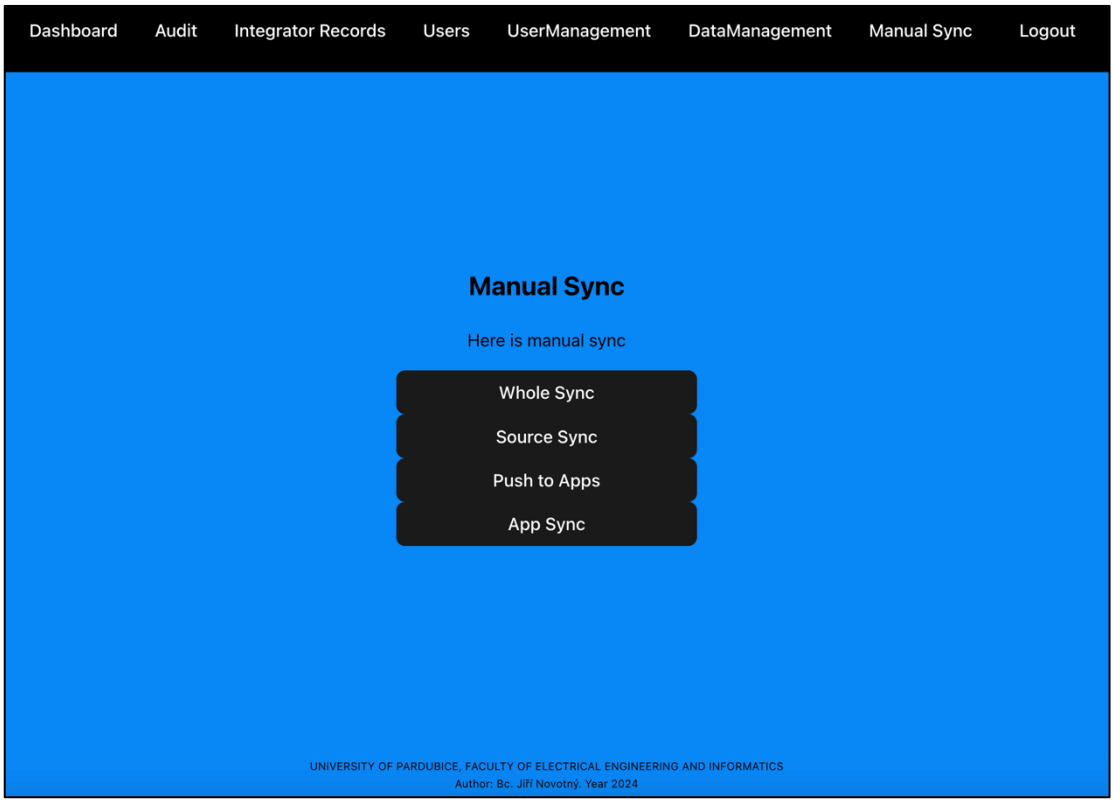
8.8. Synchronizace dat

Automatická synchronizace dat v integračním bodě, probíhá jednou za 10 minut a je řešena komponentou Scheduler, kterou nabízí Java Spring.

```java
@Component
@AllArgsConstructor
public class Scheduler {

    2 usages
    private final FetchAndSaveIntegratorService fetchAndSaveIntegratorService;

    1 usage
    private final PushFromIntegratorService pushFromIntegratorService;

    no usages
    @Scheduled(fixedDelay = 600000)
    public void fetchDataAndPush(){
        fetchAndSaveIntegratorService.initFetchFromSource();
        pushFromIntegratorService.pushDataToApp();
        fetchAndSaveIntegratorService.lastFetchFromApp();
    }
}
```

Obrázek 37: Automatická synchronizace dat v integračním bodě [Zdroj: vlastní tvorba]

Synchronizaci dat je možné vyvolat také manuálně, pomocí webového rozhraní integračního bodu.



Obrázek 38: Manuální synchronizace dat pomocí webového rozhraní [Zdroj: vlastní tvorba]

ZÁVĚR

Cílem diplomové práce byla implementace propojení databází za účelem synchronizace dat mezi nimi pomocí REST API. Propojení databází by umožňovalo data automaticky synchronizovat ze zdrojové databáze do ostatních databází bez nutnosti ”point to point” propojení každé databáze zvlášť na zdrojovou databázi.

Teoretická část diplomové práce se zabývá popisem technologie Active Directory a Lightweight Directory Access Protocol, porovnáním jednotlivých verzí, jejich nejdůležitějších funkcí a alternativ. Dále je teoretická část zaměřena na technologii aplikačního programového rozhraní s popisem historie, jeho alternativ a způsobech zabezpečení. Závěr teoretické části je věnován technologiím databází, virtualizace a konkurenčním řešením. Popsané technologie jsou užity v praktické části diplomové práce.

Praktická část diplomové práce je zaměřena na implementaci integračního bodu za účelem synchronizace dat mezi databázemi. Prvotním záměrem bylo implementovat technologii LDAP pro autentizaci a autorizaci. Vzhledem k okolnostem popsaným v diplomové práci nebylo možné tento záměr splnit, a proto byla implementována autentizace a autorizace pomocí JWT tokenů. JWT tokeny, stejně jako LDAP, umožňují implementovat řízení přístupu na základě přidělených rolí danému uživateli. LDAP je přesto zapracován do diplomové práce, a to pomocí verze openLDAP. Komunikace mezi databázemi a integračním bodem je řešena pomocí REST API. Zabezpečení komunikace je provedeno pomocí SSL certifikátů a přenášená data jsou zašifrována RSA šifrou.

Integrační bod pro propojení databází je vhodný do firemního prostředí. Vždy existují minimálně dva systémy, které si mezi sebou potřebují předávat data. V systémech ale nebývá implementována integrace vůči všem systémům. Lze předpokládat, že firmy budou řešit synchronizaci dat samostatně, dle svých podmínek, například vývojem vlastního integračního bodu.

Motivací k vypracování diplomové práce byla častá konfrontace s absencí funkčního propojení aplikací v praxi a z ní plynoucí nemožnost využití všech funkcí cílových aplikací nebo celých cílových aplikací. Řešení uvedená v diplomové práci lze užít v širokém spektru praxe. Mají potenciál zefektivnění synchronizace dat ze zdrojových systémů do ostatních systémů automaticky, pomocí integračního bodu.

POUŽITÁ LITERATURA

[1] CHAI, Wesley a Alexander GILLIS. *What is Active Directory and how does it work?* [online], 2021. [cit. 2023-12-29]. Dostupné z: *https://www.techtarget.com/searchwindowsserver/definition/Active-Directory*

[2] DESMOND, Brian; RICHARDS, Joe; ALLEN, Robbie a LOWE-NORRIS, Alistair G. *Active Directory*. 4th Edition. O'Reilly Media, 2008. ISBN 9780596554286.

[3] POSEY, Brien. *Active Directory: What Is It and How Does It Work?* [online], 2023 [cit. 2023-12-29]. Dostupné z: *https://www.itprotoday.com/active-directory/active-directory-what-it-and-how-does-it-work*

[4] Microsoft Corporation. *Operating System Property Values*. [online], 2022. [cit. 2023-12-29]. Dostupné z: *https://learn.microsoft.com/en-us/windows/win32/msi/operating-system-property-values?redirectedfrom=MSDN*

[5] WOOLSEY, Jeff. *Happy 30th Birthday Windows Server!* [online], 2023 [cit. 2023-12-29]. Dostupné z: *https://techcommunity.microsoft.com/t5/windows-server-essentials-and/30-years-of-windows-server/ba-p/3884810*

[6] HOPE, Computer. *Windows 2000*. [online], 2023 [cit. 2023-12-29]. Dostupné z: *https://www.computerhope.com/jargon/w/win2000.htm*

[7] PETRI, Daniel. *Overview of Windows Server 2003 – Standard Edition*. [online], 2009 [cit. 2023-12-29]. Dostupné z: *https://petri.com/overview_of_windows_server_2003_standard_edition/*

[8] HASSELL, Jonathan. *Learning Windows Server 2003*. O'Reilly Media, 2004. ISBN 9780596006242.

[9] KANADE, Vijay. *What Is Active Directory? Working, Importance, and Alternatives*. [online], 2023 [cit. 2023-12-29]. Dostupné z: *https://www.spiceworks.com/tech/networking/articles/what-is-active-directory/*

[10] Microsoft Corporation. *What Is Active Directory Lightweight Directory Services?* [online], 2018 [cit. 2023-12-30]. Dostupné z: *https://learn.microsoft.com/en-us/previous-versions/windows/desktop/adam/what-is-active-directory-lightweight-directory-services*

[11] Microsoft Corporation. *What is Active Directory Certificate Services?* [online], 2023 [cit. 2023-12-30]. Dostupné z: *https://learn.microsoft.com/en-us/windows-server/identity/ad-cs/active-directory-certificate-services-overview*

[12] Microsoft Corporation. *AD FS Overview*. [online]., 2023 [cit. 2023-12-30]. Dostupné z: *https://learn.microsoft.com/en-us/windows-server/identity/ad-fs/ad-fs-overview*

[13] Microsoft Corporation. *AD RMS Overview*. [online], 2018 [cit. 2023-12-30]. Dostupné z: *https://learn.microsoft.com/en-us/previous-versions/windows/desktop/adrms_sdk/ad-rms-overview*

[14] The Apache Software Foundation. *1.2 - Some Background. Directories, directory services and LDAP*. [online], 2013 [cit. 2023-12-30]. Dostupné z: *https://directory.apache.org/apacheds/basic-ug/1.2-some-background.html*

[15] OpenLDAP Foundation. *1. Introduction to OpenLDAP Directory Services* [online], 2021 [cit. 2023-12-30]. Dostupné z: *https://www.openldap.org/doc/admin24/intro.html*

[16] FreeIPA Team. *FreeIPA – Identity, Policy, Audit*. [online], 2007 [cit. 2023-12-30]. Dostupné z: *https://www.freeipa.org*

[17] Pegacat Software. *The JXplorer LDAP browser FAQ*. [online], 2004 [cit. 2023-12-30]. Dostupné z: *https://jxplorer.org/documents/faq.html*

[18] CARTER, G. *LDAP system administration*. Cambridge: O´Reilly, 2003. ISBN 9781565924918.

[19] BHARGAVA, Rajat. *What Is LDAP? The Ultimate Guide* [online], 2021 [cit. 2023-12-30]. Dostupné z: *https://jumpcloud.com/blog/what-is-ldap*

[20] MASSE, Mark. *REST API Design Rulebook*. Cambridge: O'Reilly Media, 2011. ISBN 9781449310509

[21] Amazon Web Services, Inc. *What is an API (Application Programming Interface)?* [online], 2021 [cit. 2023-12-30]. Dostupné z: *https://aws.amazon.com/what-is/api/*

[22] MIKULA, Kate. *The History and Evolution of APIs*. [online], 2023 [cit. 2023-12-30]. Dostupné z: *https://traefik.io/blog/the-history-and-evolution-of-apis/*

[23] DATA, Transparent. *Short history of API | From cabinet to big BOOM*. [online], 2022 [cit. 2023-12-30]. Dostupné z: *https://medium.com/transparent-data-eng/short-history-of-api-from-cabinet-to-big-boom-894c56b2c332*

[24] TEAM, SendGrid a Ayanna JULIEN. *What's a Webhook?* [online], 2023 [cit. 2024-01-28]. Dostupné z: *https://sendgrid.com/en-us/blog/whats-webhook*

[25] LIVENS, Jay. *What is a message queue? How an observability platform eases message queue monitoring*. [online], 2022 [cit. 2024-01-28]. Dostupné z: *https://www.dynatrace.com/news/blog/what-is-a-message-queue/*

[26] JUVILER, Jamie. *API Security Best Practices: 10+ Tips to Keep Your Data Safe*. [online], 2023 [cit. 2024-01-28]. Dostupné z: *https://blog.hubspot.com/website/api-security*

[27] ELLIS, Danielle R. *4 API Authentication Methods for a Secure REST API*. [online], 2024 [cit. 2024-01-28]. Dostupné z: *https://blog.hubspot.com/website/api-authentication*

[28] SANDIARSA, I K. *Adding an Auditing System into a Rest API*. [online], 2019 [cit. 2024-01-28]. Dostupné z: *https://medium.com/hackernoon/adding-an-auditing-system-into-a-rest-api-4fbb522240ea*

[29] YASAR, Kinza. *Web application firewall (WAF)*. [online], 2021 [cit. 2024-01-28]. Dostupné z: *https://www.techtarget.com/searchsecurity/definition/Web-application-firewall-WAF*

[30] DOWSETT, Chris. *What Is a Database?* [online], 2023 [cit. 2024-01-28]. Dostupné z: *https://builtin.com/data-science/database*

[31] LUTKEVICH, Ben. *Database (DB)*. [online], 2022 [cit. 2024-01-28]. Dostupné z: *https://www.techtarget.com/searchdatamanagement/definition/database*

[32] Quickbase. *A Timeline of Database History & Database Management*. [online], 2016 [cit. 2024-01-28]. Dostupné z: *https://www.quickbase.com/articles/timeline-of-database-history*

[33] WONG, Belle, ADITHAM, Kiran, ed. *What Is Virtualization? Definition, Benefits & Examples*. [online], 2023 [cit. 2024-03-08]. Dostupné z: *https://www.forbes.com/advisor/business/software/what-is-virtualization/*

[34] THEY MAKE DESIGN. *Virtualization in Web Development: Boosting Efficiency & Flexibility*. [online], 2023 [cit. 2024-03-08]. Dostupné z: *https://medium.com/theymakedesign/virtualization-in-web-development-f7c9c66f585*

[35] HOLMAN, Armond. *Virtualization and Docker Containers Simply Explained by a Junior DevOps Engineer*. [online], 2023 [cit. 2024-03-08]. Dostupné z: *https://medium.com/@armond10holman/virtualization-and-docker-containers-simply-explained-by-a-junior-devops-engineer-714de7801187*

[36] DALY, Nicky. *What Is a Use Case?* [online], 2022 [cit. 2024-03-08]. Dostupné z: *https://www.wrike.com/blog/what-is-a-use-case/*

[37] WALLS, Craig. Spring in Action. 5. New York: Manning, 2018. ISBN 9781617294945

PŘÍLOHY

Příloha A – Uživatelská příručka

PŘÍLOHA A – Uživatelská příručka

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Uživatelská příručka k integračnímu bodu

Bc. Jiří Novotný

Diplomová práce – Příloha A
2024

1

OBSAH

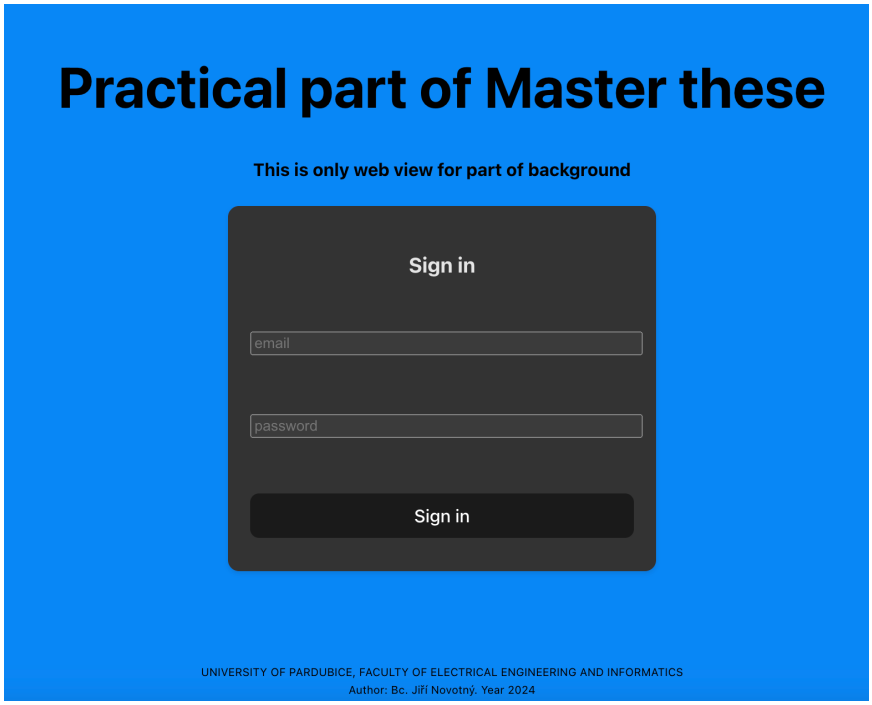
SEZNAM ILUSTRACÍ

ÚVOD

Uživatelská příručka popisuje část webového rozhraní diplomové práce Návrh a implementace aplikace pro propojení rozdílných API databází. Popsané webové rozhraní je implementováno jako součást aplikace integračního bodu. Webové rozhraní slouží ke zjištění stavu prostředí, prohlížení dat a auditních logů, správě uživatelů a jejich rolí a správě dat. Umožňuje vyvolávat ruční synchronizaci mezi systémy. Ve webovém rozhraní je použit anglický jazyk.

Přihlášení uživatele

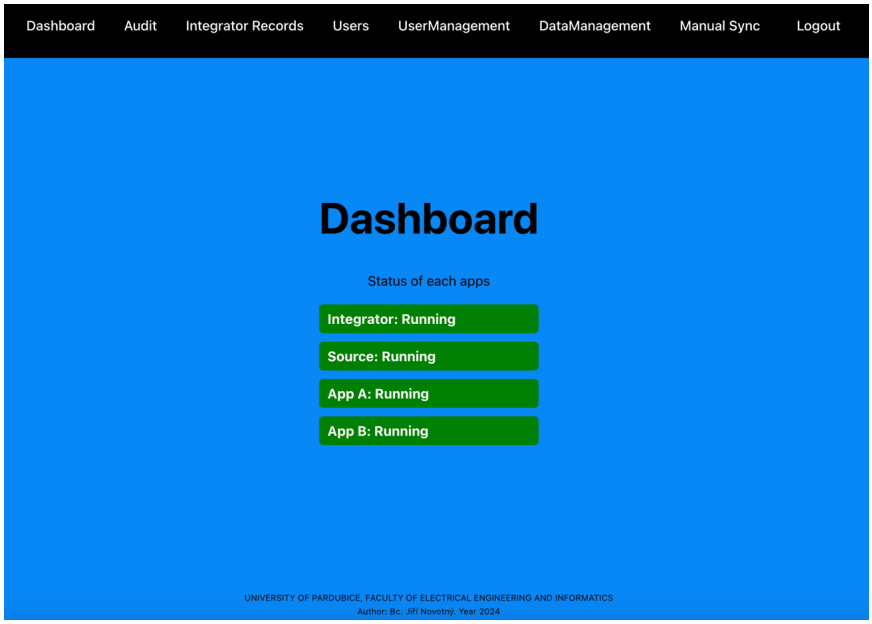
Webové rozhraní je dostupné na adrese https://localhost:8083/. Pro úspěšné přihlášení do uživatelského rozhraní je potřeba znát email a heslo k uživatelskému účtu.



Obrázek 1: Webová stránka pro přihlášení uživatele [Zdroj: vlastní tvorba]

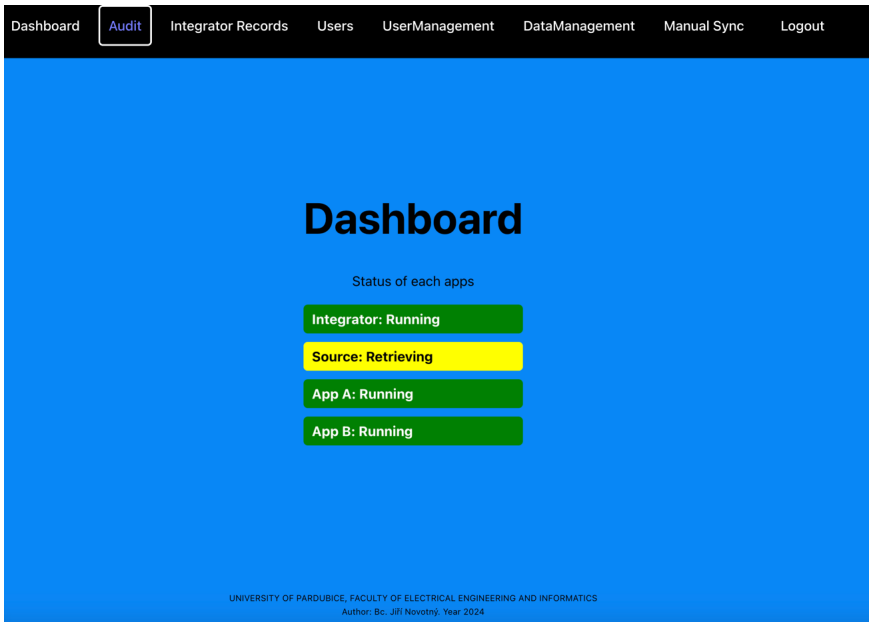
Stav aplikací

Po úspěšném přihlášení se zobrazí stav aplikací. Ve webovém rozhraní se pracuje s uživatelskými rolemi. Pokud je uživatel přihlášen k účtu s administrátorskými právy, dostane se ke všem záložkám a webovým stránkám. Pokud je uživatel přihlášen k účtu s jinou rolí, má v nabídce pouze omezený výběr. Tento uživatel má přístup na stránky sloužící k prohlížení stavu prostředí, nahlížení do auditních logů, nahlížení dat uložených v integračním bodě a možnosti odhlášení. Příručka je napsána z pohledu uživatele přihlášeného administrátorským účtem, který má přístup ke kompletní nabídce a všem webovým stránkám.



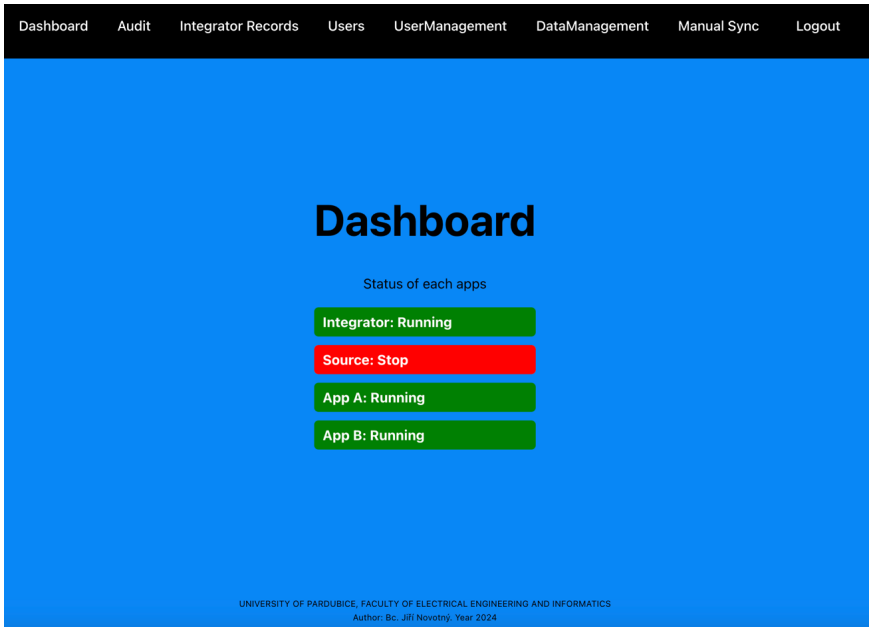
Obrázek 2: Webová stránka stavu prostředí, pokud fungují všechny aplikace [Zdroj: vlastní tvorba]

Není-li některá z aplikací funkční, nejprve se získává její stav. Během toho je ukazatel stavu dotčené aplikace vyžlucen.



Obrázek 3: Webová stránka stavu prostředí, pokud je získáván stav zdrojové aplikace [Zdroj: vlastní tvorba]

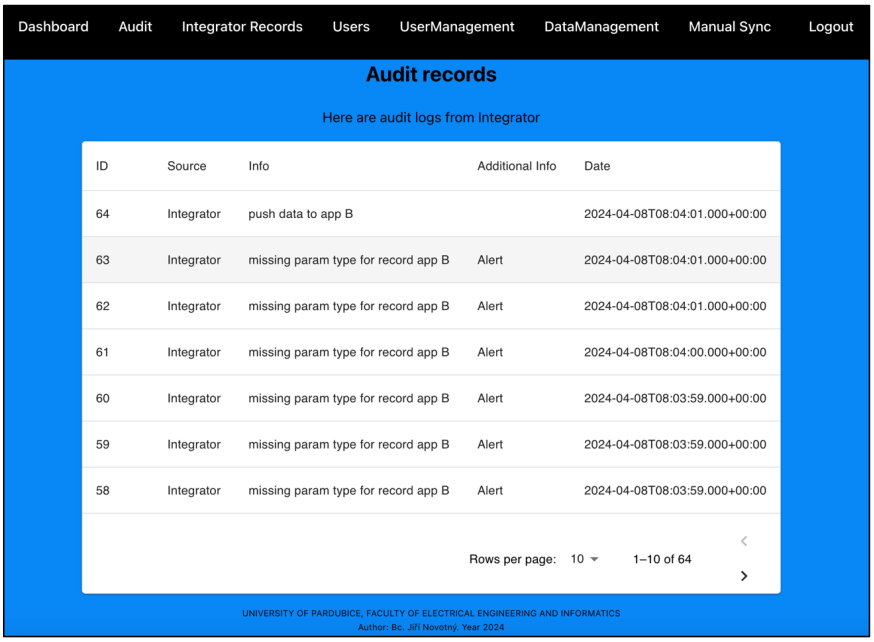
Pokud po uplynutí 5 sekund aplikace nezačne odpovídat, je stav aplikace označen jako nefunkční



Obrázek 4: Webová stránka stavu prostředí, pokud nefunguje zdrojová aplikace [Zdroj: vlastní tvorba]

Auditní záznamy

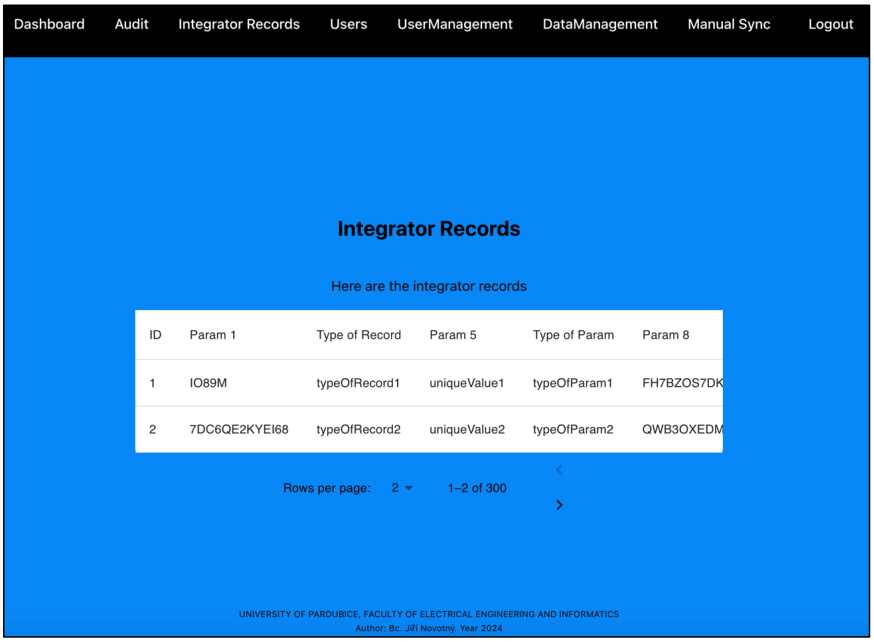
Webové rozhraní umožňuje prohlížet auditní záznamy všech aplikací. Auditní záznamy se ukládají v integračním bodě.



Obrázek 5: Webová stránka pro prohlížení auditních logů [Zdroj: vlastní tvorba]

Záznamy uložené v integračním bodě

Webové rozhraní umožňuje prohlížet uložené záznamy v integračním bodě.

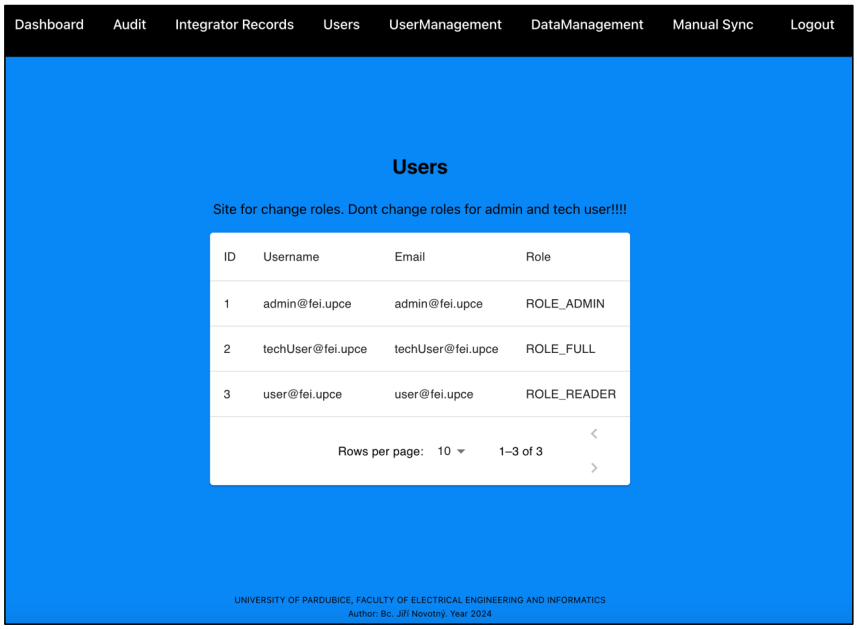


Obrázek 6: Webová stránka pro prohlížení dat v integračním bodě [Zdroj: vlastní tvorba]

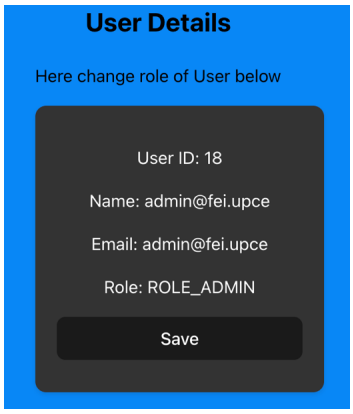
Další záložky a stránky jsou přístupné pouze pro účty administrátorů.

Správa rolí uživatelů

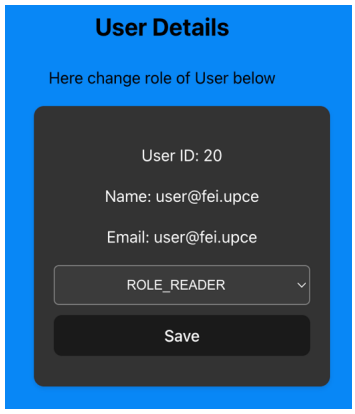
Záložka pro úpravu rolí účtu umožňuje změnu v roli uživatele. Pro vybraní uživatele stačí kliknout na jeho záznam v tabulce. Uživatelům s emaily admin@fei.upce a techUser@fei.upce nelze role měnit. Zůstane tím zachována správnost funkčnosti synchronizace a nedojde ke ztrátě přístupu k webovému rozhraní.



Obrázek 7: Webová stránka pro úpravu rolí uživatelů [Zdroj: vlastní tvorba]



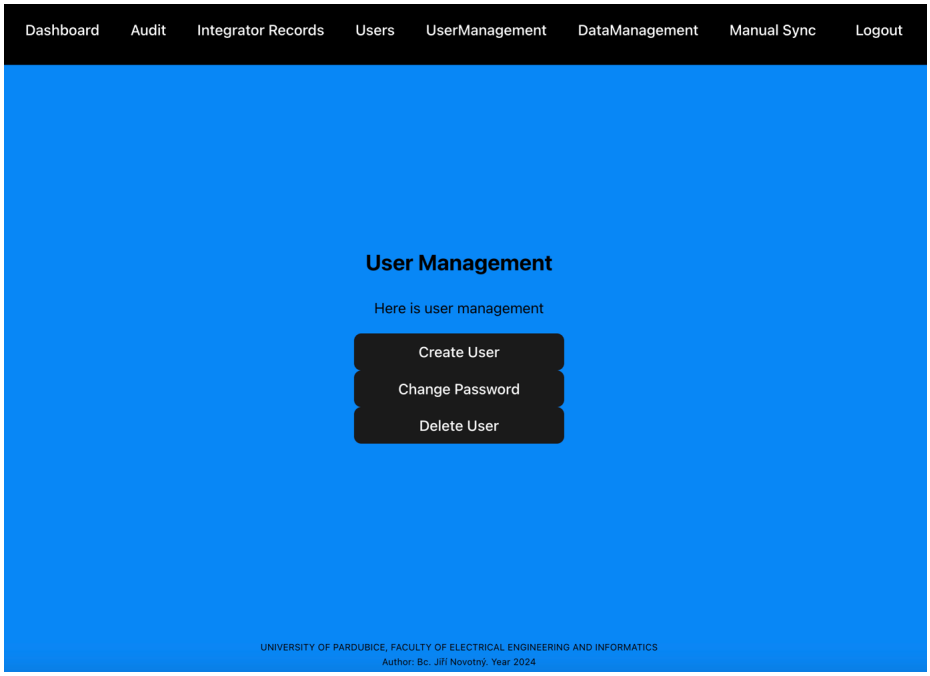
Obrázek 8: Výběr objektu administrátora [Zdroj: vlastní tvorba]



Obrázek 9: Výběr objektu uživatele [Zdroj: vlastní tvorba]

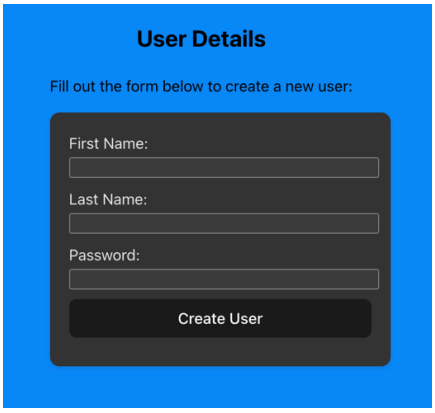
Správa uživatelů

Uživatel přihlášený administrátorským účtem může vytvářet a mazat účty a měnit hesla k účtům.



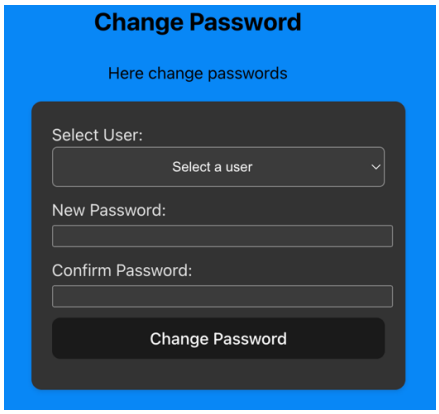
Obrázek 10: Webová stránka pro vytváření, mazání a změnu hesel k účtům [Zdroj: vlastní tvorba]

Po výběru vytvoření nového uživatelského účtu se otevře nová stránka. Nový účet je vytvořen s právem pouze pro čtení.



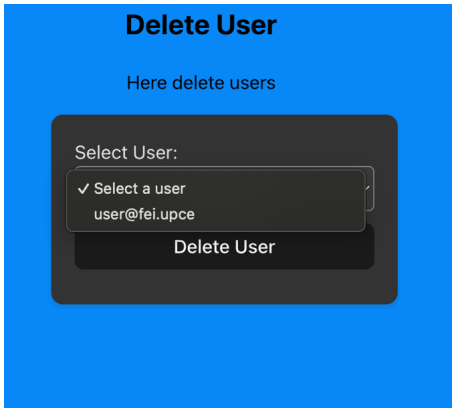
Obrázek 11: Vytvoření nového účtu [Zdroj: vlastní tvorba]

Po výběru změny hesla se otevře nová stránka.



Obrázek 12: Změna hesla k vybranému účtu [Zdroj: vlastní tvorba]

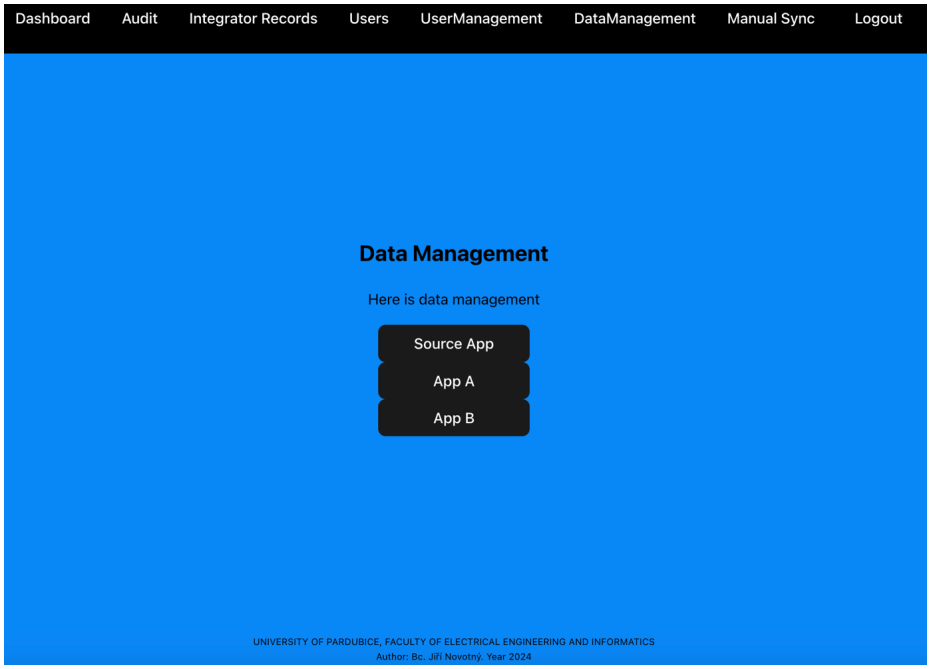
Po výběru vymazání účtu uživatele se otevře nová stránka. Z důvodu zachování správného fungovaní integrací a zachování přístupu k webovému rozhraní není možné vymazat účty technického uživatele a administrátora.



Obrázek 13: Výběr uživatelského účtu k vymazání [Zdroj: vlastní tvorba]

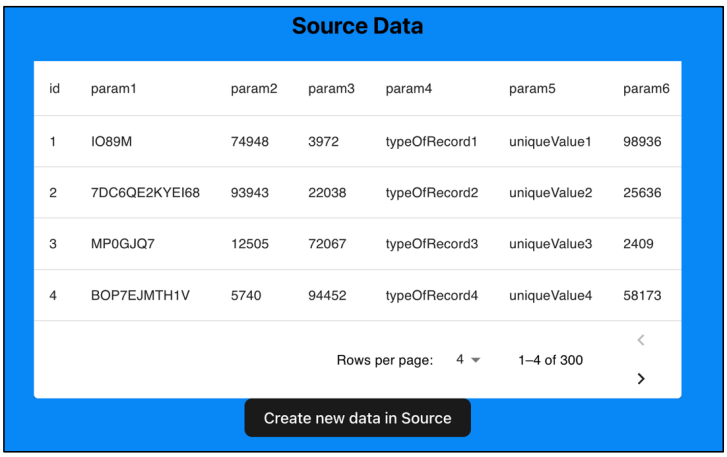
Správa a nahlížení na data v jednotlivých aplikacích

Tato záložka má pouze demonstrativní účel pro účely diplomové práce. Cílem diplomové práce je vytvoření integračního bodu. K tomu je potřeba mít v okolních aplikacích data. V reálném systému by tato záložka nebyla, aplikace by měly vlastní rozhraní.



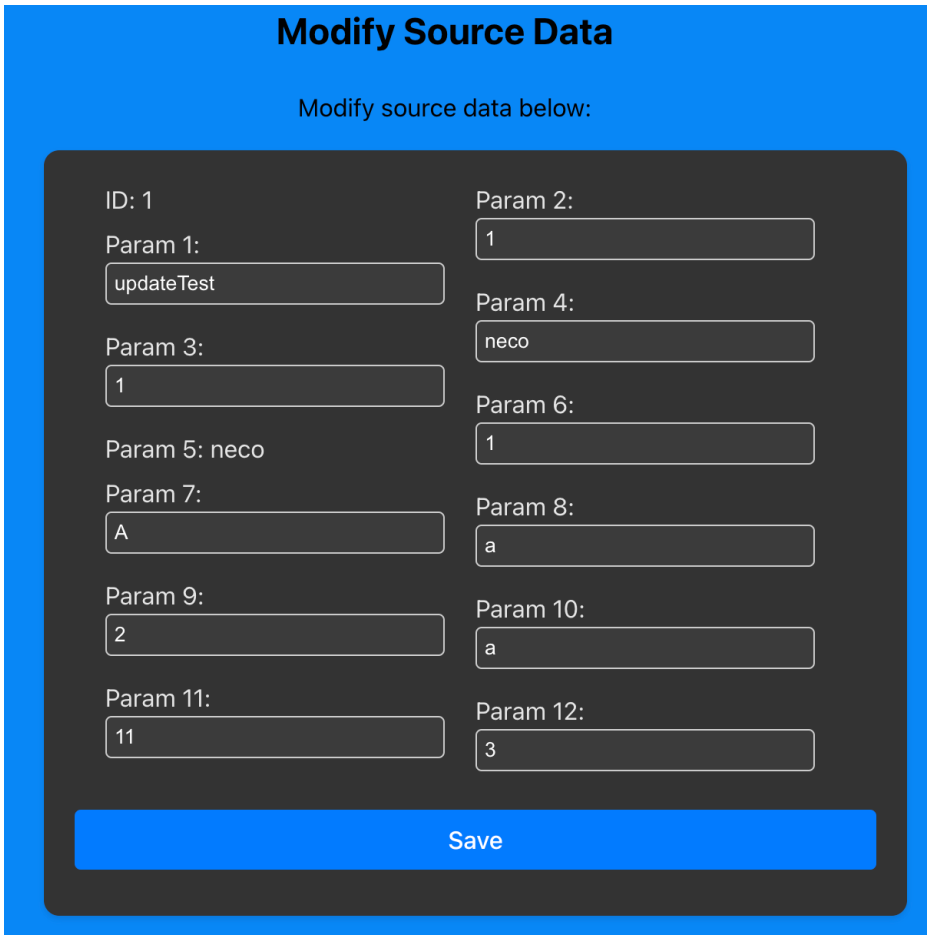
Obrázek 14: Webová stránka pro správu dat [Zdroj: vlastní tvorba]

Po prokliku na data ve zdrojové aplikaci se zobrazí tabulka s daty ve zdrojové aplikaci.



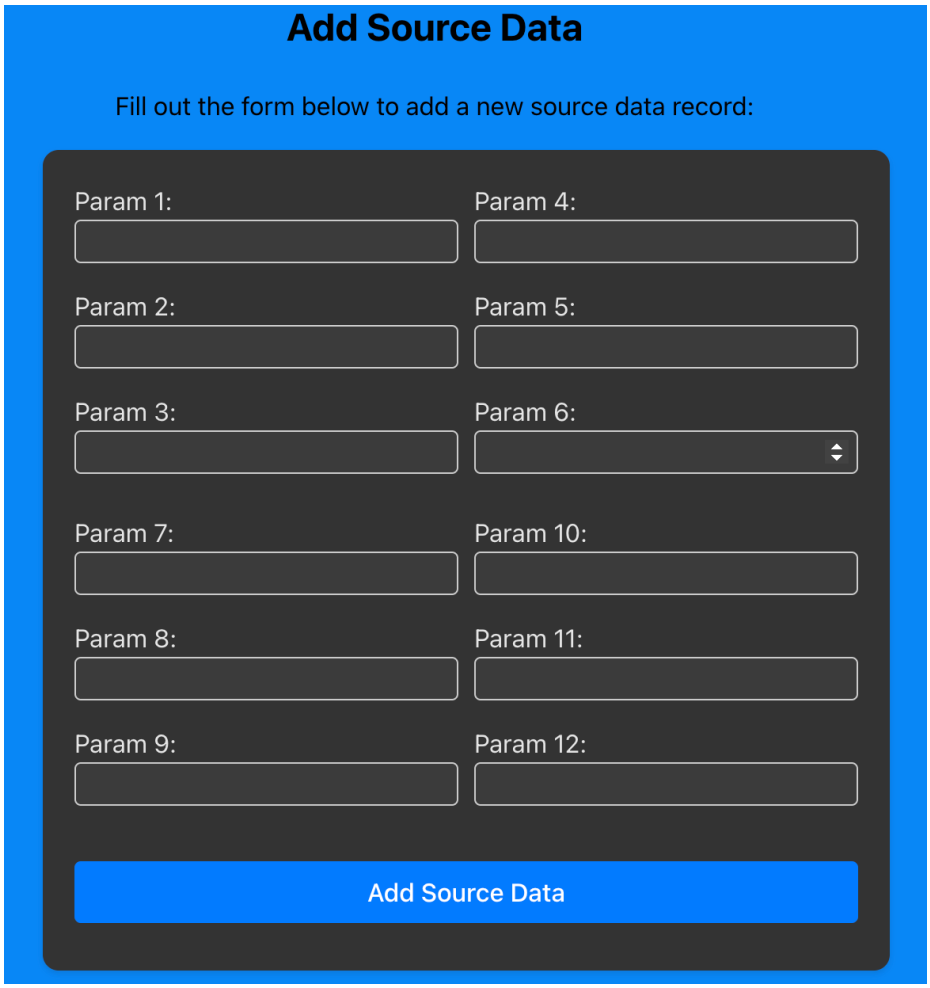
Obrázek 15: Tabulka s daty ve zdrojové aplikaci [Zdroj: vlastní tvorba]

Data je možné upravovat. Pro úpravu záznamu stačí v tabulce kliknout na vybraný záznam.



Obrázek 16: Úprava záznamu ve zdrojové aplikaci [Zdroj: vlastní tvorba]

Do zdrojové aplikace je možné vkládat záznamy.

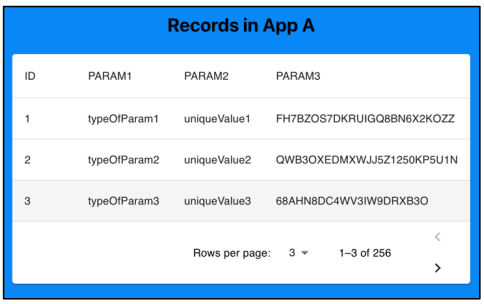


Obrázek 17: Vkládání dat do zdrojové aplikace [Zdroj: vlastní tvorba]

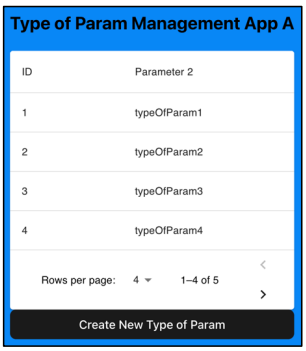
V cílových aplikacích je možné prohlížet typy záznamů a vkládat typy záznamů.

Obrázek 18: Rozcestník pro aplikaci A [Zdroj: vlastní tvorba]

Rozhraní pro cílové aplikace a operace jsou totožné.

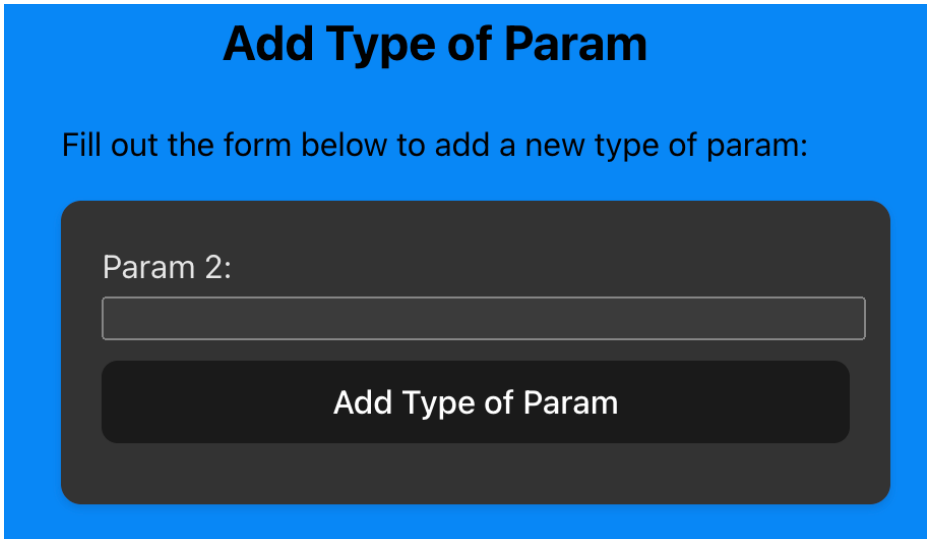


Obrázek 19: Prohlížení záznamů pro aplikaci A [Zdroj: vlastní tvorba]

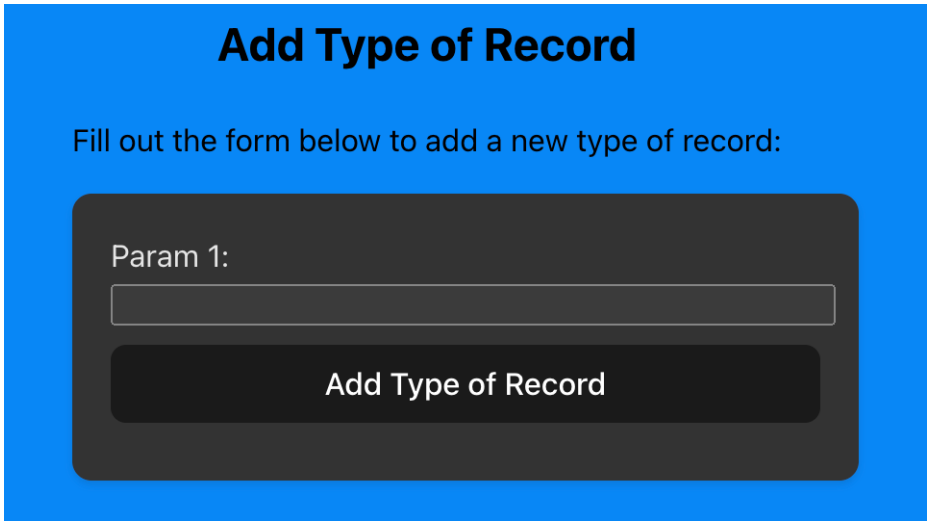


Obrázek 20: Prohlížení typů záznamů pro aplikaci A [Zdroj: vlastní tvorba]

Rozdíl v cílových aplikacích A a B je v pojmenování typů záznamů. V aplikaci A jsou typy záznamů pojmenovány „Type of Param“ a v aplikaci B jsou pojmenovány „Type of Record“.



Obrázek 21: Přidání typu záznamu pro aplikaci A [Zdroj: vlastní tvorba]



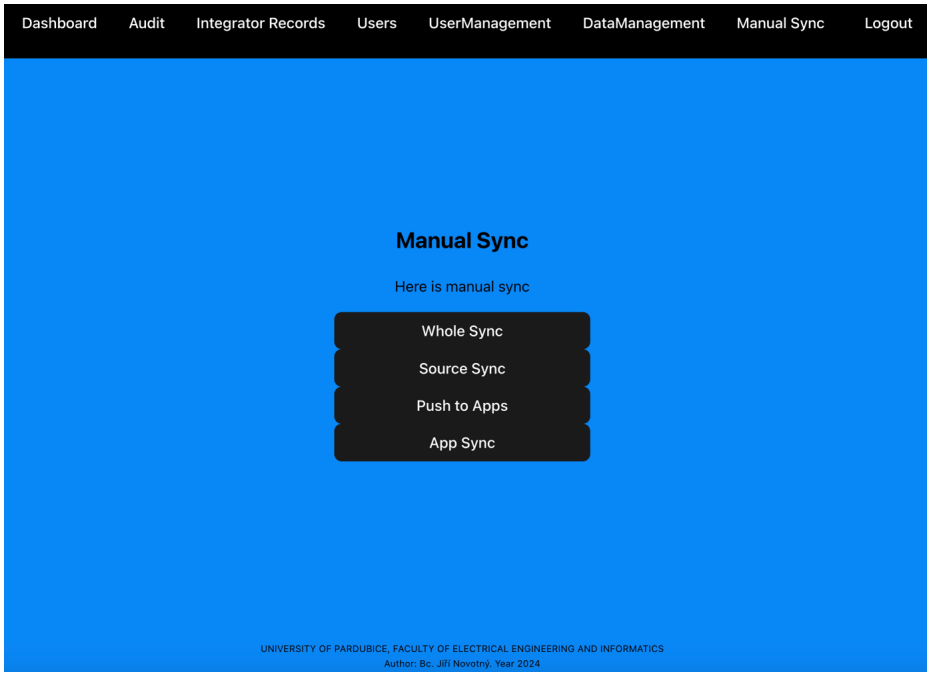
Obrázek 22: Přidání typu záznamu pro aplikaci B [Zdroj: vlastní tvorba]

Manuální synchronizace

Předposlední záložka ve webovém rozhraní slouží k vyvolání manuální synchronizace dat. Synchronizovat lze:

- Celou synchronizaci.
- Načtení dat ze zdrojové aplikace do integračního bodu.
- Nahrání dat do cílových aplikací.
- Načtení dodatečných dat z cílových systému do integračního bodu.

Po stisknutí jedné ze synchronizací se kurzor myši automaticky deaktivuje za účelem provedení synchronizace. Následně se sám zaktivuje.



Obrázek 23: Webová stránka pro synchronizaci [Zdroj: vlastní tvorba]

Odhlášení uživatele

JWT token se ukládá po přihlášení v lokálním úložišti. Je doporučeno při odchodu z webového rozhraní stisknout tlačítko pro odhlášení uživatele.

Obrázek 24: Tlačítko pro odhlášení uživatele [Zdroj: vlastní tvorba]