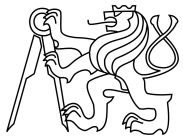
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Aplikace pro správu a integraci studentských uživatelských účtů v prostředí střední školy
Student:	Bc. David Janíček
Vedoucí:	Ing. Petr Špaček, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Pro potřeby střední školy navrhněte a implementujte systém pro správu studentských uživatelských účtů. Systém bude umožňovat integraci uživatelských účtů existujících školních systémů: AD account (NT domain), Moodle, YSoft a Bakaláři v rámci nově vytvořeného jediného školního účtu. Kromě centralizované správy účtů všech integrovaných systémů, navrhněte a implementujte také SSO autorizační službu (podobně jako SSO na FIT) pro ostatní systémy. Dále navrhněte a implementujte základní rozhraní pro napojení systému na Google Apps. Pro proces vytváření centrálního účtu implementujte možnost autorizace pomocí třetí strany (MojeID, OAuth2, Facebook).

Seznam odborné literatury

Dodá vedoucí práce.

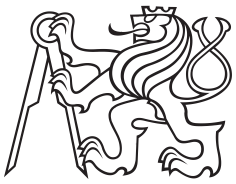
L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 24. listopadu 2015

Diplomová práce



České
vysoké
učení technické
v Praze

F8

Fakulta informačních technologií
Katedra softwarového inženýrství

Aplikace pro správu a integraci studentských uživatelských účtů v prostředí střední školy

Bc. David Janíček

Vedoucí práce: Ing. Petr Špaček, Ph.D.
Květen 2016

Odkaz na tuto práci

Janíček, David. *Aplikace pro správu a integraci studentských uživatelských účtů v prostředí střední školy.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Poděkování

Chtěl bych poděkovat zejména Ing. Petru Špačkovi, Ph.D., vedoucímu této diplomové práce, za jeho ochotu, čas a vždy rychlé a bezproblémové řešení všech otázek a událostí. Dále bych rád poděkoval firmě KOBOZ SERVICE s.r.o., především Ing. Janu Stejskalovi, za cenné rady a pomoc s definováním i realizací zadání a zprostředkováním spolupráce se Střední zdravotnickou školou a Vyšší odbornou školou zdravotnickou České Budějovice. Škole děkuji za poskytnutí prostředí a zároveň za procesní podněty zapracované do této práce. V neposlední řadě děkuji své rodině a přítelkyni za korekturu diplomové práce a velkou podporu nejen v průběhu jejího psaní, ale i během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 6. května 2016

. . . . . .

Abstrakt

Cílem diplomové práce je navrhnout a implementovat systém pro správu a integraci uživatelských účtů v prostředí střední školy. Nástroj bude integrovat definované systémy a kontrolovat životní cyklus jednotného uživatelského účtu. Uživatel se bude moci přihlašovat do integrovaných aplikací prostřednictvím jednotných uživatelských údajů a bude mu umožněno spravovat své údaje v integrační aplikaci. Nástroj bude primárně integrovat systémy Bakaláři, Active Directory a Google Apps. V produkčním prostředí bude navržena a nasazena jednotná ověřovací služba, tzv. Single Sign-On (SSO), sloužící jako poskytovatel identity pro integrované aplikace. Uživatelé se vůči ověřovací službě budou autentizovat také použitím jednotného účtu.

Klíčová slova: integrace, integrační vzory, integrační architektury, jednotný účet, Single Sign-On, SSO, SAML, OAuth, Active Directory, LDAP, ADFS, federace, Bakaláři, Moodle, Google Apps, Spring Framework

Abstract

The goal of this master thesis is to design and implement a system for management and integration of users' accounts in an environment of a high school. The tool will integrate defined systems and will control the lifecycle of a single user account. An user will be able to login to integrated applications using his single credentials and will be allowed to manage and edit some information about himself. The application will primarily integrate systems Bakalari, Active Directory and Google Apps. In the production environment there will be designed and deployed a single authentication service, so called Single Sign-On (SSO), serving as an identity provider for integrated applications. Users will authenticate against this service by providing their own credentials from single user account.

Keywords: integration, integration patterns, integration architectures, single account, SSO, SAML, OAuth, Active Directory, LDAP, ADFS, federation, Bakaláři, Moodle, Google Apps, Spring Framework

Obsah

Seznam obrázků

Seznam ukázek kódů

Seznam tabulek

Kapitola 1
Úvod

Diplomová práce má za cíl navrhnout a implementovat spolehlivý nástroj pro správu a integraci uživatelských účtů v prostředí střední školy. Většina středních škol se potýká s problémy snadné udržitelnosti, spravovatelnosti a nastavením různých aplikací a programů, zvlášť pokud se jedná o správu uživatelských účtů. Buď bývá proces vzniku, zániku a úpravy účtů prováděn ručně, nebo za přispění jednodušších skriptů či aplikací, které však mívají omezené možnosti, často slouží jen k jednomu účelu a tím pádem narůstá jejich počet a v důsledku časová a finanční ztráta při jejich údržbě administrátorem. Diplomová práce má za úkol tyto nedostatky odstranit, poskytnout ucelenou aplikaci, jež přebere většinu zodpovědnosti na sebe. Školy totiž nedisponují příliš velkými finančními možnostmi a nemohou si tak dovolit provozovat hotová, placená řešení.

Práce přímo navazuje na bakalářskou práci „Nástroj pro integraci uživatelských účtů“ (INUM) [17], s tím rozdílem, že nástroj vytvořený pro účely bakalářské práce byl nasazen do produkčního prostředí Vyšší odborné školy a Střední školy veterinární, zemědělské a zdravotnické Třebíč (SZŠ Třebíč) [1], zatímco v rámci diplomové práce jako produkční prostředí slouží Střední zdravotnická škola a Vyšší odborná škola zdravotnická České Budějovice (SZŠ ČB) [2]. Změna prostředí s sebou přináší nové aplikace a služby, které lze zahrnout do integrace. Zároveň je kladen větší důraz na obecnost a širší použitelnost nástroje tak, aby nebylo složité nasadit aplikaci i na jiné školy.

1.1 Cíl 1 - Jeden účet pro vše

Nástroj vytvořený v rámci bakalářské práce slouží primárně k integraci aplikací a služeb, přesněji jako backendová aplikace starající se o synchronizaci uživatelských účtů, jejími uživateli byli tak pouze administrátoři. V diplomové práci se aplikace přiblíží samotným uživatelům, o jejichž účty se stará. Vznikne jeden účet pro vše, pomocí kterého se uživatelé budou moci přihlašovat do integrovaných systémů, zároveň budou moci měnit ve svém účtu údaje, například hesla a emailové adresy. Změna se samozřejmě promítne do všech

[1] WWW stránky pro SZŠ Třebíč: `http://www.szstrebic.cz/`
[2] WWW stránky pro SZŠ ČB: `http://www.szscb.wz.cz/`

zapojených služeb.

Pro uživatele je velice pohodlné se do svého účtu přihlásit pomocí některého ze zaběhnutých, odzkoušených a hlavně používaných řešení třetích stran - účet na Google, Facebook a jiných třetích stran poskytujících ověření uživatele. V práci budou popsány a porovnány možnosti ověření třetích stran a některé budou implementovány jako alternativy pro přihlašování k uživatelskému účtu příslušné osoby namísto klasické kombinace přihlašovacího jména a hesla.

Dílčí část diplomové práce se bude věnovat popisu a implementaci napojení nástroje na Google Apps. Dle existujícího účtu se vytvoří a bude synchronizovat s Google Apps (GA) účtem, čímž uživatel získá přístup k velkému množství aplikací nabízených společností Google - email (Gmail), kalendář, diskové úložiště (Google Drive) a spousty dalších.

1.2 Cíl 2 - Jednotná ověřovací služba

Velký uživatelský komfort by přineslo, pokud by se uživatel nemusel do jednotlivých aplikací přihlašovat separátně, ale existovala jednotná ověřovací služba, vůči které se uživatel jednou ověří a ona poskytne dalším aplikacím informace o tomto přihlášení. Aplikace pak již uživatele považují za přihlášeného a nevyžadují opětovné zadávání přihlašovacích údajů.

Přesně tyto výhody přináší princip nazvaný Single Sign-On (SSO). V textu bude rozebráno jeho fungovaní, porovnání nabízených metod a jejich popisu a nakonec postup nasazení vybraného systému do produkčního prostředí.

1.3 Členění

Kapitola *2 Analýza integrovaných systémů a jejich rozhraní* se bude zabývat dostupnými rozhraními integrovaných systémů, kapitola *3 Analýza systémové integrace* metodami integrace a kapitola *4 Analýza mechanismů ověřování* dostupnými řešeními v oblasti autentizace a autorizace.

Navazující kapitola *5 Návrh* má za úkol navrhnout proces vzniku a zániku uživatelských kont, definovat datové přenosy mezi aplikacemi, rozhodnout, které entity se budou synchronizovat s Google Apps a zvolit řešení pro implementaci SSO a autentizaci pomocí účtů třetích stran.

V kapitole *6 Implementace* budou vybrány zajímavé a důležité části implementačního kódu a budou detailně popsány. Závěrečná kapitola *7 Testování* ověří správnost návrhu a popíše postup testování aplikace.

Kapitola 2

Analýza integrovaných systémů a jejich rozhraní

Tato kapitola popisuje systémy, které se budou integrovat v rámci vytvoření jednotného společného účtu a zároveň rozebírá jejich dostupná rozhraní. Značná část kapitoly se věnuje popisu protokolu LDAP a fungování API Google Apps. Na konci kapitoly se krátká sekce zmiňuje o konkurenční aplikaci.

2.1 Rozhraní integrovaných systémů

Sekce se zaměřuje na přístupná rozhraní a možnosti integrace systémů používaných v prostředí SZŠ ČB. Detailněji budou popsány hlavně aplikace a služby Bakaláři, Active Directory (AD), Moodle a Google Apps (GA).

2.1.1 Bakaláři

Bakaláři [1] jsou elektronický školní informační systém, který spravuje data žáků, studentů, učitelů a dalších zaměstnanců školy. Obsahuje agendy pro evidenci osobních dat (jména, adresy, emailové adresy, telefonní čísla aj.) - tzv. Školní matrika, evidenci prospěchu a docházky žáků a studentů, systém pro tvorbu rozvrhu a suplování, různá napojení na matriky a systémy Ministerstva školství, modul vysvědčení. V neposlední řadě disponují i webovou aplikací s přístupem pro žáky (studenty) a rodiče, kde je vedena jejich klasifikace a další údaje - tzv. Internetová žákovská knížka.

V textu *Školní agenda* se uvádí: „V roce 2014 nejvíce škol v České republice používá systém Bakaláři (licenci vlastní více než 3500 škol). Některé školy, zpravidla ty menší, třeba jen pro evidenci žáků, přes 2000 škol ale také pro tvorbu rozvrhu, téměř 1400 škol využívá internetovou žákovskou knížku a takřka 1200 škol vlastní licenci pro elektronickou třídní knihu.“ [18]

Bakaláři díky své rozšířenosti, jak je uvedeno v předchozím odstavci, představují nejpoužívanější systém evidence žáků pro školy v České republice. Další, méně časté, systémy jsou například Systém agend pro školy (SAS) [2],

[1] WWW stránky aplikace Bakaláři: `http://www.bakalari.cz/homepage/index.htm`

[2] WWW stránky aplikace Systém agend pro školy (SAS): `http://sas.mp-soft.cz/`

dm Evidence [3] a modernější Škola OnLine [4].

Protože Bakaláři obsahují komplexní agendy a funkce, jedná se většinou o primární zdroj dat pro celou školu. Stejně tak je tomu na škole SZŠ ČB. Tohoto faktu se drží návrh i implementace integračního nástroje INUM, který většinu dostupných dat získává právě z Bakalářů. Takové použití eliminuje nutnost vícenásobného zadávání dat, převážně o žácích a studentech. Kvůli přijímacím zkouškám a řízením jsou data o žácích a studentech v Bakalářích ještě dříve, než se osoba skutečně stane žákem/studentem příslušné školy. Když pak nastane okamžik, kdy osoba začne být pro integrační nástroj aktivní a tedy její konto se zúčastní synchronizace, stačí jen doplnit či vygenerovat chybějící údaje a není nutné osobu do aplikace například vkládat ručně. Detailně se procesu vzniku a zániku uživatelských účtů bude věnovat kapitola *5 Návrh*.

Na SZŠ ČB jsou instalovány dvě instance softwaru Bakaláři, jeden pro Střední zdravotní školu a druhý pro Vyšší odbornou školu zdravotnickou. Na střední škole je verze 15/16 na relačním databázovém systému Microsoft SQL Server 2008 (MSSQL), na vyšší odborné škole je verze 10/11 běžící na souborovém databázovém systému Visual FoxPro, s tabulkami ve formátu dBase database file (DBF).

Ani jedna z verzí nenabízí API (Application Programming Interface) pro automatický import případně export dat z Bakalářů. Lze ručně importovat v desktopové aplikaci například údaje o žácích ve formátu CSV (Comma Separated Values), pouze však po přihlášení do aplikace a proklikání se v menu programu.

V případě Bakalářů bude muset integrační aplikace přistupovat přímo do databáze, tzn. nutnost poznat její strukturu a vazby mezi entitami.

2.1.2 Active Directory

„Active Directory (AD) je síťový operační systém od firmy Microsoft. AD umožňuje administrátorům efektivně spravovat firemní informace v centrálním úložišti (repositáři), který může být distribuovaný globálně. Jakmile byly informace o uživatelích, skupinách, počítačích, tiskárnách, aplikacích a službách přidány do Active Directory, mohou být zpřístupněny k použití v celé organizaci, libovolnému množství uživatelů. Struktura informací může odpovídat struktuře organizace a uživatelé mohou dotazovat AD pro nalezení například tiskárny nebo emailové adresy kolegy. Organizačními jednotkami lze libovolným způsobem delegovat kontrolu a správu dat.“ [9]

Active Directory představuje velmi komplexní systém pro správu dat organizace, a tak se přímo nabízí možnost, kdy se velká část úkonů přesune na tento systém - konkrétně třeba synchronizace s Moodle, který má modul pro AD (viz podsekce *2.1.3 Moodle*), napojení Ysoft SafeQ (viz podsekce *2.1.5 YSoft SafeQ*) nebo jeho role Active Directory Federation Services (ADFS),

[3] WWW stránky aplikace dm Evidence: `http://www.dmsoftware.cz/zs/index.html`
[4] WWW stránky aplikace Škola OnLine: `https://www.skolaonline.cz/`

tedy řešení přístupu identit a implementací mimo jiné i SSO (viz kapitola *4 Analýza mechanismů ověřování*).

Active Directory implementuje protokol Lightweight Directory Access Protocol (LDAP) verze 3 popsaném v RFC 3377 (Request for Comments). [3] [31] Vzhledem k tomu, že aplikace INUM je napsána v programovacím jazyku Java, konkrétně ve frameworku Spring, poskytuje Spring knihovnu pro komunikaci s LDAP - Spring LDAP [5]. Protokol LDAP je podrobně popsán v sekci *2.2 LDAP - Lightweight Directory Access Protocol*.

Další z mnoha dostupných možností pro komunikaci s Active Directory je Windows PowerShell. Windows PowerShell představuje shell s příkazovou řádkou a skriptovacím jazykem navržený převážně pro systémové administrátory. Od verze Microsoft Windows Server 2008 R2 obsahuje modul pro Active Directory, který shlukuje tzv. cmdlety [6] pro správu AD. [23] Některé administrátorské příkazy nelze vykonat pomocí LDAP, proto je zapotřebí použití Windows Powershell, konkrétní příklady jsou uvedeny v kapitole *5 Návrh*.

Na škole SZŠ ČB mají nainstalovaný doménový kontroler se službou Active Directory na Windows Server 2008 R2, Service Pack 1.

2.1.3 Moodle

Moodle je online výuková platforma poskytující vyučujícím, administrátorům a studentům jeden robustní, bezpečný a integrovaný systém pro vytváření vlastních prostředí pro učení. Moodle je open-source projekt, vytvářený komunitou v rámci Moodle projektu, která je vedena a koordinována australskou společností Moodle HQ. [24]

Integraci e-learningové aplikace Moodle se detailně věnuje bakalářská práce „Nástroj pro integraci uživatelských účtů“. Integrace spočívá v napojení Moodle na Active Directory pomocí PHP pluginu. Moodle je napsaný v programovacím jazyku PHP, takže vyžaduje na operačním systému nainstalovaný PHP interpretr. Díky této skutečnosti lze v systémovém plánovači pouštět libovolné PHP skripty právě pomocí PHP interpretru, čehož se v integraci využívá. Pravidelně se spouštějí skripty pro synchronizaci uživatelů (v případě SZŠ ČB učitelů, studentů a žáků) a jejich kohort - kohorta (angl. *cohort*) je speciální název pro skupinu uživatelů v *Moodle*, většinou se jedná o třídu žáků. Hesla uživatelských účtů se nepřenášejí, ověření uživatele probíhá vůči *Active Directory*.

Výše zmíněné synchronizační PHP skripty vycházejí z projektu mc-moodle-cohort-sync hostovaném na GitHub [7], s provedenými drobnými zásahy do PHP kódu. Projekt má i novější verzi moodle_local_ldap naprogramovanou přímo jako plugin do Moodle, který se řídí vlastním plánovačem aplikace, již ne systémovým [8].

[5] WWW stránky knihovny Spring LDAP: `http://projects.spring.io/spring-ldap/`
[6] cmdlet - příkaz v prostředí Windows Powershell
[7] cohort-sync: `https://github.com/manhattancollege/mc-moodle-cohort-sync`
[8] moodle_local_ldap: `https://github.com/patrickpollet/moodle_local_ldap`

V současné situaci je tento způsob integrace dostačující, respektive přímá komunikace mezi INUM a Moodle přes jiné rozhraní by kromě větší kontroly nad daty nepřinesla nic nového. Nicméně Moodle poskytuje REST API pro komunikaci s dalšími systémy, které například implementuje Java knihovna MoodleRest Java Library [9]. Knihovna se však nyní dále nerozvíjí, poslední úprava proběhla před více než 10 měsíci (ke dni 24. března 2016) - přesunutí synchronizace na aplikaci INUM by tak byl riskantní krok s nejasným výsledkem.

V produkčním prostředí je nasazena verze Moodle 2.2.1+ (Build: 20120213).

2.1.4 Google Apps

Integrace Google Apps je jednou z velkých částí této diplomové práce. Google nabízí své webové aplikace jako Gmail (emailový nástroj), Drive (úložný prostor), Kalendář, Kontakty, Skupiny, Dokumenty a spoustu dalších v balíku služeb nazvaném GA for Education (Google Apps pro vzdělávání). Od osobního účtu na Google, který je také zdarma, se liší v několika bodech [13]:

- **profesionální email přímo na doméně školy (@szscb.cz místo @gmail.com)**
- sdílený přístup do Drive, Kalendáře, Kontaktů, Dokumentů a dalších
- přidaný úložný prostor pro Gmail a Drive
- bez reklam
- podpora na telefonu i emailu 24/7
- garantovaná dostupnost služeb 99,9%
- **rozšířené možnosti zabezpečení (například napojení na autentizační službu organizace, důležité pro Cíl 2 diplomové práce)**
- administrace všech uživatelských účtů
- aplikace Classroom (e-learningový nástroj), dostupná jen ve verzi Google Apps for Education (prozatím nevyužívána kvůli Moodle)

Nejdůležitější službou z nabízených v Google Apps pro vzdělávání je email přes aplikaci Gmail. Služba je zdarma, nabízí neomezený prostor, vše je uloženo v cloudu, takže odpadá nutnost administrace vlastního emailového serveru, je dostupná i mimo doménu organizace a další řada výhod.

Škola SZŠ ČB se zatím rozhodla, že Gmail bude nabízet jen svým zaměstnancům, tedy převážně učitelům, v aplikaci bude ale myšleno na možnost rozšíření synchronizace i pro účty žáků a studentů (bude se jednat pouze o konkrétní konfiguraci).

[9] MoodleRest Java Library: `https://sourceforge.net/projects/moodlerestjava/`

Google Apps již na škole SZŠ ČB nasazoval a integroval v rámci své bakalářské práce „Integrace Google Apps“ Luboš Palíšek. [25] Vývoj ale probíhal v roce 2013 a dále v něm nebylo pokračováno. Společnost Google mezitím označila většinu z používaných API za zastaralá, některá dokonce přestala být funkční. Jedná se zejména o Provisioning API a Profiles API (oficiálně ukončena 20. dubna 2015 [12]), která byla zcela nahrazena Directory API [10]. Kompletní změny v rozhraních lze vidět v tabulce 2.1. Podrobný popis jak migrovat jednotlivé části API je dostupný na Migrating to the Directory API [11]. Podrobný popis služeb nabízených pomocí Directory API je uveden v sekci *2.3 API pro Google Apps*.

Zrušené rozhraní API	**Náhradní rozhraní API**
Admin Audit (audit administrátorských akcí)	Admin SDK Reports API
Documents List API	Drive API
Google Apps Profiles (profily)	**Admin SDK Directory API**
Provisioning (správa účtů)	**Admin SDK Directory API**
Reporting (přehledy)	Admin SDK Reports API
Email Migration API v1	Gmail API
Reporting Visualization (vizualizace přehledů)	Náhradní řešení není k dispozici

Tabulka 2.1: Tabulka pro migraci Google Apps API. [12]

Práce Luboše Palíška má kromě API další signifikantní rozdíly, největší z nich pak způsob, odkud získává zdrojová data. Jeho aplikace počítá jen s Bakaláři jako jediným zdrojem dat, nemá svá vlastní data uložená v databázi. V této diplomové práci se za primární zdroj dat považuje samotný nástroj pro integraci, s vlastní MySQL databází a vlastní strukturou dat. Ovšem i tak zůstávají data, která není nutné duplikovat (například rozvrh hodin), INUM si tak pro ně bude sahat přímo do databáze aplikace Bakaláři.

Škola SZŠ ČB využívá verzi Google Apps pro vzdělávání (Google Apps for Education), která je zcela zdarma, nabízí účet až pro 10 000 uživatelů v doméně (lze požádat o více) a neomezený úložný prostor pro všechny. [13]

2.1.5 YSoft SafeQ

YSoft SafeQ je řešení pro správu tisku, kopírování a skenovaní. Integrace YSoft SafeQ byla vysvětlena a implementována v rámci řešení bakalářské práce INUM [17] na SZŠ Třebíč, na škole SZŠ ČB však není tato aplikace využívána, proto se nebude diplomová práce této integraci věnovat.

[10] WWW stránky: `https://developers.google.com/admin-sdk/directory/`
[11] URL: `https://developers.google.com/admin-sdk/directory/v1/guides/migrate`

Stručně vysvětleno, YSoft SafeQ [12] lze napojit na doménu Active Directory pomocí vnitřního modulu vazbou přes unikátní klíč `objectGUID`. Podmínkou pro tuto integraci je, aby v AD existoval parametr objektu osoby (uživatele) držící hodnotu čísla jeho čipové karty. Synchronizace je iniciována ze strany softwaru YSoft SafeQ, který tak řídí datový tok, včetně vytváření a zneplatnění uživatelských účtů. Konkrétní informace nalezne čtenář v příslušné kapitole bakalářské práce [17].

2.2 LDAP - Lightweight Directory Access Protocol

Velká část integračního nástroje se týká komunikace s Active Directory skrze protokol LDAP za pomoci Java knihovny Spring LDAP. Z toho důvodu je důležité podrobněji popsat způsob fungování protokolu LDAP.

LDAP protokol definuje komunikaci mezi LDAP klientem a LDAP serverem. Vyměňované zprávy specifikují požadavky vyžádané od klienta (např. operace *search* - hledat, *modify* - změnit, *delete* - smazat, *bind* - připojit, aj.), odpovědi od serveru a formát dat zpráv. LDAP zprávy jsou přenášené pomocí protokolu TCP/IP. Návrhář LDAP adresáře (angl. *directory*) se ale nestará tolik o formát zpráv definovaný protokolem, ale o samotný logický model definovaný těmito zprávami, datové typy, jak je adresář organizovaný, které operace jsou povoleny, jak jsou informace zabezpečeny apod. [31]

Typická komunikace mezi LDAP klientem a serverem vypadá následovně:

1. Klient vytvoří spojení s LDAP serverem (*session*). Tento krok se nazývá *binding* se serverem. Klient specifikuje IP adresu nebo hostname serveru a TCP/IP port, na kterém server naslouchá.
2. Klient může buď poskytnout serveru přihlašovací jméno a heslo, aby se řádně autentizoval, nebo vytvoří tzv. anonymní sezení (*session*) se základními přístupovými právy. Klient může i navázat spojení, které používá bezpečnostní prvky, jako například šifrování dat.
3. Klient následně provádí operace v LDAP adresáři, který poskytuje operace pro čtení i zápis. To znamená, že klient může jak dotazovat již existující data, tak je spravovat nebo vkládat nová. LDAP podporuje prohledávání adresáře na základě kritérií zadaných klientem. Vyhledávání je běžná a častá operace, klient je schopen definovat, kterou část adresáře prohledávat a které informace žádá.
4. Když klient dokončí své požadavky, zavře *session* (spojení) se serverem - této operaci se říká *unbinding*.

Všechny moderní adresářové servery (Active Directory nevyjímaje) používají verzi LDAP verze 3. LDAP v3 je definován několika klíčovými dokumenty RFC, které určují jeho architekturu. Specifikace je postavena na 4 modelech:

[12] WWW stránky aplikace YSoft SafeQ: `https://www.ysoft.com/cs/safeq/home`

- **informační model** (angl. information model) - popisuje strukturu informací ukládaných v adresáři LDAP
- **jmenný model** (angl. naming model) - popisuje jak jsou informace organizovány a ukládány v LDAP adresáři
- **funkční model** (angl. function model) - popisuje operace, které mohou být provedeny nad informacemi v LDAP adresáři
- **bezpečnostní model** (angl. security model) - popisuje jak mohou být informace chráněny před neautorizovaným přístupem

2.2.1 Informační model

Základní jednotka informace uložené v adresáři je záznam. Záznamy reprezentují objekty z reálného světa jako lidi, servery, organizace apod. Záznamy jsou složeny z kolekce atributů, které obsahují informace o objektu. Každý atribut má typ a jednu nebo více hodnot. Typ atributu je asociován se syntaxí. Syntaxe specifikuje, jaký typ hodnot může být uložen. Například, záznam může mít atribut, s atributem spojená syntaxe upřesňuje, že jeho hodnoty představují telefonní čísla jako tisknutelné řetězce znaků volitelně následované klíčovými slovy popisujícími velikost papíru a rozlišení. Je možné, že záznam v adresáři organizace bude obsahovat více hodnot v tomto atributu, to znamená, že organizace nebo osoba reprezentovaná tímto záznamem bude mít více telefonních a faxových čísel. [31]

K definici toho, co lze v adresáři ukládat jako hodnotu atributu, syntaxe atributu také udává, jak se tyto hodnoty chovají během vyhledávání a jiných operací. Třeba atribut telefonní číslo může specifikovat:

- lexikografické řazení
- mezery, pomlčky a znak + jsou ignorovány během vyhledávání a porovnávání
- hodnoty mohou být jen řetězce znaků (ne například binární data)

Takže například hodnoty telefonního čísla +420 777 888 999, 420777888999 a 00420 777 888 999 jsou považovány za stejné.

S typy atributů mohou být spojeny i omezení, konkrétně třeba počet hodnot atributu (maximálně jedna hodnota, právě jedna hodnota, maximálně 10 hodnot, apod.) nebo jeho celková velikost (maximálně 255 znaků, maximálně 10 KB aj.).

V tabulce 2.2 jsou uvedeny některé typické atributy používané v LDAP.

Schémata definují typy objektů uložených v adresáři. Schémata také udávají seznam atributů pro každý objekt a zda jsou atributy povinné či volitelné. Například ve schématu *person* (osoba), atribut příjmení (*sn - surname*) je povinný, zatímco atribut popis (*description*) volitelný. Kontrola schémat zajišťuje, že všechny povinné atributy jsou přítomny předtím, než se záznam uloží do adresáře. Volitelné atributy mohou být doplněny kdykoliv později.

Název	Popis
sAMAccountName	přihlašovací jméno uživatele
userPrincipalName	UPN, přihlašovací jméno účtu ve tvaru <uživatel>@<DNS-název-domény>
displayName	celé jméno používané typicky aplikacemi
givenName	křestní jméno
sn	příjmení
description	popis
mail	emailová adresa
memberOf	seznam skupin, ve kterých je objekt členem

Tabulka 2.2: Typické atributy v LDAP.

Schéma také definuje dědičnost a podtřídy objektů a kde ve struktuře DIT (hierarchii) se mohou objekty objevit.

Objekty mohou být odvozeny od jiných objektů - tomu se říká v LDAP terminologii *subclassing* (dědičnost, podtřídy). Například objekt *person* obsahuje atributy jako příjmení, jméno a tak dále. *Object class* (objektová třída) *organizationalPerson* může být podtřídou *object class person*, potom bude mít stejné atributy jako objektová třída *person* a sama může definovat další atributy jako pozice v organizaci, číslo dveří kanceláře, telefon v rámci organizace apod. Třída objektu *person* pak bude nadřazená objektové třídě *organizationalPerson*. Existuje jedna speciální objektová třída nazvaná *top*, která nemá žádné předky a obsahuje povinný atribut *objectClass* - atributy z *top* se pak objevují ve všech záznamech v adresáři (buď jako povinné nebo volitelné). Tabulka 2.3 uvádí některé *objectClasses*, které jsou použity v integrační části Active Directory aplikace INUM.

Object class	**Atributy**
top	*objectClass*
person	*cn (common name)*, *sn (surname)*, telephoneNumber, description, ...
organizationalPerson	title, telephoneNumber, street, postalCode, postalAddress, ou, ...
user	userPrincipalName, unicodePwd, ...
group	*cn*, *member*, description, ...

Tabulka 2.3: Objektové třídy a jejich atributy pro Active Directory.

Všechny záznamy v adresáři mají speciální atribut *objectClass* (buď ho dědí

z objektové třídy *top* nebo ho sami definují). Hodnota atributu *objectClass* je seznam 2 nebo více názvů schémat. Tato schémata definují jaký typ objektů záznam reprezentuje. Jedna z hodnot atributu musí být buď *top* či *alias*. *Alias* je používán pokud je záznam alias pro jiný záznam, jinak je použit *top*. *ObjectClass* atribut rozhoduje, které atributy musí a může záznam mít. [31]

Podle toho, kde se nachází záznam ve stromové struktuře, se buď jedná o *list* (nemá žádné potomky, *leaf entry*) nebo o *kontejner* (*container object*). *Kontejner* může obsahovat jeden nebo více objektů.

2.2.2 Jmenný model

Jmenný model LDAP definuje jak jsou záznamy identifikovány a organizovány. Záznamy jsou organizovány ve struktuře podobné stromu zvané Directory Information Tree (DIT). Záznamy jsou rozmístěny v DIT podle jejich Distinguished Name (DN). DN je unikátní název, který jednoznačně identifikuje jednotlivý objekt. DN se skládá z posloupnosti několika *relative distinguished names* (relativní DN - RDN). [31]

Každé RDN v DN odpovídá větvi v DIT vedené od kořene DIT ke konkrétnímu záznamu. Běžně má RDN formu `<název atributu> = <hodnota>`. DN je složena z posloupnosti několika RDN oddělených čárkami. Názvy atributů v RDN pro LDAP a AD jsou vypsány v tabulce 2.4.

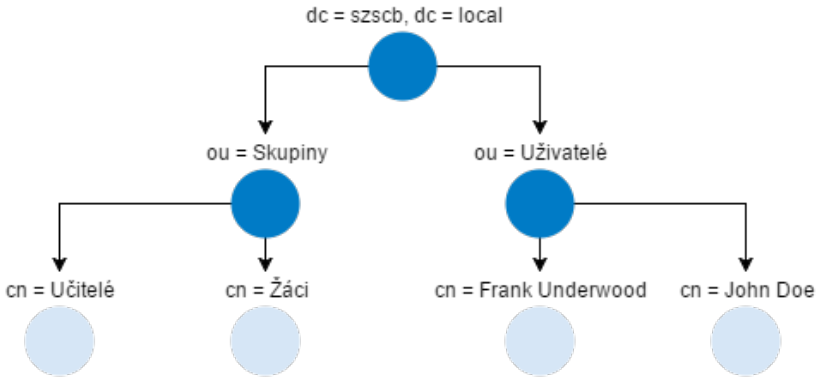
Zkratka	Název	Popis
CN	Common Name	název záznamu (musí být unikátní v rámci jedné OU)
OU	Organization Unit	organizační jednotka
O	Organization	organizace (nepoužíváno v AD)
DC	Domain Component	komponenta domény (používáno v AD)
C	Country	stát (nepoužíváno v AD)

Tabulka 2.4: Názvy atributů RDN.

Na obrázku 2.1 je vidět příklad stromové struktury DIT. Například osoba Frank Underwood by v této struktuře měla DN `cn = Frank Underwood, ou = Uživatelé, dc = szscb, dc = local`, zatímco jeho RDN je `cn = Frank Underwood`.

V LDAP je běžné identifikovat objekt pomocí jeho DN, v Active Directory existuje však více metod, jak jednoznačně identifikovat záznam (v terminologii AD se záznamu říká objekt).

Lze použít OID (*Object Identifier*), což je hierarchický, unikátní identifikátor, složený z dekadických číslic oddělených tečkou. V Active Directory má navíc objekt přiřazen další jednoznačný unikátní indetifikátor, který se označuje jako ***objectGUID*** (*Object Globally Unique Identifier*). Jedná se



Obrázek 2.1: Příklad stromové struktury s několika objekty.

o 128-bitové číslo, které jakmile je jednou vytvořené, již se nemění, ani při přesunu objektu v rámci stromové architektury. [3]. Tento atribut je velmi důležitý pro integraci Active Directory, jak bude vysvětleno v kapitole *5 Návrh*.

2.2.3 Funkční model

Funkční model LDAP se skládá ze tří kategorií operací, které lze provádět vůči adresáři splňujícímu specifikace LDAPv3:

- **Autentizace** (angl. *authentication*) - Operace *bind*, *unbind* a *abandon* používané k připojení a odpojení se od LDAP serveru, získání přístupových práv a ochraně informací.
- **Dotazování** (angl. *query*) - Operace hledání a porovnávání záznamů podle uživatelsky specifikovaných kritérií.
- **Aktualizace** (angl. *update*) - Přidání, smazání a modifikování záznamu, případně změna DN nebo RDN záznamu.

Konkrétní názvy operací a jejich popis lze vidět v tabulce 2.5.

2.2.4 Bezpečnostní model

Bezpečnostní model je založen na *bind* operaci. Existuje několik různých typů *bind* operací a s tím spojených různých bezpečnostních mechanismů. Jedna z možností je, když klient požadující přístup poskytne DN identifikující sebe sama spolu s heslem v textové podobě. Pokud není poskytnuto žádné DN ani heslo, LDAP server považuje sezení (*session*) za anonymní. Použití hesla v textové formě není doporučováno pokud transportní služba nemůže garantovat důvěrnost a mohlo by tak nastat odhalení hesla neautorizovanou stranou.

LDAPv3 poskytuje *bind* operaci podporující mechanismus *Simple Authetication and Security Layer* (SASL). Jedná se o obecný autentizační princip,

Kategorie	Operace
Autentizace	*bind* - inicializuje a autentizuje spojení
	unbind - ukončí session (spojení)
	abandon - klient chce ukončit posílání výsledků na jeho poslední dotaz
Dotazování	*search* - výběr dat pomocí filtru
	compare - porovná atribut se zadanou hodnotou
Aktualizace	*add* - vytvoří nový záznam
	modify - změní již existující záznam
	modifyRDN - přesune záznam (objekt) v DIT
	delete - smaže záznam

Tabulka 2.5: Operace poskytované LDAP. [3]

kde několik různých autentizačních metod dovoluje ověřit klienta vůči serveru - jedním z nich například *Kerberos*. Navíc jsou v LDAPv3 dostupné rozšířené operace protokolů. Takovým rozšířením v rámci bezpečnosti je třeba *Extension for Transport Layer Security* (TLS), který umožňuje operacím využít TLS k šifrování spojení k LDAP a ochraně proti podvodům (angl. *spoofing*). TLS obsahuje mechanismus ke komunikaci s SSL serverem, včetně zpětné kompatibility. [31]

2.3 API pro Google Apps

Největší výhodou služeb Google Apps je jejich dostupnost přes web a samozřejmě také fakt, že jsou zdarma, alespoň ve verzi Google Apps pro vzdělávání (*Google Apps for Education*). Umístění aplikací na web přináší řadu výhod:

- **Přenositelnost**. Pokud je program nainstalovaný na konkrétním počítači, je vázán přímo na jeho operační systém. Pokud tedy uživatel pracuje odpoledne na stolním počítači a večer chce pracovat na svém domácím notebooku, musí myslet na přenos souborů a zálohování, jinak není schopen navázat na předchozí práci. S Google Apps nemusí nic instalovat na počítač, ale přistupuje k aplikacím a svým datům odkudkoliv pomocí internetu.[6]
- **Mobilita**. Webové aplikace fungují také na mobilních zařízeních, takže uživatel může přistupovat z iPhonu, Blackberry, tabletu a dalších mobilních zařízeních.[6]
- **Spolupráce**. Pouze konkrétní uživatel má přístup k dokumentům uloženým na disku počítače. Pokud chce dokumenty sdílet, musí je poslat

emailem či jinou službou nebo je vytisknout. Když je dokument na webu, může ho sdílet s mnoha dalšími lidmi najednou. Pomocí Google dokumentů (Google Docs) může přizvat kolegy na spolupráci na dokumentech a můžou paralelně dokument upravovat. Odpadá únavné a nepřehledné posílání si dokumentů a zjišťování, která verze je současná, nebo čekání na schválení reportu. Všichni mohou pracovat na stejném dokumentu, tabulce nebo prezentaci kdykoliv se jim to hodí.[6]

- **Integrace**. Jednotlivé aplikace a služby od Google mohou komunikovat s dalšími populárními programy. Například Google dokumenty dovolí importovat a exportovat soubory z Microsoft Excel, Open Office a jiných. Dále je možná integrace se současnou infrastrukturou IT v organizaci. Pro ukázku, lze při migraci z původního mailového serveru do Gmailu zachovat staré zprávy, jejich stav přečtena/nepřečtena, organizaci mailových zpráv apod.[6]

- **Aktualizace a opravy chyb**. Protože jsou aplikace od Google webové, Google se automaticky stará o jejich aktualizaci a opravy chyb. Odpadá starost administrátora o instalaci nových verzí programů a jejich oprav, zároveň odpadá starost, že některé služby přestanou fungovat nebo fungují jinak, než se původně předpokládalo.[6]

- **Vyhledávání**. Webové služby Google Apps nabízejí také známý vyhledávací nástroj od Google. Ten umožňuje vyhledávat mimo jiné v dokumentech, emailech, kalendářích aj. Například Gmail nabízí tolik úložného prostoru, že uživatel již nemusí řešit problém s místem a mazat staré zprávy, a tak může pomocí vyhledávače efektivně procházet všechny své zprávy za dobu používání služby.[6]

- **Administrace**. Pokud se spravuje doména (jako ve verzi Google Apps pro vzdělávání), mohou se vytvářet doménové Google Apps účty, které sdružují a spojují všechny aplikace v organizaci. GA mohou být užitečné pro jakoukoliv skupinu spolupracovníků, nedbaje na jejich velikost. Pomocí několika kliknutí může administrátor třeba nastavit pravidla sdílení v doméně, jestli povolí veřejné sdílení dokumentů a kalendářů i mimo doménu apod.[6]

- **Práce online i offline**. V březnu 2008 poskytl Google novou funkcionalitu: možnost pracovat na Google dokumentech i offline. Předtím byla největší výhrada k používání Dokumentů nutnost uživatele být připojený k internetu. Bylo sice možné dokumenty exportovat a pracovat s nimi v programech jako Microsoft Word či Excel, ale uživatel musel myslet na to, že soubory je třeba zase zpět nahrát do Google Dokumentů. Nyní lze pracovat na dokumentech nezávisle na připojení a nutnosti cokoliv exportovat. Pokud tak uživatel jede například na služební cestu a nemá připojení k internetu, pracuje na dokumentu offline a ve chvíli, kdy se připojí k internetu, automaticky se nahraje současná verze.[6]

- **Bezpečnost**. Data uživatelů neleží pouze na jediném počítači na centrále Google. Společnost vlastní cloud serverů rozmístěných na mnoha různých lokacích ve světe. Namísto držení dat na jednom místě jsou informace distribuovány po serverech, a tím pádem i několikrát zálohována. Také to ztěžuje potenciálním útočníkům nějakým způsobem měnit nebo znehodnocovat data. Tato strategie přináší značné výhody oproti uchovávání dat na jediném počítači. Pokud by třeba útočník ukradl uživateli osobní notebook, sice se nemusí dostat kvůli neznalosti hesla k informacím uloženým na počítači, ale stejně tak se k datům nedostane už ani původní uživatel. Podobná situace ztráty dat nastává při zničení počítače nebo pevného disku.[6]

  Google se stará o bezpečnost dat velmi vážně. Bezpečnostní týmy se pokouší detekovat a eliminovat počítačové virusy, červy a spamy. Ochranu proti spamu například zabezpečuje službu Gmail. Google používá kombinaci sofistikovaných filtrů a uživatelských reportů, aby nedovolil spamům dostat se do doručené pošty. Nikdy není ochrana 100%, ale i tak je velmi účinná vzhledem k faktu, že spam tvoří přes 90% emailové komunikace v kyberprostoru.[6]

  Většina Google aplikací podporuje SSL (Secure Sockets Layer), který šifruje data a komunikaci. Uživatel ví, že je v bezpečném módu když se nachází na webové adrese prohlížeče začínající na *https://* namísto běžné *http://*. Kdykoliv se uživatel přihlašuje ke Google účtu, provádí to přes zabezpečené spojení, takže zamezuje komukoliv odchytit jeho uživatelské jméno nebo heslo (za předpokladu že nikdo opravdu není schopen komunikaci rozšifrovat).[6]

- **Soukromí**. Soukromí je další důležitou součástí bezpečnosti. Spekuluje se, že Google zkoumá příchozí emailovou komunikaci a hledá v ní klíčová slova, podle kterých následně vybírá a zobrazuje cílenou reklamu uživateli. Jedná se porušení soukromí? Záleží na každém uživateli. Více o ochraně soukromí a osobních dat se lze dočíst na stránce Ochrana soukromí a smluvní podmínky [13].[6]

- **Cena**. Jednotlivé aplikace od Google jsou zcela zdarma, což je, zvlášť v porovnání s cenami některých desktopových aplikací, velká výhoda. Google nabízí Google Apps ve třech edicích:[6]

  - **standardní edice** - zdarma, pro jednotlivého uživatele
  - **Google Apps for Education** - zdarma, pro vzdělávací zařízení (školy a instituce) a neziskové organizace
  - **Google Apps for Work** - placená verze (nyní 4 €/8 € za účet měsíčně [14])

Výhody webových aplikací od Google vypsané v předcházejícím seznamu na začátku této sekce slouží jako důvod, proč má smysl se integrací Google

[13] `http://www.google.com/intl/cs/policies/privacy/`
[14] `https://apps.google.com/pricing.html`

Apps vůbec zabývat. I kvůli těmto plusovým bodům, hlavně nulové ceně, robustnosti, bezpečnosti a širokém povědomí o aplikacích od Google byly Google Apps vybrány pro integraci do aplikace INUM.

„Cíl nahradit původní emailové řešení spravované organizací za outsourcované řešení pro studenty od Google Apps byl úspěšně dosažen. Obecně jsou uživatelé velmi spokojeni s variabilitou a konzistencí nabízených služeb. Emailové rozhraní Google Mailu je sice podstatně odlišné od tradičních emailových programů, ale to je pouze malá překážka, protože uživatelé mají stále možnost používat tradiční emailové klienty k přístupu k jejich poště pomocí protokolů POP a IMAP. Mnoho uživatelů již znalo prostředí Gmailu, a pro ty, kteří neznali, byla dostatečnou pomocí k zvyknutí si na nové prostředí použití nápovědy a dokumentace od Google. Integrací s ostatními službami Google Apps jako Kalendář, Talk, Dokumenty nebo Stránky získali uživatelé velkou výhodu a pomoc ve spolupráci a produktivitě.

Ušetřené náklady díky implementaci Google Apps místo vlastního emailového serveru pro studenty a tradičních desktopových kancelářských nástrojů jsou obrovské. Dokonce i pro běžné firmy jsou ceny za Google Apps velmi výhodné a konkurenceschopné. Žádná cena za licence a vybavení dohromady s velmi malými náklady za podporu by měli vyhovovat rozpočtu každé akademické entity. Ve srovnání s $9 miliony za rok za konkurenční produkt, Google Apps se stávají jasnou volbou.“ [14]

Tak velkou úsporu jako na Colorado State University, ze které pochází úryvek z výše uvedené případové studie nasazení Google Apps, nelze na škole SZŠ ČB předpokládat, už jenom kvůli nepoměrně menšímu počtu personálu a studentů. Ovšem další důvody k integraci už platí bez výjimek.

Jak bylo již zmíněno v sekci 2.1.4, nasazení Google Apps na SZŠ ČB provedl ve své bakalářské práci „Integrace Google Apps“ Luboš Palíšek. Od té doby se však diametrálně změnila většina dostupných API pro správu objektů v doméně GA, že bylo nutné kompletně aplikaci změnit. Větší smysl ale dávalo delegovat komunikaci s GA na aplikaci INUM, než udržovat dvě separátní aplikace s podobným účelem - synchronizovat uživatelská data.

Nyní je nutné se podrobně seznámit s nabízenými API pro Google Apps.

2.3.1 Directory API

Directory API [15] dovoluje administrovat uživatele, skupiny uživatelů, organizační jednotky, zařízení a jiné v doméně Google Apps.

Jedná se o REST API zabezpečené protokolem OAuth 2.0 (více o protokolu OAuth v sekci *4.5 OAuth*). Princip spočívá v komunikaci přes protokol HTTP a výměně zpráv ve formátu JSON.

Základem pro všechny služby Google Apps je Google účet. Účet je kromě vnitřního ID Google identifikován také unikátní emailovou adresou. Například pro uživatele Frank Underwood a doménu SZŠ ČB s identifikátorem `szscb.cz` může být jeho emailová adresa `underwood.frank@szscb.cz`. Jeho vložení skrze GA API by probíhalo přes HTTP dotaz (*request*) na adresu

[15] `https://developers.google.com/admin-sdk/directory/`

`https://www.googleapis.com/admin/directory/v1/users` pomocí HTTP metody POST. Tělo dotazu by pak mohlo vypadat podobně jako v kódu 1.

```
{
  "kind": "admin#directory#user",
  "id": "123456789123456789123456789",
  "etag": "sdladklsau-XzORg0bILYPVc/dskad26cvCCuMHq1hUINs",
  "primaryEmail": "underwood.frank@szscb.cz",
  "name": {
    "givenName": "Frank",
    "familyName": "Underwood",
    "fullName": "Frank Underwood"
  },
  "password": "3fd16b8996b05f69adfcee54c3e8f9bb5e276d1e",
  "hashFunction": "SHA1"
}
```

Ukázka kódu 1: Příklad JSON objektu pro vložení uživatele do domény Google Apps.

Typickými metodami pro manipulaci s objekty v Directory API jsou:

- **insert** - vytvoření nového objektu, typicky identifikovaného unikátním atributem (email pro uživatele nebo skupinu, název pro organizační jednotku)
- **get** - získání konkrétního objektu z GA (přes unikátní klíč - buď ID v GA nebo třeba email)
- **list** - získání stránkovaného seznamu objektů, může být doplněno filtrem
- **delete** - smazání konkrétního objektu z GA
- **update** - úprava konkrétního objektu
- **patch** - podobné jako update, s tím rozdílem, že se neposílá celý objekt, ale jen část, která se upravuje

Konstruovat dotazy na Google Apps lze ručně, ale velkým zjednodušením jsou dostupné knihovny pro známé programovací jazyky, mimo jiné i Java [16]. Použití knihovny usnadňuje autorizaci aplikace vůči GA a konstrukci všech výše zmíněných dotazů.

Kromě správy uživatelů, skupin uživatelů, organizačních jednotek (do kterých se vkládají uživatelé) nabízí Directory API také ovládat tzv. zdroje kalendářů (*Calendar Resources*). Tato část nahradila již zastaralé samostatné API (bude ukončeno v lednu 2017) [17] a je důležitá pro synchronizaci kalendářů tříd.

[16] `https://developers.google.com/admin-sdk/directory/v1/quickstart/java`
[17] `https://developers.google.com/admin-sdk/calendar-resource/`

2.3.2 Calendar API

API pro kalendáře funguje na stejném principu jako Directory API. Dokonce se jejich použití částečně překrývá. Kalendáře a jejich události lze totiž vkládat buď konkrétnímu uživateli do jeho osobního kalendáře, nebo do tzv. zdrojů kalendářů, které představují například kalendář pro použití místnosti nebo právě rozvrh třídy. Těmto zdrojům se následně přes Calendar API přidávají události (angl. *event*). Calendar API opět poskytuje knihovnu pro programovací jazyk Java [18].

2.3.3 GData API

Google Data Protocol [19] je REST technologie pro čtení, zápis a úpravu informací pro aplikací od firmy Google. Některá starší API jej stále používají, mezi nimi i tzv. sdílené kontakty domény [20] (angl. *Domain Shared Contacts*).

Kontakty slouží primárně jako zdroj informací o lidech, především jsou pak použitelné v našeptávačích - například při psaní emailu se při výběru adresáta po zadání několika písmen vyhledávají kontakty se shodným jménem nebo emailovou adresou v adresáři autora emailu.

Sdílené kontakty domény jsou externí kontakty, které jsou sdíleny mezi všemi uživateli domény. Mezi takové kontakty se nepočítají samotní uživatelé domény, ti mohou být totiž viditelní mezi sebou navzájem (záleží na nastavení sdílení). V situaci, kdy by v doméně Google Apps byli umístěni jak všichni učitelé, tak žáci a studenti ze SZŠ ČB, nebylo by třeba sdílené kontakty domény implementovat. Žákům a studentům se ale účty nevytvářejí, proto jsou sdílené kontakty tím pravým místem, kam informace, jako emailové adresy či příslušnost ke třídě, umístit.

Další možností by bylo ke každému doménovému uživateli informace o kontaktech umístit do jeho osobního adresáře kontaktů, tímto krokem by však úměrně narostla složitost a zkomplikovala se centrální správa kontaktů z aplikace INUM. Pokud by přece jenom byla vybrána tato možnost, synchronizace by byla prováděna přes Contacts API [21], které je opět podobné Directory API a Calendar API.

GData API bylo dříve základem většiny API pro správu Google Apps, například pro Provisioning API a Profiles API (viz sekce *2.1.4 Google Apps*). V současné situaci je pro případ této diplomové práce využitelné již jen pro sdílené kontakty domény, pokud by se účty v doméně GA vytvářely i žákům a studentům, pak využití toho API zcela odpadá.

GData API se příliš zásadně neliší principem fungovaní od Directory API nebo Calendar API, jehož největším problémem však byla příliš velká komplexnost. Google se rozhodl jednotlivé části API přemístit a roztříštit mezi více komponent, API pro adresář, kalendáře a kontakty představují jedny z takových. [11]

[18] `https://developers.google.com/google-apps/calendar/quickstart/java`
[19] `https://developers.google.com/gdata/`
[20] `https://developers.google.com/admin-sdk/domain-shared-contacts`
[21] `https://developers.google.com/google-apps/contacts/v3/`

Jedním z největších rozdílů je forma, ve které jsou posílána data o objektech spravovaných přes API. GData API používá primárně XML standard Atom, i když je schopný komunikovat i pomocí JSON objektů. Pro příklad: vložení nového sdíleného kontaktu do doménového adresáře Google Apps by probíhalo voláním URL `https://www.google.com/m8/feeds/contacts/szscb.cz/full` HTTP metodou `POST` s daty ve formátu XML v podobě, jaká je vidět v kódu 2.

```
<atom:entry xmlns:atom='http://www.w3.org/2005/Atom'
    xmlns:gd='http://schemas.google.com/g/2005'>
  <atom:category
  scheme='http://schemas.google.com/g/2005#kind'
  term='http://schemas.google.com/contact/2008#contact' />
  <gd:name>
     <gd:givenName>John</gd:givenName>
     <gd:familyName>Doe</gd:familyName>
     <gd:fullName>[1.A] John Doe</gd:fullName>
  </gd:name>
  <gd:email
  rel='http://schemas.google.com/g/2005#home'
    primary='true'
    address='john.doe@google.com'/>
  <gd:phoneNumber
  rel='http://schemas.google.com/g/2005#home'
    primary='true'>
    789456123
  </gd:phoneNumber>
  <gd:organization>
    <gd:orgName>SZS CB</gd:orgName>
    <gd:orgTitle>student [1.A]</gd:orgTitle>
  </gd:organization>
</atom:entry>
```

Ukázka kódu 2: Příklad XML objektu pro vložení sdíleného kontaktu do domény Google Apps.

2.4 Konkurence

Aplikace INUM začala vznikat v průběhu roku 2013 jako produkt bakalářské práce, její rozšířená verze v rámci diplomové práce v roce 2015. V tom samém roce se objevilo i řešení SkolniLogin.cz [22], které je do značné míry svým principem podobné aplikaci INUM.

Integruje běžné školní aplikace, mimo jiné Active Directory i Bakaláři. Základ nástroje je položen na Office 365, což je platforma podobná Google

[22] `https://www.skolnilogin.cz/`

Apps od společnosti Microsoft. Poskytuje email, kalendář, kontakty apod.

Velkou výhodu SkolniLogin.cz představuje napojení na Eduroam a možnost přihlášení uživatelů do aplikace Bakaláři, které je umožněno právě použitím Office 365, respektive souvisejícím vytvořením účtu na Windows Live, pomocí kterého Bakaláři povolují přihlašovat uživatele do svého systému. Také nabízí dostupné API pro vývojáře pomocí protokolu OAuth 2.0.

SkolniLogin.cz však není zdarma, podle dostupných informací se po měsíční zkušební lhůtě platí pravděpodobně paušální poplatek za podporu a běh aplikace. Další nevýhodou by mohla být příliš velká závislost na prostředí operačního systému Microsoft Windows.

Kapitola 3

Analýza systémové integrace

Kapitola se zaměřuje na popis a výběr integračních vzorů a vhodné architektury při implementování nástroje INUM.

3.1 Vzory integrace aplikací

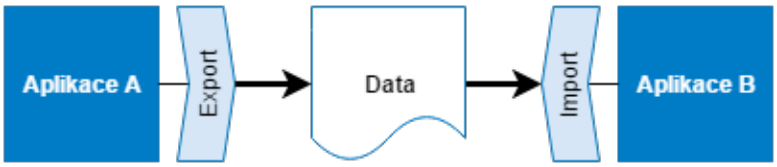
Integrace aplikací má za cíl umožnit rozličným aplikacím pracovat společně, aby nabízely ucelený shluk funkcionalit. Aplikace mohou být buď vyvíjeny organizací nebo nakoupeny od dodavatelů. Pravděpodobně budou instalovány na různých počítačích, které mohou reprezentovat různorodé platformy, a mohou být také fyzicky rozmístěny víceméně po celém světě. Některé aplikace mohou běžet mimo prostředí organizace, například u obchodních partnerů nebo zákazníků. Jiné aplikace zase nemusejí být navrhovány s myšlenkou integrace nebo se velmi těžce mění, takže jejich spojení může být komplikované. Tato sekce nabízí přehled přístupů k integraci, které mohou překonat tyto problémy. [15]

Ke každému integračnímu vzoru bude uvedeno, zda a proč se hodí/nehodí při implementaci zadání této diplomové práce, s důrazem na první cíl práce, tedy vytvořit jeden uživatelský účet pro všechny aplikace.

3.1.1 Přenos souborů

Nejjednodušší přístup k integraci nastává, když jedna aplikace produkuje soubory a poskytne je ostatním systémům. Tento princip je jednoduchý a všeobecně použitelný, protože vše potřebné je pro interagující aplikace schopnost číst a zapisovat soubory. Díky nízké náročnosti na požadavky integrace založené na souborech se jedná o běžně používané řešení, některá omezení souborového systému ale znamenají, že integrované aplikace se mohou stát komplexnějšími. [10]

Důležité rozhodnutí při integraci skrze soubory je vybrat formát souborů. Velmi zřídka bude výstup jedné aplikace odpovídat přesně vstupu druhé aplikace, takže se musí během cesty provést určité zpracování nebo předělání souborů. To znamená, že nejenom všechny aplikace, které soubor využívají, musí být schopny číst jeho obsah, ale musí být také schopny použít na



Obrázek 3.1: Integrační vzor přenos souborů (File Transfer).

souboru nějaké nástroje pro jeho zpracování a transformaci. Kvůli tomu se časem vyvinuly standardy pro formáty souborů. Mainframe systémy využívají datové zdroje založené na systémovém souborovém formátu COBOL, UNIXové systémy zase textové soubory. V současné době je preferován formát souborů XML. Kolem těchto formátů se vyvinuly nástroje pro čtení, zápis a transformaci. [15]

Dalším problémem kolem souborů je určení, kdy je vytvářet a kdy je zpracovávat. Protože vytvoření a zpracování souboru vyžaduje úsilí a systémové prostředky, není většinou žádoucí, aby se se soubory pracovalo příliš často. Typicky existuje cyklus, který řídí toto rozhodnutí - zpracovávat soubory vždy v noci, jednou týdně, čtvrtletně apod. Aplikace tento cyklus znají a zpracovávají soubory právě v tomto určeném čase. [15]

Velká výhoda souborů je, že integrátoři aplikací nemusí znát vnitřní fungování jednotlivých aplikací. Vývojáři aplikace obvykle poskytnou soubor, pouze se dohodnou s integrátory na formátu a obsahu. Integrátoři pak vyřeší transformaci souborů nebo nechají cílovou aplikaci rozhodnout, jak manipulovat a číst data. Aplikace jsou výsledkem jasně oddělené jedna od druhé. Každá aplikace pak může provádět interní změny bez ohledu a efektu na jinou aplikaci, za předpokladu, že produkuje stále ten samý shodný soubor ve smluveném formátu. Soubory se efektivně stávají vnějším rozhraním jednotlivých aplikací. [15]

Přenos souborů (angl. *File Transfer*) je jednoduchý a nevyžaduje extra nástroje nebo integrační balíky, ale také znamená velkou zátěž na vývojáře, který musí udělat mnoho práce ručně. Aplikace se musí dohodnout na konvenci pojmenování souborů a na adresářích, kam je budou ukládat. Zapisující aplikace musí implementovat strategii, která zaručí unikátní pojmenování souborů. Aplikace také musí určit, která z nich soubor smaže, z čehož plyne, že také musí vědět, kdy již soubor není dále použitelný a potřebný. Aplikace potřebují implementovat zamykací mechanismus a zajistit, aby v čase, kdy chce jedna aplikace soubor číst, v ten samý moment nechtěla druhá aplikace zapisovat. Pokud obě aplikace nemají přístup na stejné úložiště, jedna z aplikací musí převzít zodpovědnost za přenesení souboru na jiné úložiště, odkud může druhá aplikace číst. [15]

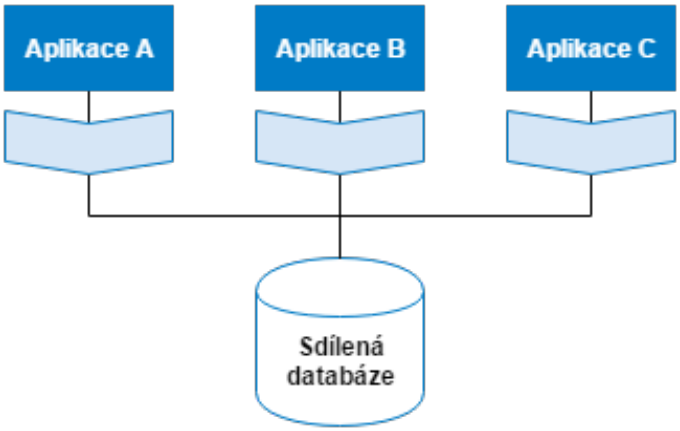
Změny v datech nastávají nepravidelně a jako výsledek mohou být systémy v nesynchronizovaném stavu, což je jedním z největších problémů přenosu souborů. Například systém pro správu zákazníků zpracuje změny poštovní adresy zákazníka a vystaví soubor se změněnými daty v noci, platební brána ale ještě v ten samý den před zpracováním souboru vystaví fakturu na původní adresu. Občas není nesynchronizovanost tak velkým problémem, uživatelé

jsou zvyklí na určitou dobu odezvy, než se informace správně propagují. Jindy však použití staré informace může způsobit problém či dokonce katastrofu. Při rozhodování o frekvenci přenosu souboru se musí brát v potaz potřeby a nároky na aktuálnost informací v jednotlivých systémech. [15]

3.1.2 Sdílená databáze

Přenos souborů dovoluje aplikacím sdílet data, ale postrádá aktuálnost, která je často kritická při integraci aplikací. Pokud se změny rychle nepromítnou do všech sdružovaných aplikací, velmi pravděpodobně se vyskytnou problémy kvůli porušení integrity dat. Pro moderní byznys je nezbytné mít aktuální a správná data. Nejenže se tím redukují chyby, ale také roste důvěra uživatelů vůči datům a samotným aplikacím. [15]

Databáze jsou pokročilejší mechanismy pro ukládání dat než soubory a zmírňují některá omezení souborového systému. Poskytují atomické operace, lépe definované datové struktury a mechanismy k zajištění datové konzistence. [10]



Obrázek 3.2: Integrační vzor sdílená databáze (Shared Database).

Časté aktualizace také pomáhají lépe zvládnout inkonzistenci. Čím častěji se synchronizuje, tím menší je pravděpodobnost, že nastanou inkonzistence a stojí méně námahy se s nimi vypořádat. Ale ať jsou změny jakkoliv časté, pořád budou nastávat problémy. Pokud se bude například adresa aktualizovat nekonzistentně několikrát rychle za sebou, jak lze určit, která z adres je ta pravá? Šlo by u každé části dat určit jednu hlavní aplikaci zdrojových dat, pak se ale musí na toto určení neustále pamatovat. [15]

Když skupina integrovaných aplikací závisí na jedné a té samé databázi, je celkem jisté, že budou konzistentní v každém okamžiku. Pokud se simultánně aktualizují data z více zdrojů, transakční systémy se o konzistenci postarají tak dobře, jak jen to je možné. Protože jsou intervaly mezi aktualizacemi velmi malé, jakákoliv chyba se snadněji nalezne a opraví. [15]

Integrační styl přes **sdílenou databázi** (angl. *Shared Database*) je jednodušší na implementaci díky velkému rozšíření SQL relačních databází. Většina

aplikačních platforem umí pracovat s SQL, často pomocí velmi sofistikovaných nástrojů. Takže se integrátor nemusí starat o velké množství formátů souborů. Protože skoro každá aplikace používá SQL, vyhýbá se tak použití další technologie, kterou musí zvládnout všechny aplikace. [15]

Jednou z největších potíží se sdílenou databází je vymyslet vhodný návrh. Vytvořit sjednocené schéma, které bude odpovídat požadavkům několika aplikací, může být velmi složité až nereálné, často pak vyústí ve schéma, jež programátoři považují za obtížně spravovatelné a využitelné. [15]

Ze sdílené databáze několika aplikací, které často čtou a mění stejná data, se lehce může stát výkonnostní úzké hrdlo, způsobující zámky na datech a nemožnost dalším aplikacím k datům vůbec přistupovat. Když jsou aplikace distribuované na více místech, přistupování k jedné sdílené databázi je typicky pomalé, aby mělo praktické využití. [15]

3.1.3 Volání vzdálených procedur

Přenos souborů a sdílená databáze dovoluje aplikacím sdílet jejich data, což je důležitá část integrace aplikací, pouze sdílení ale není často dostačující. Změny v datech obvykle vyžadují provedení určitých akcí nad různými aplikacemi. Například, změna adresy je jednoduchá změna v datech, ale také může spustit registraci nebo právní procesy, které musí být provedeny v dalších systémech. Pokud by měla jedna aplikace vyvolat takové procesy změn přímo v ostatních aplikacích, vyžadovalo by to její znalost interního fungování všech dalších návazných systémů. [15]

Obrázek 3.3: Integrační vzor volání vzdálených procedur (Remote Procedure Call).

Tento problém reflektuje klasické dilema při návrhu aplikací. Jedním z nejvíce silných mechanismů při návrhu aplikací je zapouzdření (angl. *encapsulation*), kdy moduly schovávají svá data a poskytují je pouze pomocí volání funkcí. Takto mohou reagovat na změny v datech a interně provést další akce související s konkrétní změnou dat. Sdílená databáze poskytuje velkou a nezapouzdřenou strukturu dat, která velmi stěžuje volání takových návazných akcí. Přenos souborů dovoluje aplikacím reagovat na změny v momentě zpracování souboru, ale celý proces je opožděn. [15]

Nezapouzdřenost sdílené databáze také stěžuje údržbu integrovaných aplikací. Změny v některé aplikaci mohou způsobit změny v databázi a taková změna může mít vedlejší efekty v dalších aplikacích. Kvůli tomuto problému je většina organizací používajících sdílenou databázi odmítavá vůči jakýmkoliv změnám v databázích, což velmi zpomaluje vývoj aplikací a navíc snižuje možnost reakce na změny způsobené byznys požadavky. [15]

Volání vzdálených procedur (angl. *Remote Procedure Invocation*) aplikuje princip zapouzdření při integraci aplikací. Když aplikace potřebuje nějaké informace vlastněné jinou aplikací, zeptá se aplikace přímo. Když jedna aplikace potřebuje změnit data v jiné aplikaci, udělá to přes volání funkce druhé aplikace. To umožňuje každé aplikaci zajistit vlastní integritu dat, která obsahuje. Navíc, každá aplikace může měnit formát vnitřních dat bez vedlejšího efektu na další aplikace. [15]

Volání vzdálených procedur (také Remote Procedure Call - RPC) implementuje mnoho technologií, jako například CORBA, COM, .NET Remoting nebo Java RMI (Remote Method Invocation). Technologie se liší systémy, které je podporují, a tím, jak snadno jsou použitelné. Obvykle tato prostředí přidávají další schopnosti, jako třeba transakce. Díky své všudypřítomnosti jsou současnými favority webové služby (angl. *Web Services*), které používají standardy jako SOAP a XML. Webové služby vynikají hlavně využitím protokolu HTTP, díky kterému snadno procházejí firewally. [15]

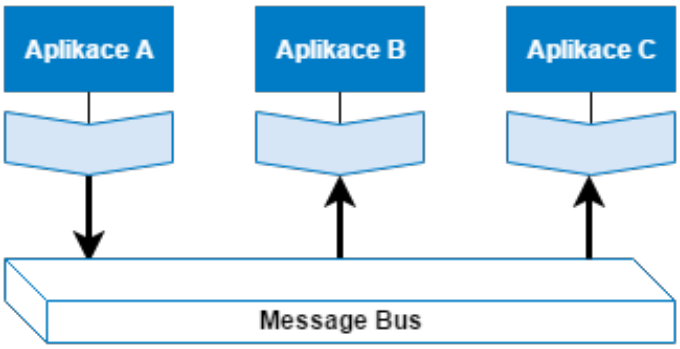
Protože v RPC metody obalují data, lze se snadněji vypořádat se sémantikou dat. Aplikace mohou poskytovat několik různých rozhraní pro stejná data umožňující některým klientům používat jeden styl a druhým jiný. I aktualizace mohou používat více rozhraní. To přináší více možností podporovat více úhlů pohledu na data než při relačním pohledu. Je však nešikovné pro integrátory přidávat transformační komponenty, takže každá aplikace si musí dohodnout vlastní rozhraní se sousední aplikací. [15]

Softwaroví vývojáři jsou zvyklí na volání procedur, takže dobře zapadá do jejich zvyklostí. Ovšem to se ukazuje spíše jako nevýhoda než výhoda. Mezi lokálním a vzdáleným voláním procedur je velký rozdíl, jak ve výkonu, tak ve spolehlivosti. Pokud vývojáři těmto rozdílům nerozumí, volání vzdálených procedur může vést k pomalým a nespolehlivým systémům. [15]

Přestože zapouzdření pomáhá redukovat spojování (angl. *coupling*) aplikací eliminací velkých sdílených datových struktur, aplikace jsou stále velmi úzce spojeny dohromady. Vzdálená volání podporovaná každým systémem mají tendenci spojovat systémy do jednoho velkého zvětšujícího se svazku (uzlu). Zejména sekvenční zpracování (dělání věcí v konkrétním pořadí) může ztížit změnu systémů nezávisle. Takové typy problémů často nastávají, protože překážky, které se jeví jako podřadné v jedné aplikaci, se stanou signifikantní při integraci více aplikací. Lidé častokrát navrhují integraci tak, jako by navrhovali pouze jednu aplikaci, nevědomi si velkých změn při vázání systémů dohromady. [15]

3.1.4 Zasílání zpráv

Přenos souborů a sdílená databáze umožňují aplikacím sdílet jejich data, ale ne jejich funkcionalitu. Volání vzdálených procedur umožňuje aplikacím sdílet i jejich funkcionalitu, ale velmi úzce je spojuje dohromady. Častou výzvou při integraci je udělat spolupráci mezi oddělenými systémy co nejvíce včasnou a bez svázání systémů dohromady (tak, že by se staly nespolehlivé ve smyslu běhu aplikace a jejího vývoje). [15]



Obrázek 3.4: Integrační vzor zasílání zpráv (Messaging).

Přenos souborů dovoluje udržovat aplikace oddělené, nesvázané, ale za cenu včasnosti dat. Systémy prakticky nejde udržet souběžně ve stejném stavu, spolupráce je příliš pomalá. Sdílená databáze drží data pohromadě a poskytuje je okamžitě, ale za cenu sdružování dat dohromady. Také selhává při spolupráci systémů. [15]

Po zvážení všech těchto problémů vypadá vzdálené volání procedur jako nejvhodnější volba. Rozšíření modelu jedné aplikace na integraci více aplikací ale přináší jiné slabé stránky, jako například základní problém distribuovaných systémů. Přestože volání vzdálených procedur vypadá podobně jako volání lokálních, nechová se stejně. Vzdálená volání jsou pomalá a více náchylná k chybám. S několika aplikacemi komunikujících v jednom prostředí je nežádoucí, aby při pádu jedné aplikace spadly i všechny ostatní. Také není chtěné navrhovat systémy vycházející z předpokladu, že volání jsou rychlá, a dále není chtěné, aby aplikace věděly detaily o dalších aplikacích, i kdyby se jednalo pouze o znalost jejich rozhraní. [15]

Naopak chtěné chování je mít něco jako přenos souborů ve smyslu přenosu spousty malých datových paketů, které lze vyrobit a přenést rychle a snadno, a aby byla cílová aplikace automaticky notifikována o nově vzniklém paketu, který je připraven ke zpracování. [15]

Přenos dat potřebuje mechanismus pro opakování, aby bylo zajištěno v případě selhání přenosu jeho opětovné úspěšné provedení. Detaily jakékoliv struktury dat na disku nebo v databázi musí být schovány před aplikacemi tak, aby na rozdíl od sdílené databáze nebylo složité provést změny v návaznosti na požadavky. Jedna aplikace by měla být schopna poslat paket dat jiné aplikaci, kde se spustí návazná akce (podobně jako ve volání vzdálených procedur), ale bez tendence k vyvolání chyb. Přenos dat by měl být asynchronní tak, že odesílatel nemusí čekat na přijímač, hlavně v momentě, kdy je třeba opakované volání. [15]

Asynchronní zprávy jsou základní reakcí na problémy distribuovaných systémů. Zaslání zprávy nevyžaduje, aby oba systémy běžely ve stejnou chvíli. Navíc myšlení v asynchronním stylu nutí vývojáře rozpoznávat, že práce se vzdálenými aplikacemi je pomalejší, což napomáhá navrhovat komponenty velmi soudržné (angl. *high cohesion*, mnoho práce se dělá lokálně) a málo přilnavé (angl. *low adhesion*, selektivní a malá práce vzdáleně). [15]

Systémy **zasílání zpráv** (angl. *Messaging systems*) také dovolují stejnou míru separace jako při použití přenosu souborů. Zprávy mohou být transformovány při přenosu bez vědomí odesílatele i přijímače. Separace (angl. *decoupling*) umožňuje integrátorům vybrat mezi rozesíláním zpráv více přijímačům, směrováním zprávy právě jednomu přijímači a dalšími možnými topologiemi. Tím se odděluje integrace od vývoje jednotlivých aplikací. [15]

3.1.5 Výběr integračního vzoru

Z výše popsaných důvodů lze vidět, že integrace pomocí **přenosu souborů**, respektive **exportu a importu**, je velice primitivní způsob spojení aplikací, který je na jednu stranu jednoduchý na implementaci, na druhou stranu však přináší řadu problémů, se kterými se musí integrátor aplikací vypořádat.

Při integraci systémů na škole SZŠ ČB je tento vzor vhodný právě v jednom případě - při integraci systému Bakaláři s aplikací INUM. Bakaláři totiž neposkytují žádné API, skrz které by se dalo komunikovat, a už vůbec nemají nástroje, kterými by se aplikace dala připojit na systém zasílání zpráv, protože se jedná o aplikaci třetí strany.

Integrace nebude probíhat čistě podle integračního vzoru, tedy exportovat z Bakaláři do souboru, který následně bude zpracovávat aplikace INUM. Bakaláři totiž neumožňují ani export dat (programově volaný), takže je nutné se přímo připojit do příslušné databáze, přečíst požadovaná data, transformovat na struktury, kterým rozumí INUM a následně uložit do databáze INUM. O veškerou práci se postará aplikace INUM, Bakaláři jsou pouze pasivním účastníkem takových transakcí.

Sdílená databáze není využitelná při návrhu společného účtu, převážně kvůli aplikaci Bakaláři. Ta obsahuje vlastní databázi (Microsoft SQL Server), úzce vázanou na aplikaci a prakticky neměnitelnou, protože při jakékoliv změně by mohla aplikace zcela přestat fungovat. Bakaláři jsou primárním zdrojem dat pro integraci účtů, systém není schopen pracovat s jinou databází než právě MSSQL, a protože na chodu aplikace závisí fungování celé školy (agenda žáků a učitelů, docházka, známkování, napojení na matriky, rozvrh hodin atd.), nelze zvolit jinou sdílenou databázi (například by se mohlo nabízet použít Active Directory).

Dalšími velkými nevýhodami, kromě rozšiřitelnosti a spravovatelnosti, databáze Microsoft SQL Server, pokud by se využívala jako sdílená databáze, jsou její vázanost na operační systém Windows a licencování. Verze Microsoft SQL Server 2008 Express Edition with Advanced Services (64 bit) nainstalovaná na SZŠ ČB je sice zdarma, ale omezena na použití 1 jádra CPU, 1GB RAM a maximálně 4GB velikosti databáze. [27]

Volání vzdálených procedur bude nejpoužívanější technikou v aplikaci INUM, převážně díky tomu, že pod něj spadá používání API třetích stran. Pro integraci Bakalářů z důvodů zmíněných v podsekci *3.1.1 Přenos souborů* RPC použít nelze, ale vzhledem k existenci API hlavně pro Active Directory a Google Apps se RPC jeví jako optimální volba.

Samozřejmě by bylo vzhledem k četným výhodám zasílání zpráv, hlavně

asynchronní podstatě komunikace, použít tento návrhový vzor, jenž však vyžaduje nasazení message bus (middleware). Největší překážk pro použití **zasílání zpráv** však představuje nutnost integrovaných aplikací vyzvedávat a zpracovávat příslušné zprávy z front zpráv. Jak již bylo řečeno, Bakaláři neposkytují vůbec žádné API, takže kromě požadavku na doprogramování rozhraní na firmu vyvíjející aplikaci Bakaláři, které by mohlo následně splnit podmínky pro možnost využití volání vzdálených procedur nebo dokonce zapojení do komunikace přes zprávy, v tomto směru neexistuje jiné řešení, jak aplikaci Bakaláři integrovat. Google Apps a Active Directory disponují již zmíněnými API, opět je ale obtížné, respektive prakticky neproveditelné, donutit samotné systémy, aby pracovaly s frontami zpráv. Pokud nedokáží plnit svoji roli v publish-subscribe stylu komunikace a příslušná akce, aby zpracovaly zprávy ve frontě musí být vyvolána třetí stranou, vytrácí se hlavní výhoda vzoru zasílání zpráv - asynchronnost, ta předpokládá, že si aplikace zprávu vyzvedne v momentě, kdy je v běžícím stavu a sama vyhodnotí vhodný čas pro její zpracování.

3.2 Integrační architektury

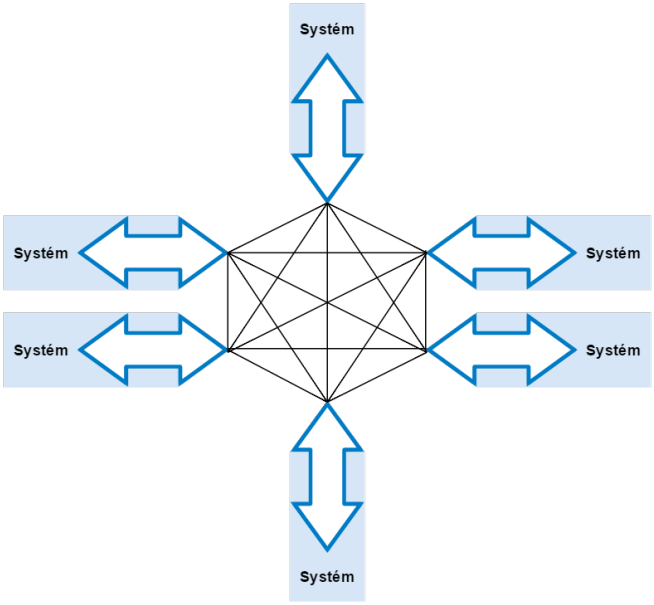
Důležitým krokem v porozumění integrační architektuře je porozumět různým existujícím topologiím integrace a rozpoznat hlavní rozdíly mezi nimi. Když integrátor skutečně těmto rozdílům porozumí, má napůl vyhráno, protože dokáže správně aplikovat vhodnou architekturu na daný konkrétní integrační problém.[1]

3.2.1 Point-to-Point řešení

Point-to-Point architektura (bod k bodu, každý s každým, na obrázku 3.5) definuje integrační řešení pro dvojice aplikací, uvažuje se o dvou koncových bodech, které spolu komunikují. Budují se protokoly a/nebo transformátory či formátovací adaptéry na jedné nebo druhé straně integrace. Jedná se o nejjednodušší architekturu, dokud se objem a komplexnost integrace drží na nízké úrovni. K realizaci se používají technologická rozhraní jako FTP, IIOP, vzdálená volání nebo dávková zpracování. Výhodou přístupu je těsná soudržnost (angl. *tight coupling*) dvojice aplikací, protože obě mají znalost o druhé v páru.[1]

3.2.2 Hub-and-Spoke řešení

Hub-and-Spoke architektuře (obrázek 3.6) se také říká message broker (zprostředkovatel zpráv) a je podobná point-to-point architektuře ve smyslu, že poskytuje centralizovaný rozbočovač (hub, broker), ke kterému jsou připojeny všechny ostatní aplikace. Charakteristicky v hub-and-spoke architektuře jsou aplikace připojeny k centrálnímu rozbočovači pomocí lightweight ("odlehčených") konektorů. Lightweight konektory napomáhají integraci aplikací



Obrázek 3.5: Integrační architektura Point-to-Point. (inspirováno z [1])

k mimimálním či vůbec žádným interním změnám v zainteresovaných aplikacích.[1]

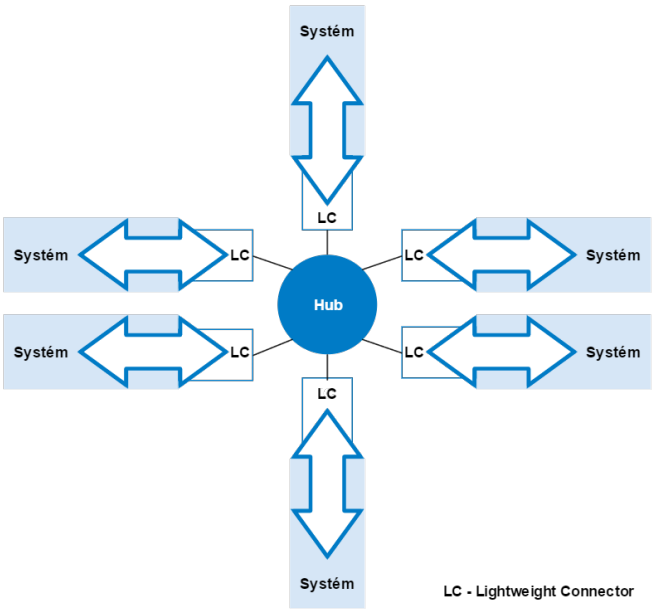
Transformace zpráv a směrování (angl *routing*) se odehrává uvnitř brokeru. Tato architektura je velkým vylepšením vůči point-to-point řešení, protože redukuje počet spojení požadovaných pro integraci. Poněvadž se aplikace nepřipojují k ostatním na přímo, mohou být z topologie kdykoliv odpojeny pouhým odebráním spojení v rozbočovači.[1]

Architektura hub-and-spoke přináší jeden velký zápor. Kvůli vlastnosti centralizace rozbočovače se jedná o jediný bod selhání (angl. *signle point of failure*). Když hub selže, selže i celá integrační topologie. Řešením takových problémů je postavit cluster rozbočovačů, tedy logický shluk několika instancí brokerů běžících simultánně a spolupracujících navzájem.[1]

3.2.3 Enterprise Message Bus

Hub-and-spoke architektura používá lightweight konektory k připojení aplikací k centrálnímu rozbočovači, a protože integrované aplikace častokrát potřebují působit odděleně, přichází potřeba je přidávat a odebírat bez efektu na druhé aplikace. Enterprise Message Bus (na obrázku 3.7) poskytuje společnou komunikační infrastrukturu, která není závislá na platformě a programovacím jazyku.[1]

Tato komunikační infrastruktura obsahuje směrování zpráv i kanály pro Publish-Subscribe komunikaci. Aplikace spolu komunikují skrze message bus s pomocí front na požadavky a odpovědi (angl. *request-response queues*). Když chce konzumující aplikace vyvolat určitou službu na jiném poskytovateli,



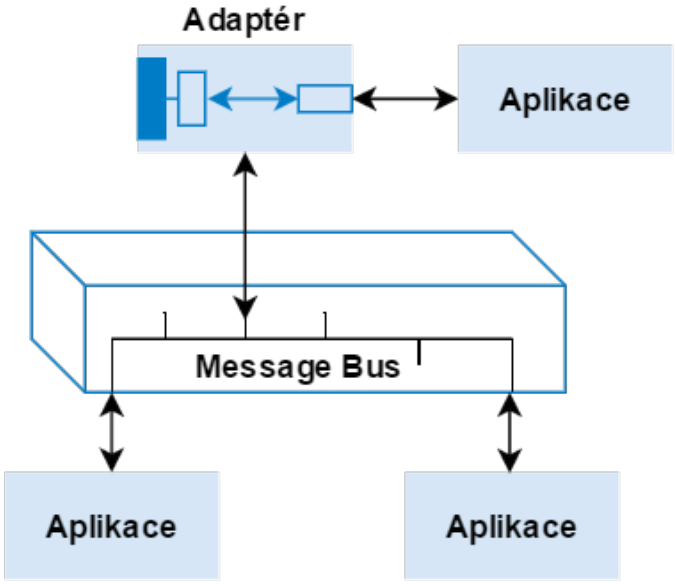
Obrázek 3.6: Integrační architektura Hub-and-Spoke. (inspirováno z [1])

vloží vhodně formátovanou zprávu s požadavkem do fronty požadavků této služby. Následně naslouchá a čeká na zprávu s odpovědí ve frontě služby s odpovědmi. Aplikace poskytující službu naslouchá na požadavky ve frontě, provede příslušnou službu a zašle odpověď do fronty na odpovědi.[1]

3.2.4 Enterprise Service Bus (ESB)

Service bus (rozbočovač služeb, obrázek 3.8) využívá technologie zásobníků k poskytnutí rozbočovače pro integraci aplikací. Různé aplikace nekomunikují na přímo spolu, místo toho komunikují skrze páteřní Service Oriented Architecture (SOA) middleware. Nejvíce významnou výhodou ESB architektury je distribuovaná podstata integrační topologie. Většina ESB řešení je postavena na technologii Web Services Description Language (WSDS) a používají XML formát pro překlad a transformaci zpráv.[1]

ESB je kolekce middleware služeb poskytujících integrační schopnosti. Tyto middleware služby leží v srdci ESB architektury, do které aplikace vkládají zprávy, jež jsou následně směrovány a transformovány. Podobně jako v hub-and-spoke architektuře i v ESB architektuře se aplikace připojují k ESB skrz abstraktní, chytré konektory. Konektory jsou abstraktní ve smyslu, že definují pouze transportní vazební (angl. *binding*) protokoly a rozhraní služeb, nikoliv však implementační detaily. Jsou chytré, inteligentní, protože mají svou logiku postavenou souběžně s ESB k selektivnímu připojování služeb za běhu. Taková schopnost vylepšuje agilitu aplikací možností připojit služby až později než v počáteční fázi a tak i odložit volbu služeb na později.[1]



Obrázek 3.7: Integrační architektura Message Bus. (inspirováno z [1])

3.2.5 Výběr integrační architektury

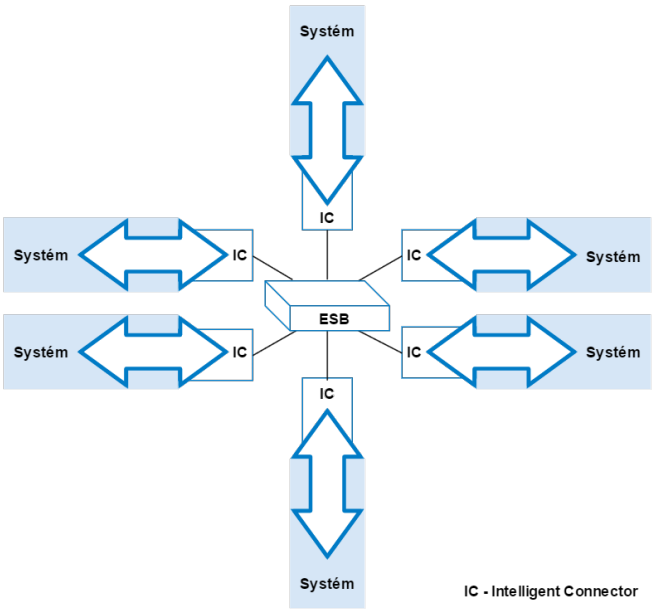
Point-to-Point architektura není v prostředí SZŠ ČB aplikovatelná, už jen například kvůli tomu, že nelze žádným jiným způsobem než pomocí třetí aplikace (INUM) propojit Bakaláře s Google Apps nebo Active Directory.

Enterprise Message Bus a Enteprise Service Bus architektury zase vyžadují běh middlewaru, schopného pracovat s frontami zpráv, respektive službami. Nasazení takového middlewaru má větší náročnost na systémové prostředky (často v jednotkách gigabytů na operační paměť) a značná část poskytovaných řešení není zdarma (Oracle Service Bus, IBM WebSphere ESB a jiné). Existují ale také open-source řešení zdarma, jako například Spring Integration, WildFly či JBoss ESB.

Hlavním důvodem, který by mluvil ve prospěch použití ESB, je možná rozšiřitelnost integrace v budoucnosti, tj. napojení více a více aplikací. Právě vůči takovému postupu ale varuje CTO společnosti MuleSource, která vyvíjí ESB Mule: „Nikdy to nebudete potřebovat (You'll Never Need It - YNNI), tento akronym tu s námi je z nějakého důvodu“, říká Mason. „To je další zabijácký problém. Pokud vyberete ESB, protože si myslíte, že jej budete potřebovat, pak určitě nemáte rozvrženou architekturu jak budete ESB používat, protože jste o ni moc nepřemýšleli. To je varovné znamení. Nasadíte technologii jenom proto, že existuje.“ [7]

Vzhledem k tomu, že se nepředpokládá rozšiřování integrace o další aplikace, zdá se použití ESB jako příliš komplikované, náročné a zbytečně komplexní s ohledem na složitost daného problému. O problémech se systémy zasílání zpráv (Enteprise Message Bus) se diskutuje v podsekci *3.1.5 Výběr integračního vzoru*.

Pro účely implementace diplomové práce byla proto vybrána integrační



Obrázek 3.8: Integrační architektura Enteprise Service Bus. (inspirováno z [1])

architektura Hub-and-Spoke. Aplikace INUM bude v takové architektuře sloužit jako hub - rozbočovač.

Kapitola 4
Analýza mechanismů ověřování

Tato kapitola se věnuje popisu ověřovacích principů, rozdílu mezi autentizací a autorizací a přehledu nabízených možností na poli ověřovacích služeb. Do detailu je vysvětlen princip autentizačního standardu SAML a autorizačního protokolu OAuth.

4.1 Autentizace vs. Autorizace

V následujících podsekcích jsou vysvětleny pojmy autentizace a autorizace. Zároveň jsou uvedeny rozdíly mezi autentizací, federalizovanou autentizací, autorizací a delegovanou autorizací.

4.1.1 Autentizace

Autentizace (angl. *authentication*) je proces ověření identity uživatele - rozpoznat, že uživatel je tím, kým tvrdí, že je. [4]

Ve běžném světě si lze představit situaci, kdy se policista zeptá na identifikaci osoby a ta se mu prokáže například tím, že její podoba na fotce občanského průkazu odpovídá vzhledu osoby. Na počítačích a webu je autentizace ověření, že uživatel sedící u klávesnice je majitelem příslušného účtu. Autentizace je typicky prováděna zeptáním se uživatele na uživatelské jméno a heslo. Uživatelské jméno reprezentuje uživatelovu identitu a softwarové aplikace předpokládají, že pokud uživatel zadal správné heslo, tak je skutečně tím pravým uživatelem. [4]

4.1.2 Federalizovaná autentizace

I když většina aplikací má vlastní systém uživatelských účtů (včetně uživatelských jmen a hesel), některé aplikace spoléhají na jiné služby při ověřování uživatelů. Tomu se říká federalizovaná autentizace (angl. *federated authentication*). V korporátním IT prostředí mohou aplikace při ověřování uživatelů důvěřovat serveru Active Directory, LDAP serveru nebo poskytovateli SAML. Na webu aplikace často důvěřují poskytovatelům OpenID (jako například

Google nebo Yahoo!), aby za ně provedly autentizaci. Federalizování přináší spoustu výhod jak pro aplikace, tak pro vývojáře i koncové uživatele. OpenID je nejběžněji používaným webovým protokolem pro federalizovanou autentizaci. [4]

OpenID bylo na webu používáno po mnoho let, v současné době má svého nástupce v další generační verzi, které se nazývá OpenID Connect a je založená na protokolu OAuth 2.0. [4] Takovým poskytovatelem je například Google [1]. Facebook poskytuje velmi podobný koncept, tzv. Facebook Connect, který je také postavený na protokolu OAuth 2.0, ale obsahuje drobné rozdíly, takže není přesnou implementací OpenID Connect [2].

4.1.3 Autorizace

Autorizace je proces ověření, že uživatel má právo provést určitou akci, jako například přečíst dokument nebo přistoupit do emailového účtu. Typicky nejdříve vyžaduje správnou identifikaci uživatele (autentizaci), aby se poté mohlo zjistit, že je uživatel oprávněný akci provést. [4]

Když policista zastaví auto kvůli překročení rychlosti, nejdříve autentizuje řidiče pomocí řidičského průkazu (aby ověřil identitu) a následně zkontroluje samotný průkaz (datum platnosti, omezení, apod.), zda je řidič oprávněný vozidlo řídit. [4]

Stejný proces probíhá online - webová aplikace nejdříve ověří identitu uživatele tím, že se přihlásí do aplikace, a pak zajistí přístup pouze k takovým datům a službám, ke kterým je uživatel oprávněný, většinou podle kontroly seznamu přístupových práv pro každou operaci. [4]

4.1.4 Delegovaná autorizace

Delegovaná autorizace je povolení (udělení) přístupu jiné osobě nebo aplikaci, aby prováděla akce namísto uživatele. [4]

Když řidič přijede k luxusnímu hotelu, obsluha mu může nabídnout, že za něj bezpečně zaparkuje auto. Řidič tak autorizuje obsluhu k řízení svého vozidla tím, že mu předá klíče od auta a nechá za sebe provést akci zaparkování auta. [4]

OAuth funguje podobně - uživatel udělí přístup aplikaci, aby za něj prováděla pouze určité akce, které uživatel autorizuje. [4]

4.2 SSO - Single Sign-On

V současném digitálním světě musejí uživatelé přistupovat do spousty systémů zajišťujících jejich každodenní byznys aktivity. Jak počet systémů roste, počet přístupových údajů pro každého uživatele také narůstá a tím pádem se zvyšuje i pravděpodobnost jejich ztráty nebo zapomenutí. Jednotné přihlášení

[1] `https://developers.google.com/identity/protocols/OpenIDConnect`
[2] `http://stackoverflow.com/questions/27194838/facebook-login-and-openid-connect`

(angl. *Single Sign-On* - SSO) může být použito k vyřešení mnoha problémů spojených s přílišným množstvím přístupových údajů pro různé aplikace. Přístup SSO k hlavnímu autentizačnímu centru dovoluje získat přístup uživatelům ke všem ostatním dostupným prostředkům. SSO pomáhá vylepšit produktivitu uživatelů i vývojářů tím, že nenutí uživatele si pamatovat několik hesel, a také snižuje čas nutný pro psaní různých hesel při přihlašování. SSO také zjednodušuje administraci správou pouze jedněch přístupových údajů namísto několika. Je také velmi jednoduché spravovat uživatelova oprávnění při nástupu do organizace, změně jeho funkce nebo při odchodu, rychle integrovat další přidané aplikace nebo třeba delegovat přístupová práva během dovolené/prázdnin bez efektu na helpdesk organizace. [26]

Existuje několik typů SSO spadajících pod různé kategorie, založených na tom, kde jsou nasazeny (Intranet, Extranet, Internet), jak jsou nasazeny (architektura - jednoduchá, komplexní), jaké přístupové údaje používají (tokeny, certifikáty, ...) a protokoly, které používají (Kerberos, SAML, OpenID, ...). [26] V tabulce 4.1 lze vidět porovnání nejpoužívanějších technologií SSO podle [20].

	OpenID	**OAuth**	**SAML**
datum vzniku	2005	2006	2001
současná verze	OpenID 2.0	OAuth 2.0	SAML 2.0
hlavní účel	SSO pro spotřebitele	API autorizace mezi aplikacemi	SSO pro enteprise uživatele
protokoly	XRDS, HTTP	JSON, HTTP	SAM, XML, HTTP, SOAP

Tabulka 4.1: Srovnání OpenID, OAuth a SAML. [20]

V článku *SAML, OAuth, OpenID* se uvádí: „Následující body mohou být užitečné při zvažování, kterou z technologií OpenID, OAuth nebo SAML zvolit:[16]“

- Pokud je úkolem vytvořit SSO, kde alespoň jednou ze stran je organizace (angl. *enteprise*), použijte SAML, jinak OpenID.
- Pokud použití zahrnuje mobilní zařízení pro API autorizaci, použijte OAuth.
- Pokud případ použití vyžaduje centralizovaného poskytovatele služeb, použijte SAML.

V případě této diplomové práce platí první a třetí bod, úkolem je totiž vytvořit centralizovanou přihlašovací službu, a školu SZŠ ČB lze považovat za *enteprise* ve smyslu jednotné organizace a domény. Navíc aplikace INUM i Moodle by bylo možné napojit na OpenID i OAuth, není možné ale připojit Google Apps, který nabízí pouze konektor na externího poskytovatele identity

s technologií SAML. Z těchto důvodu se sekce *4.4 SAML - Security Assertion Markup Language* věnuje popisu fungování SAML.

Mezi produkty a služby podporující SAML 2.0, či přímo postavené na SAML, patří mimo jiné: Shibboleth, SimpleSAMLphp, ADFS 2.0, JOSSO, OpenAM a spousta dalších [3].

4.3 Přihlášení pomocí třetí strany

Přihlášení pomocí účtu třetí strany do aplikace INUM představuje jiný případ použití než SSO, přestože v některých případech se mohou překrývat (použité technologie, princip přihlášení apod.). V tomto procesu nevlastní příslušný uživatelský účet třetí strany škola, ale právě konkrétní uživatel, organizace nemá o tomto účtu v průběhu vzniku uživatelské konta v jejím prostředí žádné informace. Uživatel tak musí sám povolit a provést propojení jeho jednotného uživatelského účtu v prostředí školy s jeho účtem v aplikaci třetí strany.

Propojení s externím účtem nemusí přinést pro aplikaci INUM pouze další možnost uživateli se přihlásit. Lze totiž využít i prostředky, které třetí strana používá. V současné době je trendem používat tzv. Social Sign-On (přihlášení k sociálním sítím).

Sociální sítě dovolují uživatelům se spojovat s rodinou, kamarády či spolupracovníky. Uživatelé také budují své profily, kde uchovávají a sdílejí s ostatními různé druhy obsahů, včetně fotek, videí a zpráv. Aktualizace profilu zajímavým obsahem je formou sebevyjádření, které zvyšuje interakci v těchto webových stránkách. Aby sociální sítě podpořily tuto interakci a poskytly bohatší obsah, vystavují svou síť webovým službám ve formě online API. API dovolují třetím stranám komunikaci se sociálními sítěmi, přistupovat k informacím a mediím sdílených uživateli a budovat sociální aplikace sdružující, zpracovávající a vytvářející obsah založený na uživatelových zájmech. OAuth 2.0 je jednou z aplikací pomáhajících překonat problémy v tradičním modelu klient-server. [19]

Z důvodu usnadnění sdílení uživatelského obsahu pro webové stránky v bezpečné formě, OAuth 2.0 protokol umožňuje uživatelům udělit přístup aplikacím třetí strany k webovým prostředkům bez sdílení jejich přístupových údajů nebo přístupu ke všem jejich datům. OAuth podporuje různé případy použití jako webové stránky, aplikace založené na agentovi, nativní mobilní aplikace, desktopové aplikace a aplikace na dalších zařízeních. Pro případ kdy je autorizován přístup k prostředku uživatelovy identity skrz webové stránky třetích stran, OAuth může být použit jako webové SSO schéma. To znamená, že například prostředky umístěné na stránkách (třeba Facebook) hrají roli poskytovatele identity (IdP), který udržuje informace o identitě uživatele a autentizuje ho, zatímco webová stránka třetí strany (v této diplomové práci aplikace INUM) vystupuje jako relying party (RP) spoléhající na autentizaci a autorizaci identity. [30]

[3] `https://en.wikipedia.org/wiki/SAML-based_products_and_services`

Detailní popis protokolu OAuth 2.0 je umístěn v sekci *4.5 OAuth*. Mezi nejznámější služby poskytující OAuth 2.0 protokol patří: Facebook, Google, Twitter, LinkedIn, Reddit, Paypal a jiné [4].

4.4 SAML - Security Assertion Markup Language

Security Assertion Markup Language (SAML) poskytuje zabezpečné řešení založené na XML pro výměnu bezpečnostních informací o uživateli mezi poskytovatelem identit (angl. *Identity Provider* - IdP), tedy organizací, a poskytovatelem služby (angl. *service provider*). Standard SAML definuje pravidla a syntaxi pro výměnu dat, zároveň je flexibilní a umožňuje přenos dalších specifických dat na externího poskytovatele služby. [22]

SAML transakce se účastní tři role - asserting party, relying party a předmět (klient, webový prohlížeč, angl. *subject*). Asserting party (poskytovatel identity) je autoritní systém podávající uživatelské informace. Relying party (poskytovatel služby) je systém, který věří informacím od asserting party a používá poskytnutá data zajištění aplikace uživatelovi. Uživateli a jeho identitě, jež jsou součástí transakce, se říká předmět. [22]

Části utvářející standard SAML jsou aserce (angl. *assertions*), protokoly (angl. *protocols*), vazby (angl. *bindings*) a profily (angl. *profiles*). Každá vrstva standardu může být přizpůsobena, tak aby odpovídala byznys požadavkům organizace. Protože scénář každé organizace může být unikátní, implementace těchto byznys případů by měla být personalizována jak pro službu, tak pro poskytovatele identity. [22]

Transakce z assertion party na relying party se nazývá SAML aserce. Relying party považuje všechna data obsažená v aserci za validní. Struktura SAML aserce je definována XML schématem a obsahuje hlavičku, předmět a prohlášení (angl. *statements*) o předmětu ve formě atributů a podmínek. Aserce může také obsahovat autorizační prohlášení, která říkají, co vše může uživatel provádět za akce ve webové aplikaci. [22]

SAML standard definuje protokoly pro požadavky (angl. *requests*) a odpovědi (angl. *responses*) používané pro komunikaci skrze aserce mezi poskytovatelem služby a poskytovatelem identity. Zde jsou příklady takových protokolů citelewis2009:

- **Authentication Request Protocol**. Definuje jak poskytovatel služeb může požadovat aserce obsahující autentizaci či další atributy.
- **Single Logout Protocol**. Definuje mechanismus pro odhlášení se ze všech poskytovatelů služeb.
- **Artifact Resolution Protocol**. Definuje jak jsou počáteční artefakty a poté hodnoty požadavků/odpovědí předávány mezi poskytovatelem služby a poskytovatelem identity.

[4] `https://en.wikipedia.org/wiki/List_of_OAuth_providers`

- **Name Identifier Management Protocol**. Definuje jak přidat, změnit nebo odebrat hodnotu identifikátoru názvu (jména) pro poskytovatele služby.

SAML vazby mapují SAML protokoly na standardní protokoly nižší vrstvy síťové komunikace používané pro přenos SAML asercí mezi poskytovatelem identity a poskytovatelem služby. Mezi takové vazby patří např.:

- **HTTP Redirect Binding**. Používá HTTP přesměrování pro zasílání zpráv.
- **HTTP POST Binding**. Definuje jak mohou být aserce přenášeny použitím obsahu zakódovaném v Base64.
- **HTTP Artifact Binding**. Definuje jak je artefakt přenášen na přijímač použitím HTTP.
- **SOAP HTTP Binding**. Používá SOAP 1.1 zprávy a SOAP přes HTTP.

Nejvyšší úroveň SAML komponent jsou profily, neboli byznys použití mezi poskytovatelem služby a poskytovatelem identity, které diktují, jak budou aserce, protokoly a vazby fungovat dohromady k poskytnutí SSO. Některé z profilů jsou uvedeny v následujícím seznamu [22]:

- **Web Browser SSO Profile**. Používá Authentication Request Protocol a některou z vazeb: HTTP Redirect, HTTP POST nebo HTTP Artifact.
- **Single Logout Profile**. Používá Single Logout Protocol, který umí odhlásit uživatele ze všech přihlášených služeb přes funkci jednotného odhlášení.
- **Artifact Resolution Profile**. Používá Artifact Resolution Protocol přes SOAP HTTP vazbu.
- **Name Identifier Management Profile**. Používá Name Identifier Management Protocol a může být použit s HTTP Redirect, HTTP POST, HTTP Artifact nebo SOAP.

Web Browser SSO Profile může být iniciován ze strany poskytovatele identity nebo ze strany poskytovatele služby. Pokud je iniciován poskytovatelem identity, aserce je buď digitálně podepsána, zašifrována nebo obojí. V tomto profilu jsou všechny informace v aserci poslány najednou poskytovateli služby použitím některé z HTTP vazeb a protokolů. Poskytovatel služby dešifruje (pokud je nutné) a zkontroluje integritu zprávy proti podpisu. Dále pak rozebere XML zprávu a shromáždí zaslané atributy, až nakonec provede SSO pomocí služby pro zpracování aserce (angl. *Assertion Consumer Service*).[22]

Když uživatel přistupuje na externí webovou stránku bez projití interním federačním poskytovatelem identity, poskytovatel služby musí zaslat SAML požadavek na poskytovatele identity na místo uživatele. Tento proces se

nazývá iniciace SSO poskytovatelem služby. V takovém případě uživatel přijde na stránku specifickou pro aplikaci, bez SAML aserce. Poskytovatel služby přesměruje uživatele na stránku poskytovatele federační identity se SAML požadavkem (ukázka požadavku v kódu 3), volitelně s query parametrem `RelayState` říkajícím, kterou SAML entitu použít při posílání aserce zpět poskytovateli služby.[22]

Po přijetí požadavku od poskytovatele služby poskytovatel identity zpracuje SAML požadavek stejně, jako kdyby přišel interně (viz předchozí případ). Tento způsob použití je důležitý, protože dovoluje uživatelům si například uložit záložku ve webovém prohlížeči s externí stránkou poskytovatele služby a stejně se dosáhne SSO pomocí přesměrování v prohlížeči.[22]

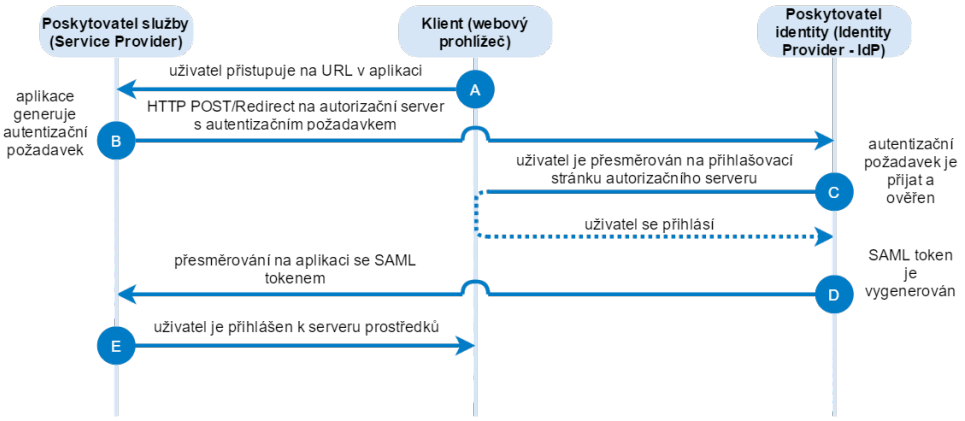
Příklad SAML XML odpovědi poskytovatele identity poskytovateli služeb lze vidět v příloze A.

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:AuthnRequest
xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
AssertionConsumerServiceURL="https://mujucet.szscb.cz/saml/SSO"
Destination="https://idp.szscb.cz/adfs/ls/"
ForceAuthn="false"
ID="id..."
IsPassive="false"
IssueInstant="2016-04-21T12:47:33.390Z"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Version="2.0">
    <saml2:Issuer
    xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
        https://mujucet.szscb.cz/saml/metadata
    </saml2:Issuer>
    <saml2p:RequestedAuthnContext
    Comparison="exact">
        <saml2:AuthnContextClassRef
        xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
            urn:oasis:names:tc:SAML:2.0:ac:classes:Password
        </saml2:AuthnContextClassRef>
    </saml2p:RequestedAuthnContext>
</saml2p:AuthnRequest>
```

Ukázka kódu 3: Příklad SAML XML požadavku na IdP od aplikace INUM.

Na obrázku 4.1 je vidět jeden možný scénář zobrazující průběh SAML autentizace, který bude používaný i při implementaci SSO řešení na škole SZŠ ČB. Scénář odpovídá iniciování SSO ze strany poskytovatele služby, v tomto případě aplikace INUM. V následujícím seznamu jsou detailně popsány jednotlivé body průběhu:

- **A**. Uživatel otevře webový prohlížeč a přijde na WWW stránky `https://mujucet.szscb.cz` (INUM). Chce se přihlásit tlačítkem *Přihlásit SSO*.
- **B**. Aby aplikace INUM autentizovala uživatele pomocí SSO, vygeneruje SAML autentizační požadavek (angl. *SAML Authnrequest*), digitálně ho



Obrázek 4.1: Diagram běžného průběhu autentizace pomocí SAML. (inspirováno z [8])

podepíše, dle volby ho může i zašifrovat (angl. *encrypt*), a zakóduje ho (angl. *encode*). Následně přesměruje uživatelův prohlížeč na poskytovatele identity (angl. *Identity Provider - IdP*), kde se bude uživatel autentizovat. IdP přijme požadavek, dekóduje ho (angl. *decode*), rozšifruje (angl. *decrypt*) pokud je nutné, a ověří digitální podpis.

- **C**. Pokud je autentizační požadavek validní, IdP zobrazí uživateli přihlašovací formulář, kde může zadat uživatelské jméno a heslo. Uživatel může být již přihlášen, v takovém případě tento krok odpadá.

- **D**. Jakmile je uživatel přihlášen, IdP vygeneruje SAML token obsahující informace o identitě uživatele (uživatelské jméno, email, organizace apod.). IdP přesměruje uživatelům prohlížeč zpět na poskytovatele služby (INUM) se SAML tokenem.

- **E**. Poskytovatel služby ověří SAML token, rozšifruje pokud je nutné, a vytáhne z něj informace o identitě uživatele - o koho se jedná a jaká jsou jeho oprávnění. INUM přihlásí uživatele do svého systému, většinou za pomoci cookie a session.

Na konci procesu je uživatel přihlášen do aplikace poskytovatele služby a může s aplikací pracovat jako běžně ověřený uživatel přes přihlašovací formulář. Uživatelovy přístupové údaje ale nikdy neprošly přes aplikaci INUM, pouze přes IdP.

Jak poskytovatel identity, tak poskytovatel služeb vystavují svá SAML metadata. Metadata mohou být použita ke specifikování sdílení informací o konfiguraci mezi dvěma komunikujícími entitami. Například tato data mohou obsahovat podporu entity pro určité typy SAML vazeb, informace o identifikátorech či třeba infrastrukturu veřejného klíče (angl. *Public Key Infrastrucute - PKI*). [5] Příklad metadat poskytovatele služeb, v tomto případě aplikace INUM, lze vidět v kódu 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor
xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
ID="https___mujucet.szscb.cz_saml_metadata"
entityID="https://mujucet.szscb.cz/saml/metadata">
    <md:SPSSODescriptor AuthnRequestsSigned="true"
    WantAssertionsSigned="true"
    protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <md:KeyDescriptor use="signing">
            <ds:KeyInfo
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                    <ds:X509Certificate>
                        certifikat...
                    </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:KeyDescriptor use="encryption">
            <ds:KeyInfo
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>
                    <ds:X509Certificate>
                        certifikat...
                    </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:SingleLogoutService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
        Location="https://mujucet.szscb.cz/saml/SingleLogout"/>
        <!-- pokracuji dalsi tagy md:SingleLogoutService -->
        <md:NameIDFormat>
            urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
        </md:NameIDFormat>
        <!-- pokracuji dalsi tagy md:NameIDFormat -->
        <md:AssertionConsumerService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
        Location="https://mujucet.szscb.cz/saml/SSO" index="0"
        isDefault="true"/>
        <!-- pokracuji dalsi tagy md:AssertionConsumerService -->
    </md:SPSSODescriptor>
</md:EntityDescriptor>
```

Ukázka kódu 4: Příklad SAML XML metadat aplikace INUM jako poskytovatele služby.

4.5 OAuth

V tradičním autentizačním modelu klient-server klient požaduje přístup ke chráněným prostředkům na serveru autentizací vůči serveru použitím přístupových údajů vlastníka prostředků. Aby se mohl poskytnout přístup aplikacím třetích stran k těmto chráněným prostředkům, vlastník prostředků sdílí své přístupové údaje s těmito třetími stranami. To přináší několik problémů

a omezení:[19]

- Aplikace třetích stran jsou nuceny ukládat přístupové údaje vlastníka prostředků pro budoucí využití, typicky heslo v čitelné formě.[19]
- Servery musí podporovat přihlašování heslem navzdory bezpečnostním slabinám obsaženým v tomto stylu přihlašování.[19]
- Aplikace třetích stran získají neomezený přístup ke chráněným prostředkům, bez možnosti vlastníka prostředků omezit dobu přístupu nebo limitovat přístup jen na část těchto prostředků.[19]
- Vlastníci prostředků nemohou odebrat přístup individuálně třetí straně bez odebrání přístupu všem třetím stranám a nemají jinou možnost, jak to provést, než změnit přístupové heslo. Kompromitování jakékoliv aplikace třetí strany má za výsledek odhalení hesla koncového uživatele a přístup ke všem datům chráněným tímto heslem. [19]

OAuth 2.0 (Open Authentication) protokol adresuje tyto problémy zavedením autorizační vrstvy a oddělením rolí klienta od vlastníka prostředků. V OAuth požaduje klient (angl. *client*) přístup k prostředkům (angl. *resources*) kontrolovaným vlastníkem prostředků (angl. *resource owner*), umístěným na serveru prostředků (angl. *resource server*), a jsou mu poskytnuta jiná pověření (angl. *credentials*) než vlastníkovi prostředků. Namísto použití pověření vlastníka prostředků k přístupu ke chráněným prostředkům, klient obdrží přístupový token - řetězec znaků označující specifický rámec (angl. *scope*), dobu životnosti a jiné přístupové atributy. Přístupové tokeny jsou vyžádány klienty třetích stran u autorizačního serveru se souhlasem vlastníka prostředků. Klient následně používá přístupové tokeny k přístupu ke chráněným prostředkům umístěným na serveru prostředků.[19]

Například, koncový uživatel (vlastník prostředků) může udělit přístup tiskové službě (klient) k jeho chráněným fotkám uloženým na službě se sdílením fotek (server prostředků), a to bez sdílení jeho uživatelského jména či hesla s tiskovou službou. Namísto toho se uživatel autentizuje přímo vůči serveru (autorizační server), kterému důvěřuje tisková služba, jež poskytne tiskové službě specifické přístupové údaje (přístupový token).[19]

Role v OAuth 2.0

OAuth protokol definuje 4 role:

1. **Vlastník prostředků** (angl. *Resource owner*). Entita schopná povolit přístup ke chráněnému prostředku. Když je vlastník prostředků osoba, odkazuje se na ni jako na koncového uživatele (angl. *end-user*).
2. **Server prostředků** (angl. *Resource server*). Server obsahující chráněné prostředky, schopný přijímat a odpovídat na žádosti k přístupu ke chráněným prostředkům skrz přístupové tokeny.

3. **Klient** (angl. *Client*). Aplikace provádějící požadavky na chráněné prostředky jménem vlastníka prostředků a s jeho svolením (autorizací).

4. **Autorizační server** (angl. *Authorization server*. Server poskytující přístupové tokeny klientovi poté, co se úspěšně vlastník prostředků autentizuje

Abstraktní diagram průběhu autorizace pomocí protokolu OAuth znázorněný na obrázku 4.2 popisuje interakci mezi 4 rolemi v OAuth:

1. Klient požaduje autorizaci (angl. *authorization request*) od vlastníka prostředků. Autorizační požadavek může být adresován přímo na vlastníka prostředků (jako na obrázku 4.2) nebo přednostně nepřímo přes autorizační server jako zprostředkovatele.[19]

2. Klient obdrží autorizační povolení (angl. *authorization grant*), což je pověření reprezentující autorizaci vlastníka prostředků, vyjádřeného pomocí jednoho ze 4 typů povolení definovaných ve specifikaci. Typ povolení závisí na metodě použité klientem k požadavku na autorizaci a také na typech, které podporuje autorizační server.[19]

3. Klient požaduje přístupový token (angl. *access token*) autentizací vůči autorizačnímu serveru a poskytnutím autorizačního povolení.[19]

4. Autorizační server autentizuje klienta a validuje autorizační povolení, pokud je vše v pořádku, vystaví přístupový token.[19]

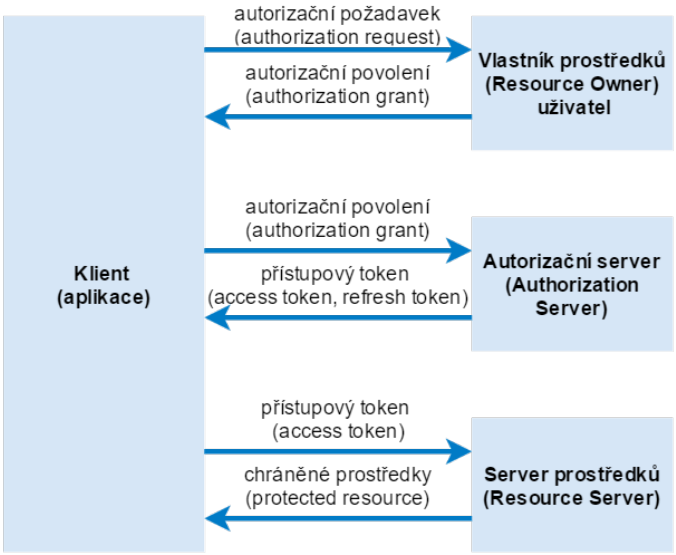
5. Klient požaduje chráněné prostředky ze serveru prostředků a autentizuje se pomocí přístupového tokenu.[19]

6. Server prostředků validuje přístupový token a pokud je v pořádku, zpracovává požadavek od klienta (například mu poskytne chráněné prostředky).[19]

4.5.1 Autorizační povolení

Autorizační povolení (angl. *authorization grant*) jsou povolení, pověření (angl. *credentials*) reprezentující autorizaci vlastníka prostředků (povolení přístupu k jeho chráněným prostředkům), používané klientem k získání přístupového tokenu. V OAuth 2.0 existují 4 typy autorizačních povolení:

- **Autorizační kód** (angl. *authorization code*). Autorizační kód je získán za pomoci autorizačního serveru jako prostředníka mezi klientem a vlastníkem prostředků. Namísto požadavku na autorizaci přímo vlastníka prostředků, klient směruje vlastníka prostředků na autorizační server, který následně směruje vlastníka prostředků zpět na klienta s autorizačním kódem. Před směrováním zpět na klienta autorizační server autentizuje vlastníka prostředků a získá od něj autorizaci. Protože se vlastník prostředků ověřuje pouze vůči autorizačnímu serveru, jeho přihlašovací údaje nejsou nikdy sdílena s klientem.[19]



Obrázek 4.2: Abstraktní diagram průběhu autorizace pomocí OAuth. (inspirováno z [19])

- **Implicitní** (angl. *implicit*). Implicitní povolení je zjednodušenou verzí autorizačního kódu, optimalizované pro klienty implementované ve webovém prohlížeči, například v jazyku JavaScript. Namísto získání autorizačního kódu, klient získá přístupový token přímo (jako výsledek autorizace vlastníka prostředků). Typ povolení je implicitní, protože nejsou získány žádné zprostředkující údaje (jako právě autorizační kód).[19]

- **Pověření vlastníka prostředků heslem** (angl. *resource owner password credentials*). Přihlašovací údaje vlastníka (uživatelské jméno a heslo) jsou použity přímo jako autorizační povolení k získání přístupového tokenu. Tento typ povolení by měl být použit pouze v případě, kdy je vysoká důvěra mezi vlastníkem prostředků a klientem (například je klient součástí operačního systému zařízení nebo privilegovaná aplikace) a pokud nejsou dostupné žádné jiné typy povolení.[19]

- **Přístupové údaje klienta** (angl. *client credentials*). Přístupové údaje klienta (či jiné formy autentizace klienta) mohou být použity jako autorizační povolení, když je autorizační scope (rozsah, rámec) pod kontrolou klienta limitován na chráněné prostředky nebo chráněné prostředky byly již sjednány s autorizačním serverem. Přístupové údaje klienta jsou používány jako autorizační povolení typicky, když klient jedná sám za sebe (klient je zároveň vlastník prostředků) nebo požaduje přístup ke chráněným prostředkům na základě již dříve povolené autorizace od autorizačního serveru.[19]

4.5.2 Přístupový token

Přístupový token (angl. *access token*) je identifikační údaj používaný k přístupu ke chráněným prostředkům. Token je řetězec znaků poskytnutý klientovi a reprezentující autorizaci. Řetězec je obvykle pro klienta nečitelný (pouze řetězec zdánlivě nic neříkajících znaků). Tokeny představují specifické rozsahy (angl. *scopes*) a na jak dlouho je přístup povolen, uděluje je vlastník prostředků, a jsou vyžadovány serverem prostředků i autorizačním serverem. Token může označovat identifikátor používaný k získání autorizačních informací nebo sám obsahuje informace v ověřitelné formě (tj. řetězec obsahující data a digitální podpis). Přístupový token poskytuje abstraktní vrstvu a nahrazuje různé autorizační konstrukce (např. uživatelské jméno a heslo) jediným tokenem, kterému rozumí server prostředků. Tato abstrakce dovoluje získávat přístupové tokeny ve více omezující míře, než tomu bylo zvykem u autorizačních povolení, a zároveň odstraňuje nutnost serveru požadavků rozumět velkému množství autentizačních metod. Přístupové tokeny mohou mít různé formáty, struktury a metody využití (třeba kryptografické prvky), odpovídající bezpečnostním požadavkům serveru prostředků.[19]

4.5.3 Obnovující token

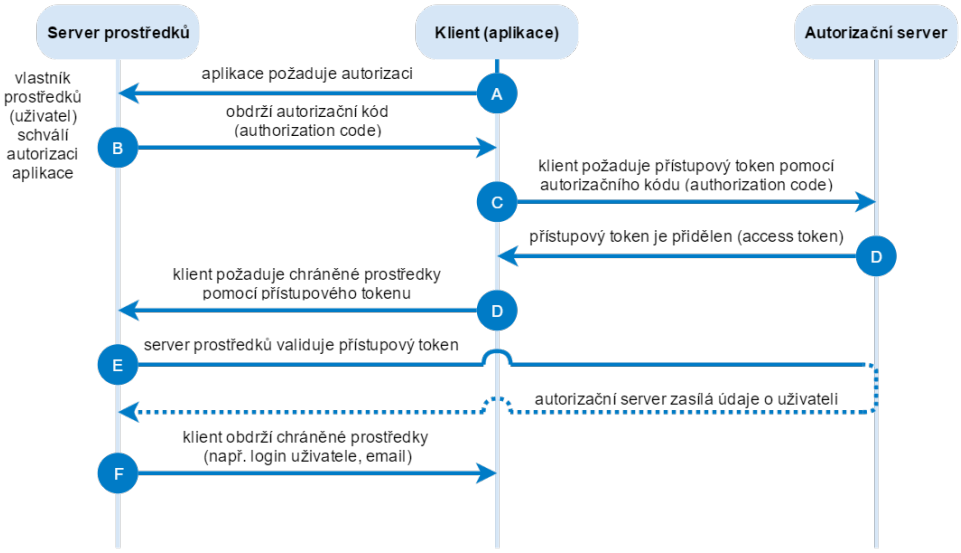
Obnovující tokeny (angl. *refresh tokens*) jsou identifikační údaje používané k získání přístupových tokenů. Tokeny jsou poskytnuty klientovi autorizačním serverem a jsou využity k získání nového přístupového tokenu, když se současný přístupový token stane nevalidním nebo vyprší doba jeho platnosti, nebo k získání dalších přístupových tokenů se stejným či užším rozsahem (přístupové tokeny mohou mít kratší životnost a méně oprávnění, než bylo povoleno od vlastníka prostředků). Získání obnovujícího tokenu je nepovinné a závisí na uvážení autorizačního serveru. Když autorizační server obnovující tokeny poskytuje, je součástí vystavení přístupového tokenu. Obnovující token je řetězec znaků reprezentující autorizační povolení udělené klientovi vlastníkem prostředků. Token představuje identifikátor používaný k získání autorizačních informací. Na rozdíl od přístupových tokenů jsou obnovující tokeny zamýšleny pro použití pouze s autorizačním serverem, nejsou nikdy posílány serveru prostředků.[19]

4.5.4 Přihlášení pomocí OAuth

Autor publikace *Getting Started with OAuth 2.0* Boyd uvádí, že nejvhodnější použití OAuth protokolu pro webové aplikace je tzv. *Server-Side Web Application Flow*, jindy nazývaný jako *Authorization Code Flow*. Tento přístup se používá v případě, že je vyžadován dlouhodobý a pravidelný přístup ke chráněným prostředkům nebo když OAuth klient je webová aplikace. [4] Obě tyto podmínky pro implementaci přihlášení přes účet třetí strany v aplikaci INUM platí.

V následujícím seznamu je popsán konkrétní způsob ověření uživatele

pomocí protokolu OAuth 2.0 odpovídající obrázku 4.3, který bude použit při implementaci přihlášení uživatele pomocí účtu třetí strany (Google, Facebook) podporující protokol OAuth 2.0:



Obrázek 4.3: Diagram Server-Side Web Application Flow autorizace pomocí OAuth. (inspirováno z [8])

- **A**. Uživatel se pomocí webového prohlížeče jakýmkoliv způsobem přihlásí do svého účtu v aplikaci INUM. Následně chce povolit přihlašování do svého účtu pomocí aplikace třetí strany, například Facebook. Klikne na tlačítko propojit a je přesměrován na autorizační server s požadavkem na autorizace aplikace (s definovaným rozsahem - scope). Uživateli je zobrazen přihlašovací formulář, pokud ještě není přihlášen, s hláškou, zda schvaluje aplikaci přistupovat k jejich účtu.
- **B**. Aplikace obdrží autorizační kód a uživatel je přesměrován zpět na aplikaci.
- **C**. Aplikace vyžaduje přístupový token po autorizačním serveru. V požadavku je obsažen autorizační kód získaný z předchozí komunikace.
- **D**. Pokud je autorizační kód validní, autorizační server zašle klientovi (aplikaci) přístupový token. Součástí odpovědi může být i obnovující (refresh) token.
- **D**. Po určité době (ale může být i okamžitě, proto stejné označení písmenem D) vyžaduje aplikace přístup ke chráněným prostředkům na serveru prostředků. V případě autentizace je chráněným prostředkem ověření, že uživatele může aplikace přihlásit. Aplikace tedy zašle přístupový token na server prostředků v rámci požadavku na chráněné prostředky.
- **E**. Server prostředků validuje přístupový token. Server prostředků může být totožný s autorizačním serverem.

- **F**. Pokud byla validace přístupového tokenu v pořádku, server prostředků vrátí požadované chráněné prostředky aplikaci, která následně přihlásí uživatele. Může tak provést například párováním emailu vráceném v odpovědi serveru prostředků s emailem uvedeným v databázi uživatelů aplikace.

Kapitola 5
Návrh

5.1 Původní stav integrace systémů

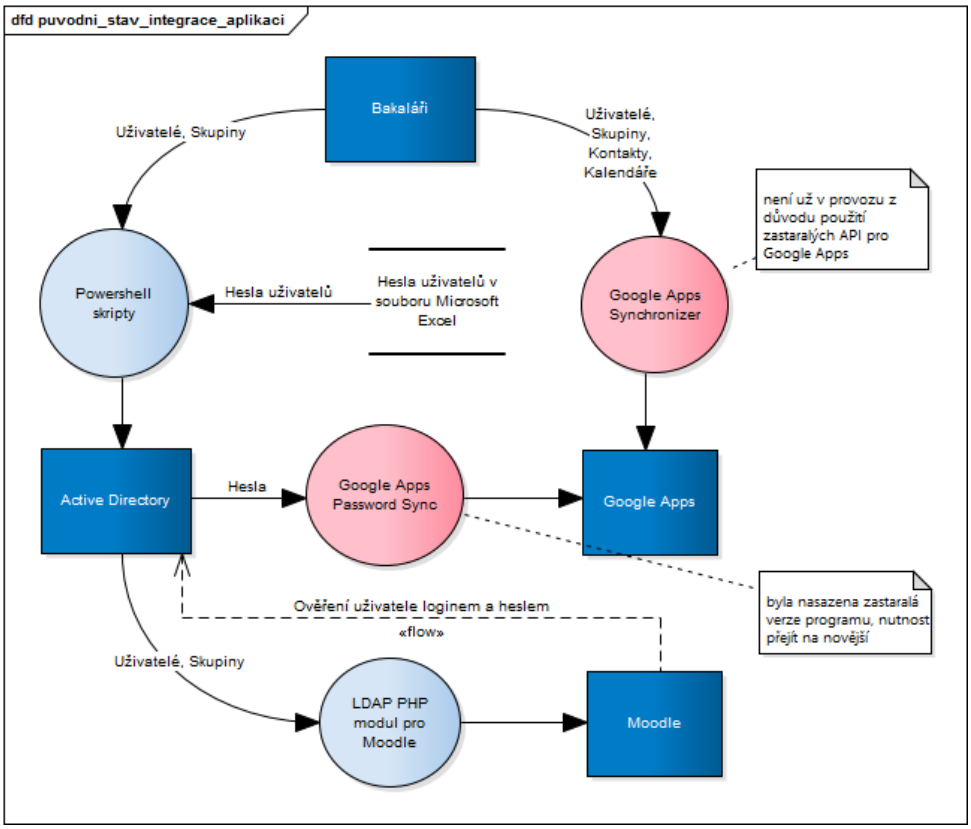
Před nasazením aplikace INUM se o valnou většinu synchronizace staraly skripty napsané v jazyku Powershell. V prostředí operačního systému Windows se jedná asi o nejlepší volbu, jak systém administrovat. Skripty jsou ale hůře spravovatelné a méně ohebné. Synchronizace bývá většinou jednoduchá a slouží právě jednomu účelu. Nevýhodou skriptů je však jejich neohebnost a malá možnost, jak do nich zapojit uživatele - třeba změna hesla, změna emailu apod. Zároveň do skriptů může zasahovat pouze osoba znalá samotného jazyka Powershell a konkrétního prostředí. INUM na webovém rozhraní dokáže administrovat i méně zkušený uživatel, určitá znalost prostředí je však vyžadována i zde (nikoliv však znalost programovacího jazyka).

Na obrázku 5.1 je zobrazen diagram toku dat v původním stavu komunikace aplikací. Jednotlivé části budou rozebrány v následujících podsekcích.

5.1.1 Bakaláři a Active Directory

Uživatelské účty a skupiny uživatelů v doméně Active Directory vznikaly a zanikaly pomocí Powershell skriptů, které přistupovaly přímo do databáze aplikace Bakaláři a přenášely informace do adresáře. Neexistoval mezikrok, který by data transformoval nebo který by rozhodl, které účty mají už být přenášeny a které ne - cokoliv se objevilo v databázi Bakalářů, ihned se přeneslo do domény ve stejné podobě. To s sebou přináší problémy jako vznik účtů v doméně, které by správně měly existovat zatím pouze v Bakalářích. Například se jedná o nově přijaté žáky a studenty, kteří musejí být evidováni v Bakalářích kvůli přijímacímu řízení, ale z hlediska domény AD ještě nejsou plnohodnotnými uživateli, kteří potřebují doménový přístup.

Dalším velkým problémem tohoto způsobu synchronizace jsou uživatelské přihlašovací údaje - login a heslo. Přístup uživatele k účtu v aplikaci Bakaláři probíhá buď přes webové rozhraní (přístup žáků a studentů ke své žákovské knížce a přístup učitelů k administraci docházky, známek aj.) nebo přes desktopovou aplikaci Bakalářů, která slouží převážně pro administraci celého systému Bakalářů. V určitý moment administrátor Bakalářů v desktopové



Obrázek 5.1: Původní stav integrace aplikací.

aplikaci vygeneruje přístupové údaje buď jednotlivému účtu, nebo hromadně třeba celé třídě žáků. Toto je jediný okamžik, kdy lze získat heslo ve strojově čitelné podobě, konkrétně jako export hesel do souboru aplikace Microsoft Excel, lze ho ovšem provést pouze ručně, nikoliv strojově. Dále je pak už heslo uložené v databázi Bakalářů pomocí jednosměrné hashovací funkce, nelze ho tak získat z databáze zpět v textové formě. Zajisté je to správný a bezpečný způsob, jak uživatelská hesla uchovávat. Nevýhoda je ale uzavřenost vůči ostatním aplikacím, protože Bakaláři nenabízejí žádný oficiální způsob, kterým by šlo heslo změnit externě pomocí API. Heslo si ale uživatel může měnit, pouze ručně na webovém rozhraní a opět ho nelze strojově získat.

Login, neboli přihlašovací jméno, může administrátor Bakalářů buď vyplnit ručně nebo nechat generovat pomocí přednastavených šablon v Bakalářích. Opět není jiný způsob, jak informace získat z Bakalářů, než přímý přístup do databáze.

Powershell skripty z výše uvedených důvodů nebyly schopny číst hesla uživatelů z Bakalářů, nebylo tak možné dosáhnout jednotného hesla pro všechny aplikace. Tento nedostatek byl částečně nahrazen postupem, kdy administrátor v momentě generování hesel pro uživatele vyexportoval soubor ve formátu Microsoft Excel a uložil ho do předem daného adresáře na serveru. Synchronizační skripty tuto složku kontrolovaly na přítomnost daného souboru a pokud byl přítomen, heslo uživatele se podle loginu uživatele spárovalo

s uživatelským účtem v doméně AD. Pokud si však uživatel změnil heslo v doméně nebo v Bakalářích, druhý systém se o změně nedozvěděl a byla tak porušena zamýšlená integrita hesla.

Přihlašovací jméno do Bakalářů a AD bylo totožné. Vazba mezi uživatelskými konty v Bakalářích a v Active Directory probíhala přes unikátní klíč INTERN_KOD uživatele v Bakalářích (v AD byl uložen v atributu employeeId).

5.1.2 Active Directory a Moodle

Životní cyklus uživatelských účtů v Moodle je řízen pomocí synchronizačních PHP skriptů, jak je popsáno v podsekci *2.1.3 Moodle*. Moodle je závislý na datech v AD, tzn. i na přihlašovacím jménu a heslu. Pokud se jedno z nich v AD změní, uživatel ihned přistupuje do Moodle změněnými údaji. Na straně Moodle nelze měnit žádné uživatelské údaje, všechny atributy se berou jako výchozí v Active Directory.

V tomto ohledu neproběhne v rámci imlementace aplikace INUM žádná změna.

5.1.3 Bakaláři a Google Apps

Jak je zmíněno v podsekci *2.1.4 Google Apps*, na škole SZŠ ČB byl implementován nástroj čistě pro přesun dat z Bakalářů do Google Apps v rámci bakalářské práce „Integrace Google Apps“ Luboše Palíška. Nástroj vytvářel a zakazoval uživatelské účty v doméně GA, vytvářel Google skupiny uživatelů na základě tříd a skupin tříd, vytvářel sdílené kontakty domény a přenášel rozvrh hodin jednotlivých tříd.

Používal však stará API, která postupně společnost Google vypínala a nutila programátory k přechodu na nová API. Zároveň čelil stejnému problému se synchronizací hesel jako v případě Active Directory, který se částečně podařilo vyřešit nasazením malé aplikace přímo od společnosti Google, tzv. Google Apps Password Sync [1], který je schopný odchytit událost změny hesla v AD a tuto změnu propagovat do GA. Tento stav je ale limitován nemožností změny hesla v Bakalářích, takže když se změní heslo v Bakalářích, nedostane se ani do Active Directory, ani do Google Apps. Když se změní heslo v AD, tak se sice změní v GA, ale již se nepropaguje do Bakalářů. Opět nastává porušení integrity dat.

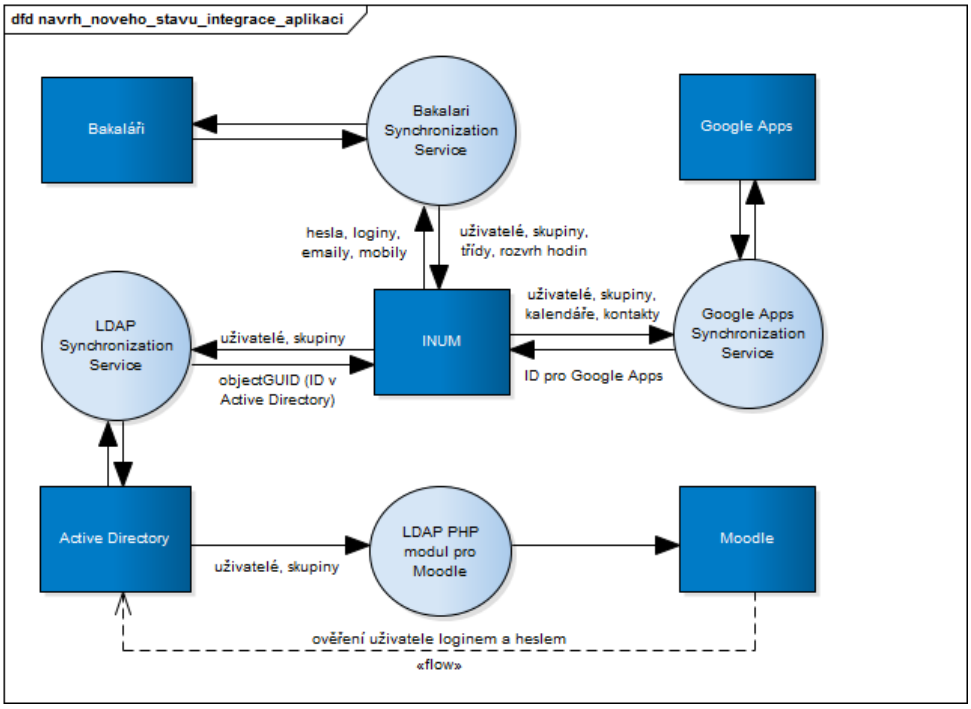
Nástroj musel být z důvodů zastarání API vypnut. V úvahu připadala varianta, že by se pouze aplikace přeprogramovala na nová API a opět spustila. Varianta byla ale zamítnuta, protože aplikace nebyla instalována přímo v prostředí školy SZŠ ČB, ale na serveru třetí strany, a k tomu nebyla zabezpečena proti neoprávněnému přístupu, kdokoliv znalý URL by mohl nakládat s citlivými údaji a procesy organizace. Byl proto zvolen postup přímé integrace aplikace INUM s Google Apps. Tím se eliminují i případné

[1] `https://support.google.com/a/answer/2611859?hl=en&ref_topic=4497963`

vyšší nároky na hardware a konkurování si dvou souběžně puštěných aplikací, které mají ve své podstatě velmi podobný cíl.

5.2 Návrh nového stavu integrace systémů

Na schématickém obrázku 5.2 lze vidět návrh integrace systémů po nasazení aplikace INUM. Je z něj patrné, že aplikace INUM se stane středobodem celé integrace, podobně jako je tomu v integrační architektuře Hub-and-Spoke (viz sekce *3.2.2 Hub-and-Spoke řešení*).



Obrázek 5.2: Návrh nového stavu integrace aplikací po nasazení aplikace INUM.

Pro každý z integrovaných systémů bude mít INUM službu starající se o synchronizaci dat právě s konkrétní aplikací. Protože INUM přímo komunikuje s Bakaláři (primární zdroj dat na škole), Active Directory a Google Apps, bude mít tedy 3 takové služby. Cílem je najít v každém systému unikátní identifikátor pro běžně synchronizované entity (uživatelé, skupiny uživatelů) a pro následnou synchronizaci entit mezi aplikacemi používat výhradně tento identifikátor. Aplikace INUM bude iniciátorem těchto výměn, to znamená, že bude spouštět procesy synchronizace sama plánovačem nebo na vyžádání od administrátora.

5.2.1 Synchronizační služba pro aplikaci Bakaláři

Vzhledem k absenci API v aplikaci Bakaláři, jak bylo popsáno v kapitole *2 Analýza integrovaných systémů a jejich rozhraní*, je nutné přistupovat přímo do

interní databáze. V programovacím jazyku Java existuje elegantní řešení pro připojení se k relačnímu databázovému datovému zdroji, a to API nazývané JDBC (Java Database Connectivity), respektive jeho obálka ve frameworku Spring - JdbcTemplate. JDBC umí komunikovat i s databází MSSQL. Po úspěšném připojení pak zbývá už jen konstruovat SQL (Structured Query Language) dotazy pro získání nebo aktualizaci entit.

Bakaláři jsou primárním zdrojem dat celé integrace, tzn. jakákoliv změna v aplikaci proběhne, včetně výskytu nového záznamu, je žádoucí, aby se vše promítlo do návazných aplikací. Primární zdrojem je především pro data o žácích (tabulka `zaci`, primární klíč `INTERN_KOD`), studentech (stejná tabulka jako u žáků) a učitelích (tabulka `ucitele`, primární klíč `INTERN_KOD`). Pokud má osoba přístup k webovému rozhraní aplikace Bakaláři, má záznam v tabulce `webuser`, kde je primárním klíčem sloupec `ID`. Záznamů v tabulce `webuser` může být několik - například separátní přístup pro samotné žáky a rodiče - je tak důležité měnit pouze správný odpovídající záznam. Tyto atributy budou mapovány na atributy entity `Person` aplikace INUM (viz obrázek 5.3)- primární klíč uživatele na atribut `bakalariId`, přístup k webovému rozhraní na atribut `bakalariWebuserId`. Dle typu uživatele (žák, student, učitel) se rozhodne o atributu `type`.

Velmi důležitým atributem synchronizovaných entit bude `source`, neboli zdroj, ze kterého entita pochází. Pokud jsou tedy konkrétně na SZŠ ČB instalovány dvě instance programu Bakaláři, jeden pro střední školu a jeden pro vyšší odbornou školu, bude určeno, ke které instanci entita patří. To umožní nejenom přidávat libovolné množství zdrojů (další instance aplikace Bakaláři, ruční zadání uživatelů administrátorem, aj.), ale i určit při distribuci dat z aplikace INUM, ve kterém zdroji se entita nachází. V případě, že jsou zdrojem dat Bakaláři, všechny vzniklé entity mají nastaven příznak synchronizovat s Bakaláři (`bakalariIsSynced`) na „ano“. Pokud zdrojem dat nejsou Bakaláři, musí být tento příznak nastaven na „ne“, INUM nikdy nevytváří nové záznamy v Bakalářích, aby nedošlo k porušení integrity dotyčné databáze.

Dalšími důležitými entitami jsou skupiny (tabulka `skupiny`, složený primární klíč `KOD_SKUP, KOD_TRID a SKOLNI_ROK`) a třídy (tabulka `tridy`, primární klíč `KOD_TRID`). Skupiny patří pod třídy. Příkladem může být skupina „celá třída“, „1. polovina“ nebo „dívky“, které mají u sebe uvedeny specifickým způsobem seznamy žáků/studentů (někdy výčtem primárních klíčů, někdy filtrem apod.). Třídy mají uvedeny mimo jiné třídní učitele (jeho primární klíč). Vazba na atributy entity `Group` aplikace INUM (obrázek 5.3) je velmi podobná jako v případě uživatelů.

Ještě musí být zmíněn jeden důležitý atribut - `isActive` (obrázek 5.3). Když se záznam v Bakalářích označí jako smazaný (atribut `DELETED_RC`) nebo je kompletně archivován, tj. přesunut do příslušné archivační tabulky, INUM situaci vyhodnotí a označí svou entitu za neaktivní. Tímto atributem se následně řídí proces zániku/zneplatnění uživatelského konta (viz *5.3 Jednotný uživatelský účet*).

Bakaláři jsou také zdrojem dat pro rozvrh hodin přenášený do kalendářů

tříd v Google Apps.

5.2.2 Synchronizační služba pro Active Directory

Služba starající se o komunikaci mezi INUM a Active Directory bude vybudována na protokolu LDAP, jehož principy fungování jsou vysvětleny v sekci *2.2 LDAP - Lightweight Directory Access Protocol*. Ke konkrétní implementaci je vybrána Java knihovna Spring LDAP [2], která usnadňuje LDAP operace a je postavena na vzoru JdbcTemplate z frameworku Spring. Knihovna usnadňuje uživatelům provádění standardních úkolů jako vyhledávání a uzavírání kontextů, iterování přes výsledky vyhledávání, zakódování/rozkódování hodnot, filtrování apod.

S Active Directory se synchronizují 2 entity z aplikace INUM, osoby (`Person`) a skupiny (`Group`). Synchronizace osob (žáků, studentů, učitelů) s AD je důležitá nejenom k možnosti uživatelů se svým kontem přihlašovat k počítačům v doméně, ale i pro jejich přihlašování se do e-learningové aplikace Moodle (vysvětleno v *5.1.2 Active Directory a Moodle*). Existence uživatelského účtu v AD bude mít velkou roli také v SSO řešení, které bude na škole nasazeno. Skupiny uživatelů, tedy hlavně skupiny žáků a studentů, slouží pro účely aplikace Moodle při vytváření kurzů a zapisování uživatelů do kurzů.

Adresář Active Directory obsahuje několik unikátních identifikátorů objektů v něm umístěných (viz *2.2 LDAP - Lightweight Directory Access Protocol*). V původní verzi aplikace vytvořené v rámci bakalářské práce ([17]) se pro identifikaci objektů používalo `sAMAccounName`, které je sice unikátní, ale ne neměnné. Proto byl pro jednoznačnou identifikaci objektů vybrán atribut `objectGUID`, jenž je po vytvoření objektu stálý a neměnný a zároveň je unikátní i mimo doménu AD (například při propojení více domén). AD atribut `objectGUID` bude mapován na atribut entit s názvem `ldapObjectGUID` (viz diagram 5.3). Příznak, zda je osoba nebo skupina synchronizována s AD je uložen v atributu `ldapIsSynced`, který je pro všechny záznamy vzniklé z aplikace Bakaláři ve výchozím nastavení nastaven na „ano“. Osoby mají veden ještě jeden atribut specifický pro Active Directory - `ldapExpirationDate`, jenž uvádí, od kterého data nebude osobě umožněno přihlášení, tedy její účet bude deaktivován a archivován.

Součástí synchronizace INUM s Active Directory je také použití shellu se skriptovacím jazykem nazývaným PowerShell. Některé příkazy čistě z jazyku Java nebo Java knihovny v doméně AD nelze nebo velmi obtížně provádět, mezi takové významnější příkazy patří třeba znemožnění uživateli si změnit heslo (důležitá část pro jednotný účet, protože heslo zpět z AD v čitelné formě nelze získat) nebo například operace s oprávněními na domovských složkách uživatelů. Naopak z prostředí PowerShell je snadné a přirozené tyto akce provádět. Možnost pouštět PowerShell skripty byla implementována již do původní verze aplikace INUM z bakalářské práce, velkou nevýhodou však byl velmi pomalý průběh způsobený neustálým spouštěním a ukončováním

[2] `http://projects.spring.io/spring-ldap/`

procesu PowerShell konzole, často tak jednoduché PowerShell příkazy trvalo provést i několik sekund. Podle dostupných logů konkrétně 4 až 5 sekund na provedení jednoho skriptu. Tento čas se samozřejmě nasčítá, pokud je skript prováděn pro stovky uživatelů. Jedním z dílčích cílů této práce je tak vytvořit nepřetržitě běžící PowerShell konzoli vázanou na běh aplikace INUM, čímž by se dosáhlo rapidního snížení doby trvání provádění skriptů i snížení zátěže na systémové prostředky eliminací opakovaného spouštění systémové PowerShell konzole.

5.2.3 Synchronizační služba pro Google Apps

Poslední ze synchronizačních služeb v aplikaci INUM je zcela nová část, oproti bakalářské práci, související s integrací Google Apps. Pro synchronizační účely budou využita API popsaná v sekci *2.3 API pro Google Apps*, respektive jejich knihovny v jazyku Java.

Podobně jako v Active Directory, i s Google Apps se budou synchronizovat entity osoba (`Person`) a skupina (`Group`). Z osob však na přání zákazníka, školy SZŠ ČB, pouze učitelé, tzn. Google účet a k němu přidružené služby jako Gmail, Google Disk, Google kalendář aj. nebudou dostupné pro žáky a studenty, alespoň ne ze školního emailu (ve formátu frank.underwood@szscb.cz). Proto je u entit veden atribut `googleAppsIsSynced` ve výchozím nastavení s hodnotou „ano“ pro záznam učitele vzniklý z aplikace Bakaláři a „ne“ pro všechny ostatní. Takové nastavení umožňuje následně selektivně zapínat či vypínat synchronizaci pro jednotlivé osoby.

Atribut `googleAppsIsSynced` bude veden i u skupin. Skupiny v GA slouží jako emailové distribuční skupiny pro žáky a studenty, například pokud chce uživatel domény Google Apps (učitel) zaslat hromadný email všem žákům příslušné třídy, napíše jeden email s adresátem této distribuční skupiny a email bude doručen všem přidruženým emailovým adresám. Aby skupin v GA nevznikalo příliš mnoho, bude atribut nastaven na hodnotu „ano“ pouze při vzniku skupiny z Bakalářů s označením celá třída (nevznikne tak distribuční skupina třeba 1. polovina třídy). Hodnotu atributu lze kdykoliv ručně změnit, opět se tak dá selektivně zapínat či vypínat synchronizaci skupin.

API pro Google Apps vrací pro vzniklé objekty svůj vlastní identifikátor - ID, který bude mapován na atribut v INUM `googleAppsId`. Jedná se o unikátní řetězec znaků. Po jeho uložení bude synchronizace a identifikace objektu probíhat opět pouze jeho pomocí.

Se skupinami v GA je třeba synchronizovat jejich členy (v názvosloví Directory API se jim říká `Member`). V podstatě se jedná o emailovou adresu člena skupiny, v případě této diplomové práce emailové adresy žáků a studentů. Ty budou získány při procesu označovaném jako aktivace účtu (více v sekci *5.3 Jednotný uživatelský účet*).

S Google Apps budou synchronizovány ještě další objekty - sdílené kontakty domény a kalendáře s jejich událostmi. Sdílené kontakty domény budou vznikat z osob žáků a studentů, a to právě z důvodu absence vytváření jejich kont v Google Apps. Aby bylo snadnější pro uživatele domény (učitele)

kontaktovat konkrétní osobu, bude její kontakt obsažen v adresáři celé domény, nebude přístupný z vnějšku, aby nedošlo k vystavení citlivějších dat, třeba emailové adresy kontaktu. Přístup do adresáře tak mají pouze uživatelé domény. Stejně tomu bude u kalendářů, kde vznikne kalendář pro každou třídu a jeho události bude tvořit rozvrh hodin převzatý z aplikace Bakaláři.

5.2.4 INUM - Integrated User Manager

Aplikace INUM bude pro komunikaci s integrovanými aplikacemi používat výhradně výše zmíněné synchronizační služby, každou zodpovědnou právě za jednu aplikaci. Vždy po synchronizaci konkrétního objektu s příslušnou aplikací bude uložen čas jeho poslední synchronizace (atributy v INUM `bakalariLastSynced`, `ldapLastSynced` a `googleAppsLastSynced`).

Zatím se jedná pouze o informační údaj, ale s pomocí auditačního projektu Hibernate Envers [3], který je již v aplikaci implementován k entitě osoby, se bude dát výrazně snížit zátěž na jednotlivé integrované aplikace. Například již nebude nutná každodenní pravidelná synchronizace všech objektů do všech aplikací, ale pouze těch, u kterých nastala od jejich poslední synchronizace změna. Z velké části by tak byl kompenzován nedostatek architektury Hub-and-Spoke, tj. pokud je aplikace v čase synchronizace nedostupná, v příštím cyklu synchronizace je přesně dáno, které objekty je nutné z důvodu jejich změny synchronizovat a které ne, a nemusí tak probíhat iterace synchronizace přes všechny synchronizované objekty jen pro ujištění, že každá změna byla promítnuta a byla tak zajištěna integrita a konzistence.

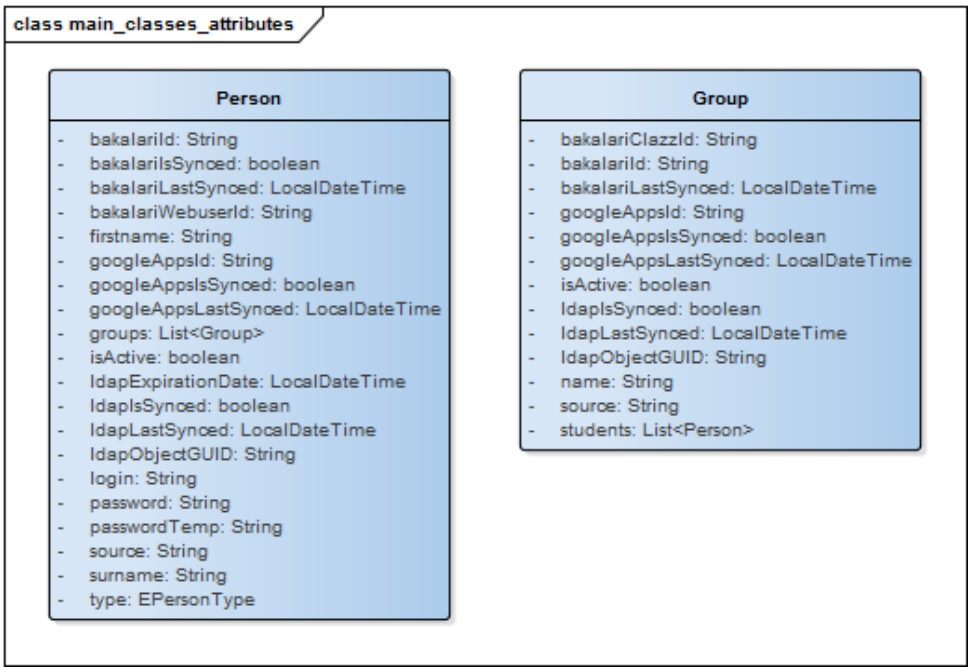
Na schématu 5.3 jsou uvedeny dvě hlavní entity celé integrace, osoby (třída `Person`) a skupiny (třída `Group`) s některými důležitými atributy, hlavně pro procesy synchronizace a vytváření jednotného uživatelského účtu. Kompletní diagram entit, jejich atributů a vztahů mezi entitami je uveden v příloze *B ER diagram*.

Aplikaci INUM budou smět spravovat uživatelé s rolí administrátora. Tito uživatelé budou uchováváni ve zcela jiné databázové tabulce než osoby, kterých se týká jednotný uživatelský účet a jednotná ověřovací služba. Administrátor aplikace totiž nemusí mít v dané organizaci účet, například autor této diplomové práce. Přístup do administračního rozhraní integrační aplikace bude odlišný od běžných uživatelů a pouze za použití uživatelského jména a hesla. Po přihlášení bude moci administrátor ručně pouštět synchronizační akce, spravovat konfigurační položky aplikace, procházet systémové logy, nastavovat emailové šablony a spousty dalších akcí.

5.3 Jednotný uživatelský účet

Jednotný uživatelský účet bude spravován z aplikace INUM. Aplikace bude poskytovat webové rozhraní přístupné všem uživatelům na adrese `https:`

[3] `http://docs.jboss.org/envers/docs/`



Obrázek 5.3: Hlavní třídy v aplikaci INUM a jejich atributy.

`//mujucet.szscb.cz`. Akce, které může uživatel provádět, jsou vidět na obrázku 5.4 a jsou rozděleny pomyslně do dvou kategorií.

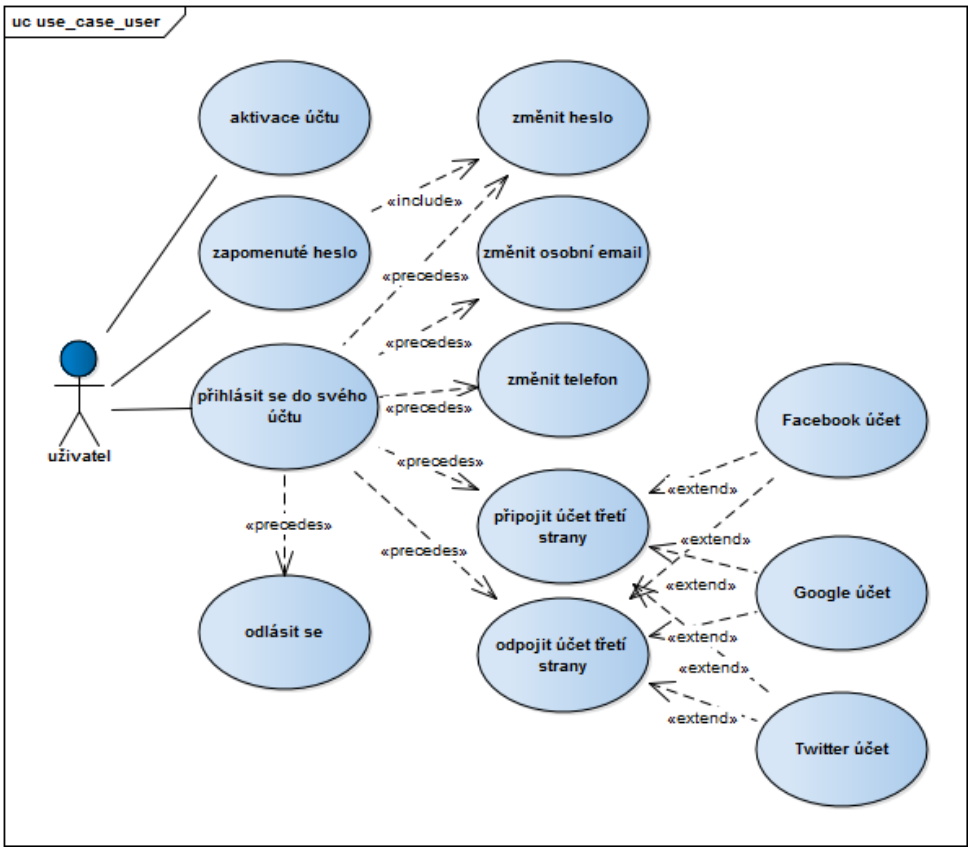
První kategorie akcí jsou ty, které lze provádět bez ověření (přihlášení) uživatele vůči aplikaci INUM. Akce budou pouze dvě, tzv. aktivace účtu (viz proces vzniku uživatelského účtu) a zapomenuté heslo.

Procesem aktivace účtu si bude muset projít každý uživatel jednotného účtu. Bez něj nelze provádět akce zapomenutého hesla, přihlásit se do administrace účtu, dokonce bez předchozí aktivace nevzniknou účty v žádné další integrované aplikaci (Active Directory, Google Apps, Moodle).

Při provedení akce zapomenutého hesla bude uživatel vyzván k zadání své osobní emailové adresy (uvedenou při aktivaci nebo následně změněnou v administraci účtu), na kterou mu budou odeslány instrukce ke změně hesla, konkrétně unikátní odkaz do aplikace INUM platný pevně specifikovaný čas, po jehož navštívení bude moci uživatel nastavit nové heslo. Tímto mezikrokem přes zaslaný email se ověří identita uživatele. Nikdo nebude schopný heslo změnit jinému uživateli za předpokladu, že nemá přístup k jeho osobní emailové adrese.

Druhou kategorií akcí představují ty, které lze provádět až po ověření vůči aplikaci INUM, použitím jednoho z mechanismů přihlášení přes standardní formulář, přihlášení přes SSO nebo přihlášení pomocí účtu třetí strany (viz obrázek 5.5).

Po přihlášení do svého účtu uvidí uživatel základní údaje (jméno a příjmení, pozici, osobní či pracovní email, telefon) a bude moci provádět akce změny hesla, změny osobního emailu, změny telefonu a připojení či odpojení účtu třetí strany. Samozřejmě se bude moci odhlásit, pokud byl přihlášen přes SSO,



Obrázek 5.4: Případ užití aplikace INUM uživatelem.

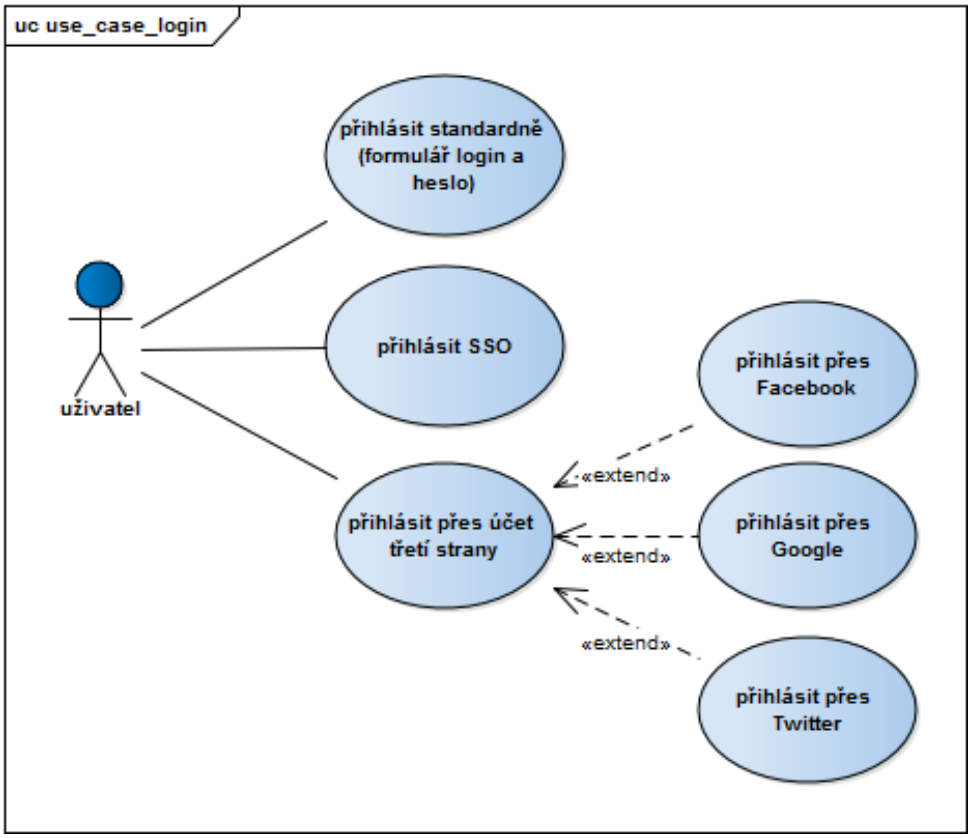
bude mít možnost se odhlásit i celkově od poskytovatele identity (takzvané odhlášení SSO).

Změna hesla bude probíhat bez mezikroku se zasláním emailu s instrukcemi a odkazem, jako tomu bude v případě zapomenutého hesla, protože uživatel se již ověřil vůči aplikaci INUM. Bude přesměrován na formulář, kde bude nucen zadat původní heslo (ochrana proti zneužití cizího účtu) a dvakrát nové heslo, následně po potvrzení bude okamžitě provedena změna hesel v integrovaných aplikacích (jen v těch, se kterými je jeho účet synchronizován).

Při změně osobního emailu bude na nově zadanou emailovou adresu zaslán potvrzovací email, z důvodu ověření vlastnictví uživatele dané emailové schránky. Teprve poté bude osobní email změněn a distribuován do návazných aplikací.

Údaj telefonu nenabývá takové důležitosti jako osobní email, jeho změnu tak zajistí jednoduchý formulář s novou hodnotou mobilního (telefonního) čísla. Opět se bude distribuovat do dalších aplikací, v tomto ohledu je nejdůležitější promítnutí změny do aplikace Bakaláři, kde se telefonní číslo používá při kontaktování žáka, studenta nebo učitele převážně v mimořádných situacích.

Poslední akcí v detailu účtu uživatele je připojení, respektive odpojení účtu třetí strany. Uživatel autorizuje aplikaci třetí strany, ve většině případů použitím protokolu OAuth. Po autorizaci externího účtu se bude schopen



Obrázek 5.5: Možnosti přihlášení uživatele do aplikace INUM.

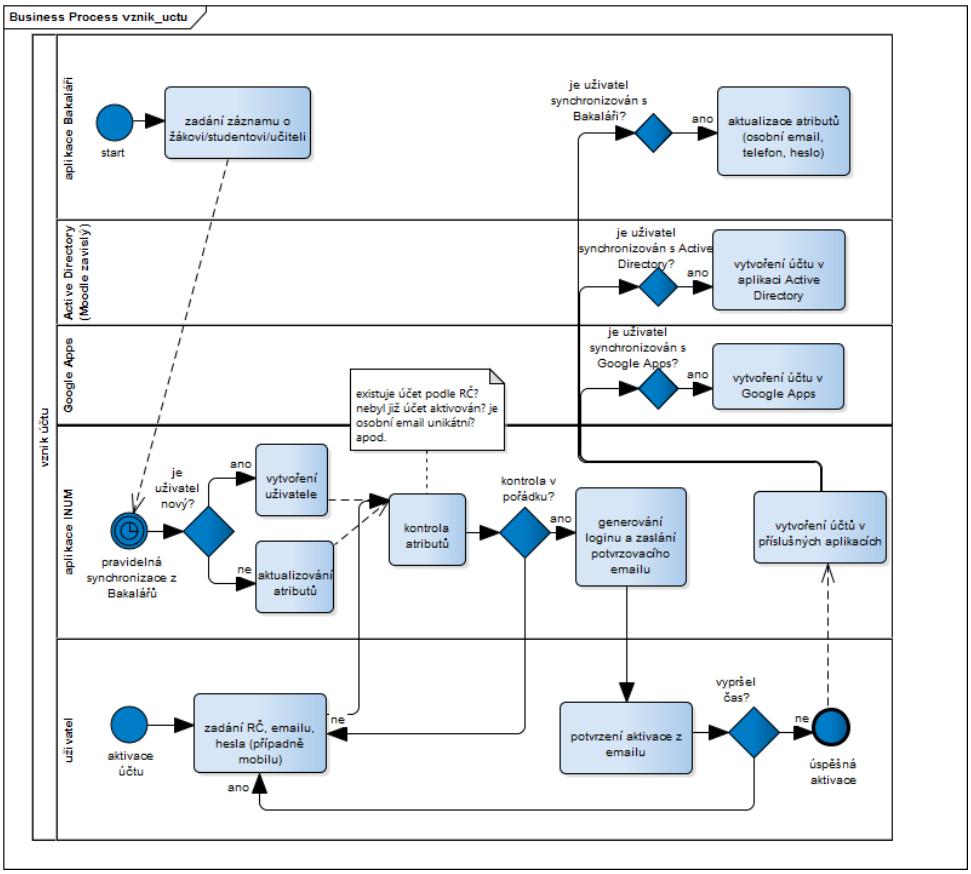
pomocí něj přihlašovat ke svému jednotnému účtu v aplikaci INUM. Více v sekci *5.4 Jednotná ověřovací služba*.

5.3.1 Vznik a zánik uživatelského účtu

Nejdůležitějšími procesy z hlediska jednotného uživatelského účtu jsou jeho vznik a zánik. Musí být přesně určeny podmínky, za kterých bude účet vytvořen a kdy se bude synchronizovat s napojenými aplikacemi. Stejně tak kdy je nutné účet zneplatnit, případně zcela vymazat.

Vznik uživatelského účtu definuje proces zachycený na diagramu 5.6. V první řadě by měl vzniknout záznam o osobě v aplikaci Bakaláři. Nejedná se sice o nutnou podmínku, lze si představit případ, kdy bude záznam vytvořen, například přímo skrze formulář aplikace INUM, nicméně ve většině případů jsou všechny osoby působící na škole vedeny v aplikaci Bakaláři. Po uložení záznamu v aplikaci Bakaláři INUM automaticky a pravidelně alespoň jednou denně iniciuje synchronizaci s Bakaláři. Tu lze vyvolat i ručně přes administrační rozhraní. Běžný případ využití je však automatický proces. INUM záznam zpracuje a vytvoří ve vlastní databázi odpovídající záznam s vazbou na unikátní identifikátory z aplikace Bakaláři.

Identifikátory jsou dva, jeden pro osobu (z tabulek žáků či studentů)



Obrázek 5.6: Proces vzniku uživatelského účtu.

a jeden z tabulky přístupů k webovému rozhraní aplikace Bakaláři. Bakaláři při vytváření záznamu o osobě generují podle vlastních pravidel login a heslo. Z pohledu vzniku jednotného účtu je však důležitý pouze vznik záznamu v příslušné tabulce, login totiž bude generován aplikací INUM a heslo si zvolí uživatel své vlastní.

V momentě, kdy je záznam přítomen v aplikaci INUM, uživatel může provést akci nazvanou **aktivace účtu**. Tato akce byla zavedena z důvodu sběru informací o uživateli, hlavně jeho hesla a osobního emailu, čímž se odstraní administrativní zátěž na administrátora aplikace Bakaláři (distribuce vygenerovaných loginů a hesel uživatelům), případně na třídní učitele (sběr emailů studentů ve třídě, kontaktních telefonů apod.).Bez provedení aktivace se s uživatelským kontem neděje vůbec nic, uživatel nezná ani login a heslo z aplikace Bakaláři.

Uživatel se musí při aktivaci nějakým způsobem spojit (ověřit) se záznamem v databázi INUM. Jediný, těžce uhodnutelný a dostatečně unikátní údaj, o kterém by měla vědět jenom aplikace INUM (respektive Bakaláři), je v momentě aktivace rodné číslo uživatele.

Použití rodného čísla může být problematické. Úřad pro ochranu osobních údajů na svých stránkách uvádí: „Výčet osobních údajů, které jsou dle § 4 písm.

b) zákona č. 101/2000 Sb., o ochraně osobních údajů a o změně některých zákonů, považovány za citlivé údaje, je taxativní, tj. jde o uzavřenou skupinu údajů. Vzhledem k tomu, že rodné číslo není v tomto výčtu uvedeno, nelze jej považovat za citlivý údaj. Je však nepochybně osobním údajem ve smyslu podle § 4 písm. a) zákona o ochraně osobních údajů, a ačkoli nejde o citlivý údaj, o jeho důležitosti a zvláštním charakteru svědčí, že úprava jeho využití je upravena v zákoně č. 133/2000 Sb., o evidenci obyvatel a rodných číslech a o změně některých zákonů.“ [32]

Uživatelé však podepisují se školou souhlas se zpracováním osobních údajů, který se týká i rodného čísla. Za předpokladu, že jsou údaje osob v aplikaci INUM dostatečně chráněny (šifrování apod.) a aplikace údaje nikde nešíří ani nezveřejňuje, mělo by být ověření rodným číslem v pořádku. Pokud by přeci jen uživatel měl se zadáním citlivého údaje problém, kontaktuje správce aplikace, který provede aktivaci ručně v administraci INUM.

Aktivace kromě spojení se záznamem pomocí rodného čísla obsahuje i zadání osobní emailové adresy (kontroluje se, že se nejedná o emailovou adresu z domény organizace), která musí být unikátní přes všechny osoby v aplikaci. Například, když by na škole působili sourozenci a z nějakého důvodu používali totožnou osobní emailovou adresu, aktivace se povede pouze prvnímu z nich, druhý by musel použít odlišnou adresu. Dále se uvádí nepovinně telefonní (mobilní) číslo, u něhož se kontroluje jen správný formát, nikoliv existence, a v neposlední řadě uživatelovo heslo (pro kontrolu zadávaní dvakrát).

Po odeslání formuláře se kontrolují zadané hodnoty. Nejprve proběhne kontrola, zda existuje v aplikaci INUM účet s příslušným rodným číslem a zda již tento účet nebyl aktivován - proces aktivace lze provést právě jednou. Pokud jsou podmínky splněny, tj. osobní email uživatele je unikátní a heslo splňuje bezpečnostní politiku (ve výchozím stavu minimálně 8 znaků a 1 velké písmeno a 1 číslo), na zadaný osobní emailový účet uživatele je zaslán potvrzovací email.

Obsahem emailu jsou instrukce, unikátní odkaz pro dokončení aktivace v aplikaci INUM a nově vygenerovaný unikátní login uživatele, představující název jednotného účtu uživatele v prostředí organizace. Login je konstruovaný podle předem specifikovaných podmínek:

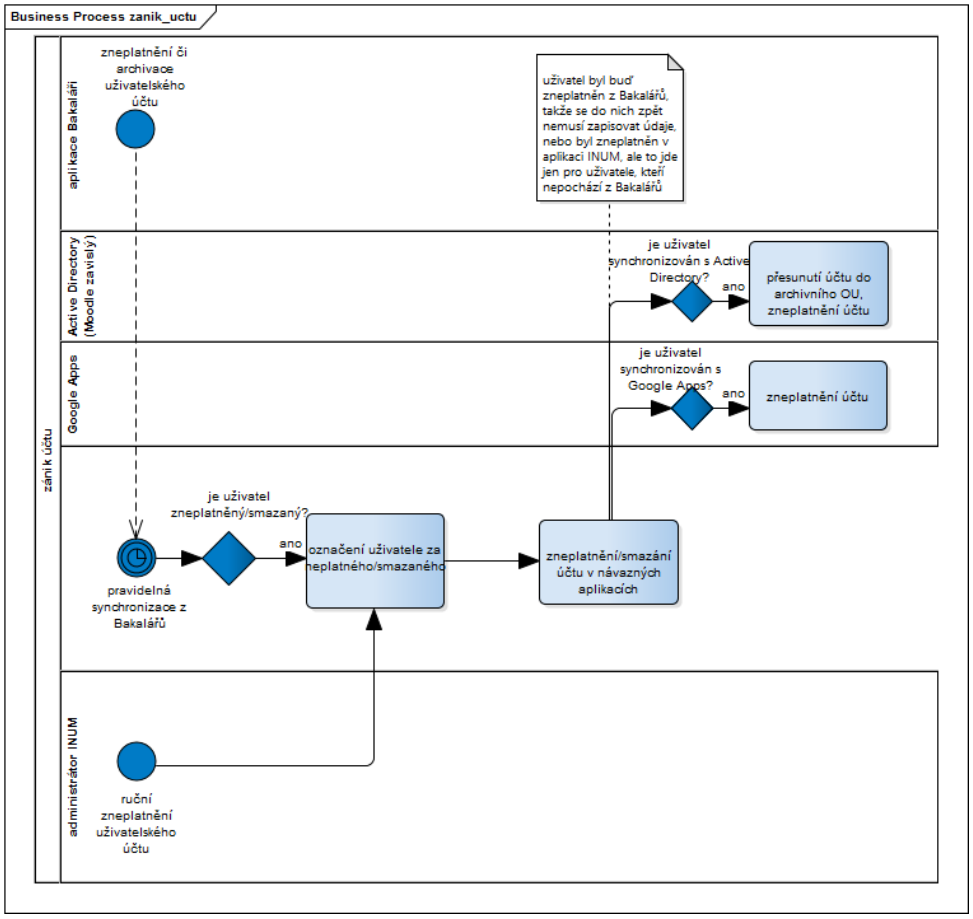
- **Učitelé** - příjmení, pokud je obsazené tak příjmení + tečka + jméno, pokud je obsazené příjmení + tečka + jméno + sekvenční číslo (například `underwood`, resp. `underwood.frank`, resp. `underwood.frank1`).
- **Žáci a studenti** - prvních 5 písmen z příjmení + 3 sekvenční čísla (například `under001`, resp. `under009`).

Kontrola unikátnosti loginu je provedena vůči všem integrovaným aplikacím, tj. ověří se, zda je login unikátní v INUM, v aplikaci Bakaláři, v Active Directory a pokud se jedná o učitele, i v Google Apps.

Po kliknutí uživatele na odkaz v potvrzovacím emailu je uživatel informován o zdařilé aktivaci účtu a zároveň je mu řečeno, že během chvíle se bude moci přihlašovat do aplikací nově vygenerovaným loginem a jím zvoleným heslem.

INUM se totiž bude snažit ihned nově získané informace od uživatele distribuovat do všech navázaných systémů. Pokud by se, například z důvodu chyby nebo nedostupnosti určité aplikace, distribuce nezdařila, nejpozději následující noci se INUM pokusí synchronizaci provést znovu v rámci automatické kompletní synchronizace.

Zánik účtu, respektive jeho zneplatnění, zobrazené na diagramu 5.7, představuje značně jednodušší proces než vznik účtu. Nevyžaduje totiž pochopitelně součinnost uživatele, o jehož konto se jedná. Není důvod, aby sám uživatel chtěl svůj účet zneplatnit, čímž by si zamezil přístup do školních aplikací.



Obrázek 5.7: Proces zániku/zneplatnění uživatelského účtu.

Zneplatnit účet může buď administrátor aplikace Bakaláři nebo administrátor aplikace INUM. Zneplatnění pomocí Bakalářů může probíhat třemi způsoby. Buď je osoba plně archivována, tedy zmizí z příslušné tabulky do archivační tabulky, nebo je její atribut `DELETED_RC` nastaven na hodnotu „ano“ nebo je nastaven její atribut platnosti (týká se žáků a studentů, konkrétně atribut `EVID_DO`). V každém případě INUM vyhodnotí účet jako neplatný a spustí proces deaktivace účtu v návazných aplikacích.

Administrátor aplikace INUM může zneplatnit účet přepínačem ve formuláři

úpravy osoby, v současném nastavení ale jen v případě, že uživatel nevznikl z aplikace Bakaláři. V takovém případě se berou jako směrodatné údaje vedené v její databázi.

Zneplatnění účtu v Active Directory, pokud je osoba s AD synchronizována, se provádí zakázáním účtu a přesunutím celého objektu do archivační organizační jednotky. Zakázáním účtu se znemožní i přihlášení k účtu a kvůli tomu se nepůjde uživateli přihlásit ani do aplikace Moodle. Pokud je osoba synchronizovaná s Google Apps, zneplatnění probíhá zakázáním daného účtu v doméně GA, opět se uživateli nepodaří se přihlásit, například ke službě Gmail. Zakázání účtu v aplikaci Bakaláři neprobíhá, buď je totiž iniciováno právě z aplikace nebo osoba není vůbec s Bakaláři synchronizována.

5.3.2 Změna jednotného hesla

Diagram 5.8 zobrazuje proces změny hesla. Změnu hesla lze provést dvěma způsoby. Zaprvé po přihlášení uživatele k vlastnímu účtu v aplikaci INUM a provedením akce změnit heslo, nebo zadruhé spuštěním procesu zapomenutého hesla. Obě možnosti ze strany uživatele již byly popsány dříve v této sekci.

Po úspěšné změně hesla uživatelem spustí aplikace INUM proces aktualizace hesla v návazných aplikacích. Pro každou integrovanou aplikaci, v níž se nachází odpovídající účet, se zavolá metoda pro změnu hesla. V Active Directory tato akce představuje aktualizaci jednoho atributu objektu typu `person` a spuštění PowerShell skriptu pro nastavení přepínačů v AD - nemožnost uživatele si přímo v AD změnit heslo a nastavení platnosti hesla nastálo. Uživatelé používající Moodle se ověřují vůči doméně Active Directory, takže i pro ně má změna okamžitou platnost. V aplikaci Bakaláři se musí změnit SQL dotazem příslušný řádek v tabulce webových přihlášení (`webuser`), heslo v odpovídající zahashované formě. Pro Google Apps (týká se učitelů) se provede tzv. `patch request` obsahující heslo také jako hash.

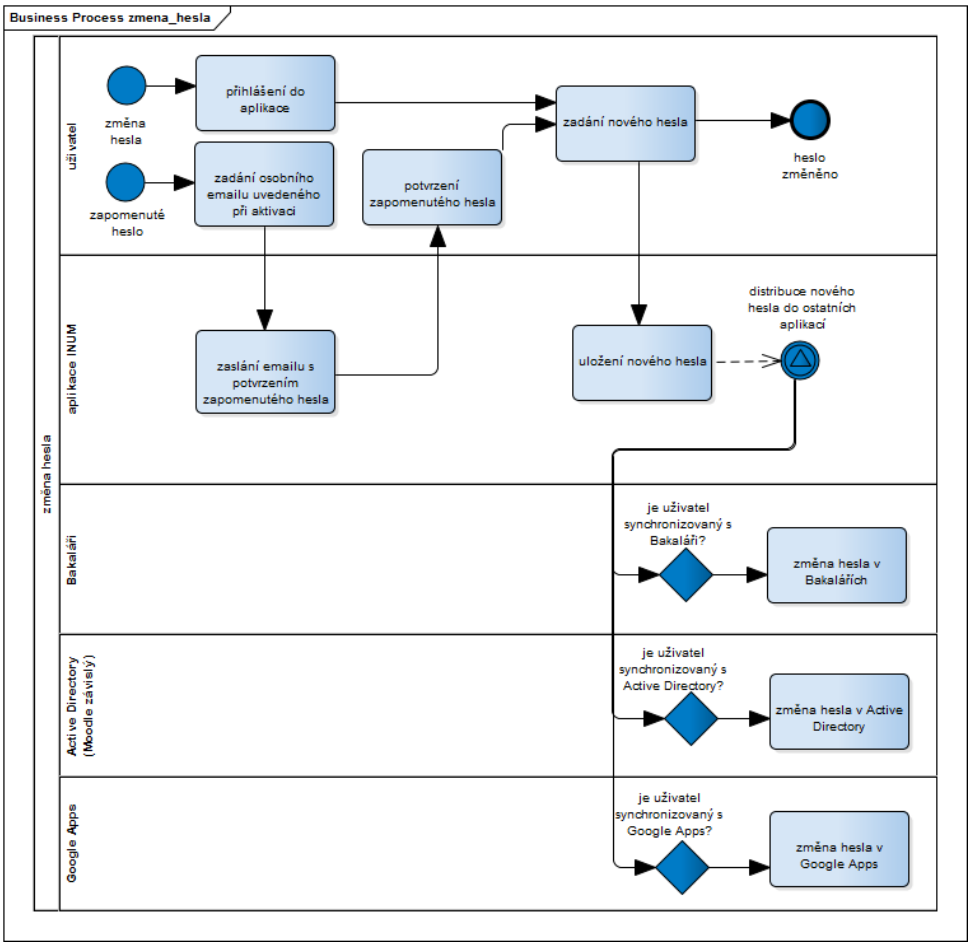
Ve všech aplikacích kromě INUM je zakázáno si heslo změnit nebo takovou možnost vůbec nemají. To zajišťuje, že jediným místem a tím pádem i správcem hesel je právě aplikace INUM.

5.4 Jednotná ověřovací služba

Jedním ze dvou hlavních cílů diplomové práce je návrh a implementace SSO jednotné ověřovací služby. V kapitole *4 Analýza mechanismů ověřování* bylo vysvětleno, proč se vyplatí použít službu postavenou na SAML. Takových služeb existuje spousta, z těch známějších například OpenAM, JOSSO, PingFederate, Shibboleth, SimpleSAMLphp, ADFS a jiné. [4]

Univerzálním řešením by bylo použít např. Shibboleth, protože se jedná o open source, avšak jeho nasazení je náročné nejen na systémové prostředky, ale i na konfiguraci. Prostředí školy SZŠ ČB je skoro celé založené na operačním

[4] `https://en.wikipedia.org/wiki/SAML-based_products_and_services`



Obrázek 5.8: Proces změny hesla uživatele.

systému Window Server, zdá se tak vhodnější a jednodušší na konfiguraci volba ADFS. Je nutné zmínit, že volba služby není žádný neměnný krok, jedná se pouze o poskytovatele SAML identit a SSO jednotné ověřovací služby. Kdykoliv, pokud by se zvolené řešení zdálo nevhodné, je možné poskytovatele identity zaměnit a nasadit jiného.

5.4.1 ADFS - Active Directory Federation Services

Jak se přechází ze služeb umístěných v korporátních datacentrech na služby působící externě v „cloudu“, vyvinula se i potřeba pro rozšíření autentizačních mechanismů. Odpověď Microsoftu na tuto potřebu je Active Directory Federation Services (ADFS). ADFS je samostatná služba nesoucí značku Active Directory, takže se předpokládá že administrátoři AD budou připraveni službu ADFS nasadit a spravovat. [9]

Srdcem a duší ADFS prostředí je **federační server** (angl. *federation server*). Federační server je schopen fungovat jako poskytovatel identity (IdP) i RP (angl. *relying party*, spoléhající strana), v závislosti na scénáři nasazení.

Všechny protokoly, které ADFS ovládá (hlavně tedy SAML a WS-Fed), jsou poskytovány federačním serverem.[9]

Federační server musí být správně a řádně zabezpečen, jak fyzicky, tak i logicky. Federační servery jsou jako doménové kontrolery ve smyslu kontroly přístupu k aplikacím a službám. Kdokoliv s přístupem k federačnímu serveru by se mohl vydávat za jiného uživatele v organizaci vytvořením falešného tokenu podepsaného certifikátem federačního serveru. RP by se token zdál autentický a validní a zacházela by s ním jako by byl pravý.[9]

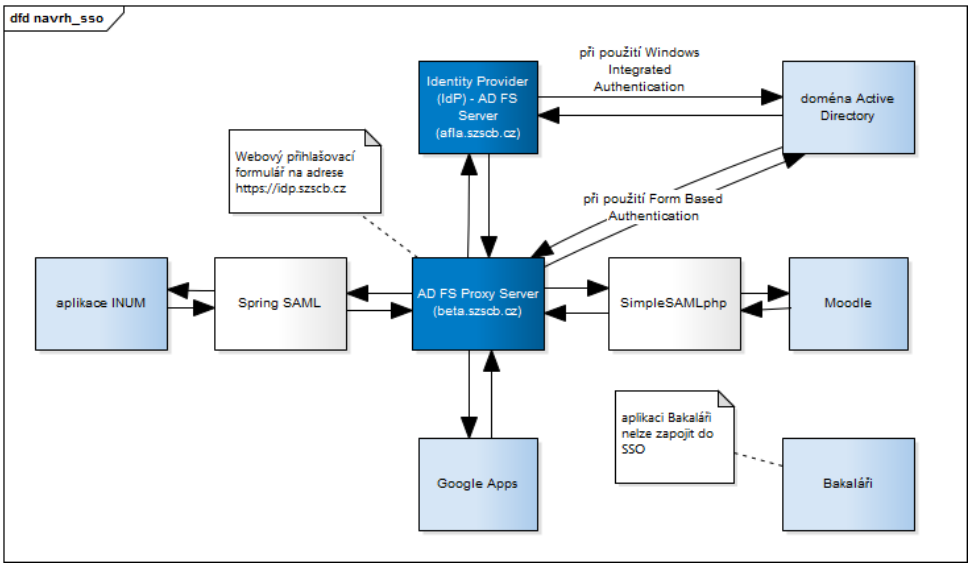
Proxy federačního serveru je volitelná komponenta používaná k poskytnutí vrstvy mezi federační server a internet. Neexistuje funkce, kterou by nešlo v ADFS provádět bez nasazení ADFS proxy serveru, ale existují scénáře, kdy je daleko snazší pracovat s pomocí proxy.[9]

Hlavní výhodu, kterou přináší federační proxy server, je možnost rozdílné autentizace pro interní a externí uživatele. Pokud se přihlašují uživatelé zevnitř organizace na korporátní síť a chtějí získat token, můžeme spoléhat na Windows Integrated Authentication (Kerberos a NTML) k projití skrze ADFS bez jakýchkoliv vyskakovacích oken a hlášek. Externě se naopak mohou uživatelé připojovat například z jejich PC nebo z kavárny. V takovém případě raději poskytneme uživatelsky přívětivý HTML formulář (Form Based Authentication) k autentizaci uživatele. Federační proxy server také izoluje externí klienty od relativně citlivých dat na federačním serveru použitím soukromých klíčů, které podepisují tokeny. To přináší další vrstvu bezpečnosti často oceňovanou a chtěnou organizacemi.[9]

Jeden ze scénářů nasazení ADFS vyžaduje dva servery, jeden bude fungovat jako federační server a starat se o konfigurační databázi, druhý jako proxy federačního serveru, jenž vystavuje ADFS na internet. Tato topologie je vhodná pro organizace, které potřebují federovat s jedním nebo více partnery a aplikacemi, ale nemá potřebu duplikovat na úrovni serverů. Architektura ADFS relativně ulehčuje později tuto topologii škálovat k poskytnutí více federačních serverů nebo více federačních proxy serverů. [9]

Na obrázku 5.9 návrhu architektury SSO je vidět využití pravě scénáře se dva servery, server `alfa` slouží jako federační server a uchovává konfiguraci ADFS (zároveň slouží i jako doménový kontroler), server `beta` je zase nasazen jako federační proxy server, tím pádem DNS záznamy z internetu směřují pro IP adresu tohoto serveru, který vystavuje ADFS ven z organizace.

Pro interní uživatele, přihlášené do domény Active Directory, lze zvolit dva typy autentizace, buď Windows Integrated Authentication nebo Form Based Authentication. Z důvodu konzistence přihlášení bude v současné době použito přihlášení pomocí formuláře, u kterého lze snadno upravovat vzhled přihlašovací stránky, zatímco při použití integrované autentizace ve Windows může uživateli vyskočit klasické přihlašovací okénko ve vzhledu systému Windows, se kterým by si neznalý uživatel nemusel vědět rady. Navíc se počítá spíše s externím využitím SSO, převážně pro žáky a studenty z jejích vlastních PC nepřipojených do domény organizace. Webový formulář má tu nevýhodu, že i když je uživatel připojený do domény, minimálně jednou po něm bude chtít ADFS zadat přihlašovací údaje znovu, zatímco Windows



Obrázek 5.9: Návrh architektury SSO.

Integrated Authentication považuje takového uživatele již za přihlášeného.

Část konfigurace ADFS federačního serveru si lze prohlédnout v příloze *C Konfigurace ADFS*. V této příloze je uveden i navržený vzhled přihlašovací stránky ADFS federačního proxy serveru, který je přizpůsoben vzhledu aplikace INUM, aby měl uživatel dojem uceleného jednotného účtu a přihlašování.

Každá z aplikací zapojená do SSO autentizace musí být nastavena v ADFS federačním serveru jako Relying Party, v ADFS označovaném jako Vztahy důvěryhodnosti předávající strany. V příloze *C Konfigurace ADFS* jsou zachyceny úseky konkrétní konfigurace pro aplikaci Moodle.

Aplikace INUM bude napojena na federační proxy server pomocí knihovny Spring Security SAML [5], respektive jejího rozšíření spring-security-adfs-saml2 [6]. Část konfigurace SAML SSO konektoru je uvedena v příloze *D Konfigurace SSO v INUM*.

E-learningový nástroj Moodle nedisponuje žádnou knihovnou podporující přímo SAML, alespoň ne ve verzi instalované na škole SZŠ ČB. Bude proto nutné nasadit a nakonfigurovat PHP aplikaci SimpleSAMLphp [7] jako Relying Party spoléhající na ADFS a zároveň doinstalovat do Moodle rozšíření moodle-auth_saml [8] komunikující právě se SimpleSAMLphp. Některé části konfigurace lze vidět v příloze *F Konfigurace SSO v Moodle*.

Google Apps nevyžadují žádnou dodatečnou knihovnu nebo aplikaci, v jejich administrační konzoli je možné nastavit přihlašování k libovolné SSO službě postavené na SAML. I to je jeden z důvodů, proč bylo pro implementaci SSO řešení použito SAML. Snímek obrazovky s konkrétním nastavením SSO v Google Apps je umístěn v příloze *E Konfigurace SSO v Google Apps*.

[5] `http://projects.spring.io/spring-security-saml/`
[6] `https://github.com/choonchernlim/spring-security-adfs-saml2`
[7] `https://simplesamlphp.org/`
[8] `https://github.com/piersharding/moodle-auth_saml`

Aplikaci Bakaláři nelze žádným způsobem zapojit do SSO, v současné podobě nepodporují knihovny třetích stran ani neposkytují přímé napojení jako třeba Google Apps. V této situaci možná nastane změna, vzhledem k chystanému předělání designu aplikace.

ADFS vyžaduje po předávající straně (*relying party*), jedná-li se i webovou službu, zabezpečení protokolem HTTPS. Google Apps již tuto podmínku splňují, pro aplikace INUM a Moodle byl vygenerován bezplatný certifikát od StartSSL [9] a aplikován na webový server Apache.

5.4.2 Přihlášení přes účty třetích stran

V aplikaci INUM nelze vytvářet účty na základě informací z účtu třetí strany, například Facebook, protože uživatelé jsou vázáni na organizaci a nepřipadá tak v úvahu, že by její aplikace a prostředky využívala externí osoba, která s organizací nemá nic společného.

Z tohoto důvodu tak nejde aplikovat podobný proces jako například u internetového obchodu, kde při vzniku účtu zákazníka lze načíst jeho údaje po autorizaci právě třeba ze sociální sítě Facebook nebo poskytovatele OpenID, například MojeID.

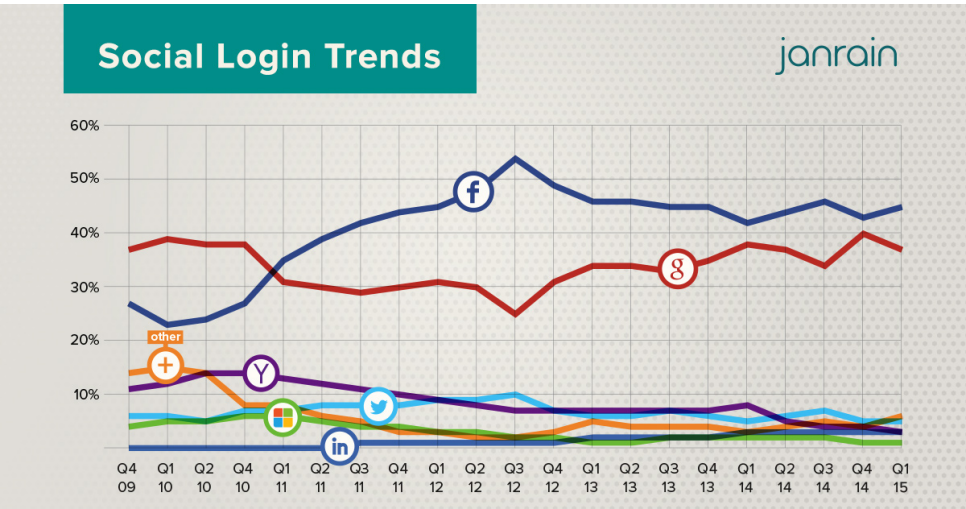
Lze však využít funkcionality ověření přes účet třetí strany. Pokud by uživatel autorizoval aplikaci INUM třeba v aplikaci Facebook, INUM by následně mohl uživatele autentizovat pomocí jeho účtu třetí strany. Když je uživatel přihlášen ke svému účtu v aplikaci Facebook, INUM se aplikace dotáže na autentizaci, nejčastěji pomocí protokolu OAuth, a Facebook následně poskytne aplikaci INUM údaj, že se jedná o danou osobu, která je pak schopná uživatele přihlásit k jeho účtu.

Bylo nutné vybrat vhodné aplikace třetích stran. Průzkum společnosti Janrain uvádí (viz graf 5.10): „ První čtvrtletí 2015 opět trochu rozšířilo mezeru - podíl Google spadl o 3% na na 37%, Facebook vyrostl o 2% na 45%. Zatímco oba poskytovatelé identity (IdP) pociťují velmi malou konkurenci od ostatních sociálních sítí jako Twitter (5%), Yahoo (3%) a LinkedIn (3%), vertikální nestálost pokračuje v reflektování změn preferencí konzumentů, s tím, jak různí IdP poskytují odlišné výhody pro odlišné typy webových stránek.“ [21]

Výběr aplikací byl převážně řízen těmito preferencemi uživatelů v oblasti Social Login, protože velkou část uživatelů aplikace INUM, a tedy jednotného uživatelského účtu, tvoří žáci a studenti, u kterých je daleko větší předpoklad používání sociálních sítí než u učitelů. Pro implementaci přihlášeni byly tak navrženy 3 aplikace třetích stran - Facebook, Google a Twitter. Pro aplikace postavené na Spring Framework, jako právě INUM, existuje knihovna určená přesně k těmto účelům - Spring Social [10].

[9] `https://www.startssl.com/`

[10] `http://projects.spring.io/spring-social/`



Obrázek 5.10: Graf trendů v Social Login v 1. čtvrtletí 2015. (převzato z [21])

Kapitola 6
Implementace

V této kapitole budou ukázány zajímavé či důležité úseky z implementace nástroje po potřeby diplomové práce. Jedná se většinou o úseky související s implementací jednotného uživatelského účtu, návrh a popis jednotné ověřovací služby je uveden v předchozí sekci nebo v přílohách, protože se ve většině případů jedná o konfiguraci cílových aplikací.

Aplikace INUM je v základu postavena na Spring Framework, aplikující softwarový návrhový vzor *Dependency Injection* (DI, vkládání závislostí). Bezpečnost je zaručena knihovnou Spring Security, úzce spojenou se Spring Framework. Díky použití Spring Security bylo pro cíl jednotné ověřovací služby využito knihoven Spring SAML a Spring Social, kde první umožňuje komunikaci s poskytovatelem identity skrze SAML a druhá dovoluje se do aplikace přihlašovat účty třetích stran, většinou pomocí protokolu OAuth.

Pro komunikaci s Active Directory je využívána knihovna Spring LDAP, s Bakaláři JDBCTemplate (součást Spring Framework) a s Google Apps již popsané knihovny komunikující s dostupnými API.

Databáze INUM používá transakční databázi MySQL, k persistování a mapování objektů (tzv. ORM - *Object-Relational Mapping*, objektově relační mapování) pak projekt Hibernate.

O vizuální stránku aplikace se stará šablonovací jazyk Freemarker, front-end je vyveden ve frameworku Bootstrap.

Webový server tvoří ve vývojovém prostředí Jetty, v produkčním prostředí školy SZŠ ČB naopak Apache Tomcat. Celá aplikace je spravována a řízena buildovacím nástrojem Maven.

6.1 Volání synchronizačních úloh

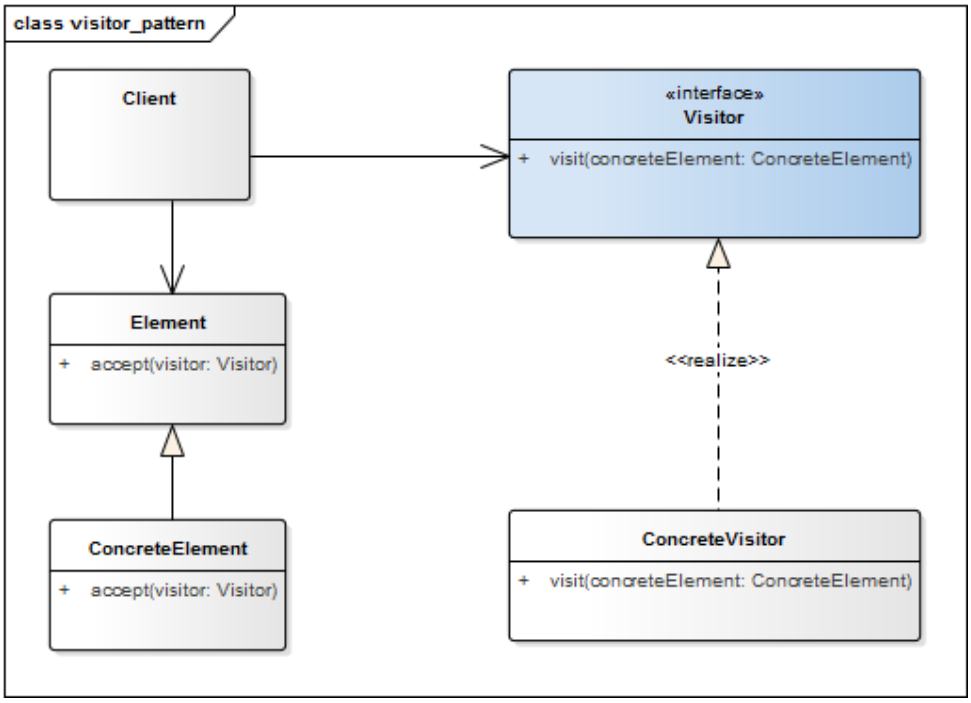
`Visitor`, známý jako behaviorální návrhový vzor, je používán k řízení algoritmů, vazeb a zodpovědností mezi objekty. Definice vzoru `Visitor` poskytnutá v původní knize *Gang of Four* (GoF) pro návrhové vzory říká[28]:

> Dovoluje aplikovat za běhu jednu nebo více operací na množinu objektů a odděluje operace od struktury objektu.

Vzor `Visitor` ve skutečnosti vytváří externí třídu, která používá data v jiné třídě. Když je potřeba vykonávat operace na množině objektů, `Visitor`

může být tím pravým vzorem. Kniha GoF uvádí, že vzor přináší třídě další funkcionalitu bez toho, aniž by ji programátor musel změnit. [28]

Diagram tříd je zobrazen na obrázku 6.1. Jádrem vzoru je rozhraní `Visitor`, které definuje `visit` operaci pro každý typ `ConcreteElement` ve struktuře objektů. `ConcreteVisitor` implementuje tyto operace definované v rozhraní. Konkrétní `visitor` si uchovává vnitřní stav, typicky jak traverzuje přes množinu objektů. Rozhraní `element` jednoduše definuje metodu `accept`, která dovoluje, aby `visitor` prováděl některé akce na elementu. `ConcreteElement` implementuje tuto metodu `accept`.[28]



Obrázek 6.1: Návrhový vzor Visitor (inspirováno v [28]).

V aplikaci INUM je návrhový vzor `Visitor` použit pro volání synchronizačních úloh, jak je vidět v ukázce kódu 5. Úloh je velké množství, v aplikaci se rozdělují na dvě skupiny - úlohy synchronizující všechny objekty najednou (někdy taky nazývané *batch* operace, dávkové operace) a úlohy synchronizující jeden jediný objekt, například konkrétní osobu.

INUM synchronizuje tři hlavní entity - osoby (`Person`), skupiny (`Group`) a třídy (`Clazz`), proto jsou i tři synchronizační služby, pro každou entitu jiná (PersonSyncClient, GroupSyncClient, ClazzSyncClient). Synchronizační služby představují v návrhovém vzoru `Visitor` objekt `ConcreteElement`, který implementuje rozhraní `Element`, v aplikaci nazváno jako rozhraní SyncAllVisitable (pro všechny objekty), respektive SyncSingleVisitable (pro jednotlivé objekty).

Konkrétní synchronizační úlohy pak implementují rozhraní SyncAllVisitor (pro všechny objekty), respektive SyncSingleVisitor. V ukázce 5 je uvedena jedna třída synchronizační úlohy starající se o hromadný přenos dat o všech

osobách z aplikace Bakaláři do aplikace INUM. Takových tříd (úloh) je v aplikaci velké množství, starající se o přenos z a do Bakalářů, Active Directory a Google Apps.

```
@Service
public class PersonSyncClient
    implements SyncAllVisitable<Person, PersonSyncClient>{

  @Override
  public String accept(SyncAllVisitor<Person, PersonSyncClient> visitor) {
    return visitor.visit(this);
  }
  // dalsi metody
}

public interface SyncAllVisitable<T, S~extends SyncClientInterface<T>> {
  String accept(SyncAllVisitor<T, S> visitor);
}

public interface SyncAllVisitor<T, S~extends SyncClientInterface<T>> {
  String visit(S syncClient);
}

public class VisitorPersonBakalariSyncAllFrom
    implements SyncAllVisitor<Person, PersonSyncClient> {
  /**
   * nacte vsechny osoby z~Bakalaru pro vsechny zdroje
   * a~sparuje je s~osobami v~databazi INUM
   */
  @Override
  public String visit(PersonSyncClient personSyncClient) {
    String message = "Synchronizace osob ->"
    for (String source : ConfigurationItem.getAppSources()) {
      // ... samotny kod synchronizace osob z~Bakalaru
    }
    return message;
  }
}
```

Ukázka kódu 5: Návrhový vzor Visitor při provádění synchronizačních úloh.

6.2 Konfigurace aplikace INUM

Konfigurační položky v aplikaci INUM slouží k větší flexibilitě kódu a zároveň přenositelnosti aplikace do jiného prostředí, ať se jedná o přenos mezi vývojovým a produkčním prostředím nebo zcela jiným prostředím, například jiná škola, na které by se aplikace nasazovala. Namísto napevno určených hodnot si aplikace INUM načítá pod unikátním kódem z databáze položky a v daných místech kódu používá jejich aktuální načtenou hodnotu, tím pádem se mohou některé položky měnit i za běhu aplikace, bez jejího restartování nebo úplného vypnutí.

Vytvoření konfigurace (konfiguračních položek) v aplikaci INUM představuje poměrně komplexní problém, jednak kvůli různým datovým typům

položek a také kvůli jejich velkému množství. Konfigurace se ale většinou pojí k určité části zdrojového kódu, kde jsou pak využívané. Z toho důvodu je vhodné položky seskupovat právě podle těchto částí.

V INUM byla navržena konfigurace takto seskupených položek za použití formátu JSON, v součinnosti s Java knihovnou pro práci s JSON objekty of Google pojmenovanou Gson[1].

Nejdříve se musí určit skupiny konfiguračních položek, to znamená ty, které spolu souvisí, a vytvořit třídu s konkrétními atributy (položkami) a jejich požadovanými datovými typy. V ukázkovém kódu 6 se jedná o třídy třeba ConfigurationApp, ConfigurationScheduler, ConfigurationSource apod. Třídy odpovídají JSON objektu, naříklad tedy třída ConfigurationApp odpovídá JSON objektu v databázi uvedeném v kódu 7.

JSON objekt je držen v databázi INUM v textové podobě. Po zaregistrování typů v RuntimeTypeAdapterFactory (viz kód 6) se za pomoci metody getConfigurationItemDTOMapFromJSON z řetězce znaků a adaptéru konvertuje JSON struktura na Java objekt příslušné třídy. Následkem této konverze pak vznikají jednotlivé položky s unikátním kódem, jehož prefix tvoří unikátní kód rodičovského JSON objektu (také uložen v databázi) a sufix tvoří samotný název JSON atributu. Například objekt třídy ConfigurationApp má unikátní kód v databázi `app`, atribut toho JSON objektu `url` pak bude mít unikátní kód `app.url`, použitelný a vyhledatelný kdekoliv v prostředí aplikace. Samotné položky jsou v současné době uloženy v paměti, vyplatí se totiž obětovat nepatrný kousek paměti pro uchování, než neustále konvertovat JSON objekty za použití výše uvedené metody. Konverze se provede vždy při startu aplikace a také při změně kterékoliv databázové konfigurační položky.

```
1  public abstract class AbstractConfiguration implements Serializable {
2    public static final RuntimeTypeAdapterFactory<AbstractConfiguration>
3    RUNTIME_TYPE_ADAPTER_FACTORY = RuntimeTypeAdapterFactory
4        .of(AbstractConfiguration.class, "configurationType")
5        .registerSubtype(ConfigurationApp.class)
6        .registerSubtype(ConfigurationGoogleApps.class)
7        .registerSubtype(ConfigurationLdapGroup.class)
8        .registerSubtype(ConfigurationLdapPerson.class)
9        .registerSubtype(ConfigurationSAML.class)
10       .registerSubtype(ConfigurationScheduler.class)
11       .registerSubtype(ConfigurationSource.class);
12
13   public static final TypeToken<AbstractConfiguration> TYPE_TOKEN =
14       new TypeToken<AbstractConfiguration>() {};
15
16   public static Map<String, ConfigurationItemDTO>
17       getConfigurationItemDTOMapFromJSON
18       (List<ConfigurationItemDTO> originalList) {
19     List<ConfigurationItemDTO> configurationItemListJSON =
20         Lists.newArrayList();
21     final Gson gson = new GsonBuilder()
22         .registerTypeAdapterFactory(RUNTIME_TYPE_ADAPTER_FACTORY).create();
23     originalList.stream()
24         .filter(c -> EConfigurationItemType.JSON.equals(c.getType()))
```

[1] https://github.com/google/gson

```
25          .forEach(c -> {
26            AbstractConfiguration abstractConfiguration =
27            gson.fromJson(c.getString(), TYPE_TOKEN.getType());
28            configurationItemListJSON.addAll(
29                abstractConfiguration.getConfigurationItemDTOList(c));
30            });
31      return ConfigurationItemDTO
32          .getConfigurationItemDTOMap(configurationItemListJSON);
33    }
34      // dalsi metody
35  }
```

Ukázka kódu 6: Návrhový vzor Visitor při provádění synchronizačních úloh.

```
1  {
2      "configurationType": "ConfigurationApp",
3      "environment": "PRODUCTION",
4      "locale": "cs_CZ",
5      "powershellLibPath": "e:\\inum\\data\\powershell\\lib\\",
6      "removeOlderThanDaysEmails": 60,
7      "removeOlderThanDaysFiles": 10,
8      "removeOlderThanDaysSystemLogItems": 30,
9      "sources": ["SZSCB", "VOSZCB"],
10     "url": "http://mujucet.szscb.cz"
11 }
```

Ukázka kódu 7: Konkrétní JSON konfigurační položka pro nastavení aplikace.

6.3 Active Directory a objectGUID

Při použití knihovny Spring LDAP pro komunikaci s protokolem LDAP, v aplikaci INUM využívaným ke komunikaci s Active Directory, je obtížné používat unikátní atribut v AD - `objectGUID`. Již bylo vysvětleno, že `objectGUID` není klasickým atributem v LDAP, ale specifickým, unikátním atributem pouze v Active Directory. Je využíván při synchronizaci objektů (osob a skupin) mezi aplikací INUM a Active Directory.

Využití `objectGUID` přinese jednoznačnou identifikaci objektů a zvýší rychlost vyhledání objektu, protože AD ho interně využívá k hledání[2]. Atribut je 128 bitová hodnota (16 bytová), v doméně Active Directory často zobrazovaná jako řetězec znaků ve formátu XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX. Za pomoci standardních nástrojů Spring LDAP ji nelze získat jinak než jako pole bytů (lze získat i řetězec znaků, ale v nečitelné podobě).

Aby byla aplikace INUM schopna uchovávat `objectGUID` v čitelné formě u příslušného objektu, ke kterému patří, je potřeba konverze pole bytů na tento formát. V ukázce kódu 8 je právě taková metoda uvedena. Autor diplomové práce se metodou inspiroval v článku *Java LDAP/JNDI: 2 Ways Of Decoding*

[2] `https://technet.microsoft.com/en-us/library/cc961625.aspx`

And Using The objectGUID From Windows Active Directory [3]. V aplikaci INUM existují i metody pro zpětné zakódování `objectGUID` i transformaci na jiné formáty.

```
1  public static String convertObjectGUIDToDashedString(byte[] objectGUID) {
2    StringBuilder displayStr = new StringBuilder();
3
4    displayStr.append(prefixZeros((int) objectGUID[3] & 0xFF));
5    displayStr.append(prefixZeros((int) objectGUID[2] & 0xFF));
6    displayStr.append(prefixZeros((int) objectGUID[1] & 0xFF));
7    displayStr.append(prefixZeros((int) objectGUID[0] & 0xFF));
8    displayStr.append("-");
9    displayStr.append(prefixZeros((int) objectGUID[5] & 0xFF));
10   displayStr.append(prefixZeros((int) objectGUID[4] & 0xFF));
11   displayStr.append("-");
12   displayStr.append(prefixZeros((int) objectGUID[7] & 0xFF));
13   displayStr.append(prefixZeros((int) objectGUID[6] & 0xFF));
14   displayStr.append("-");
15   displayStr.append(prefixZeros((int) objectGUID[8] & 0xFF));
16   displayStr.append(prefixZeros((int) objectGUID[9] & 0xFF));
17   displayStr.append("-");
18   displayStr.append(prefixZeros((int) objectGUID[10] & 0xFF));
19   displayStr.append(prefixZeros((int) objectGUID[11] & 0xFF));
20   displayStr.append(prefixZeros((int) objectGUID[12] & 0xFF));
21   displayStr.append(prefixZeros((int) objectGUID[13] & 0xFF));
22   displayStr.append(prefixZeros((int) objectGUID[14] & 0xFF));
23   displayStr.append(prefixZeros((int) objectGUID[15] & 0xFF));
24
25   return displayStr.toString();
26 }
27
28 private static String prefixZeros(int value) {
29   if (value <= 0xF) {
30     StringBuilder sb = new StringBuilder("0");
31     sb.append(Integer.toHexString(value));
32     return sb.toString();
33   } else {
34     return Integer.toHexString(value);
35   }
36 }
```

Ukázka kódu 8: Konverze pole bytů na čitelný řetězec znaků atributu objectGUID.

6.4 Mapování objektů z Bakalářů

Pro vytváření objektů obvykle programátoři používají teleskopické konstruktory (angl. *telescoping constructors*), kdy se poskytne konstruktor pouze s požadovanými parametry, další pouze s jedním volitelným parametrem, třetí se dvěma volitelnými parametry a podobně. Končí se konstruktorem se všemi volitelnými parametry.[2]

[3] `http://www.developerscrappad.com/1109/windows/active-directory/java-ldap-jndi-2-ways-of-decoding-and-using-the-objectguid-from-windows-active-directory/`

Návrhový vzor teleskopického konstruktoru funguje, ale je obtížný pro psaní klientského kódu, kdy je přítomno příliš mnoho parametrů, a ještě obtížnější na čtení. Čtenář musí přemýšlet, co všechny parametry znamenají a musí je důkladně počítat. Dlouhé sekvence identických typů parametrů může způsobit neočekávané chyby. Když klient náhodou prohodí dva parametry, `compiler` si nebude stěžovat, ale program se bude chovat nepředvídatelně za běhu.[2]

Další alternativou, když se čelí problému s mnoha parametry konstruktoru, je vzor JavaBeans, kde se volá bezparametrický konstruktor k vytvoření objektu a následně se volají `setter` metody k nastavení každého požadovaného parametru a každého volitelného parametru, pokud je žádoucí ho nastavit. Tento vzor nemá nevýhody teleskopického konstruktoru, je jednoduchý na čtení koncového kódu, i když trochu mnohoslovný , výřečný.[2]

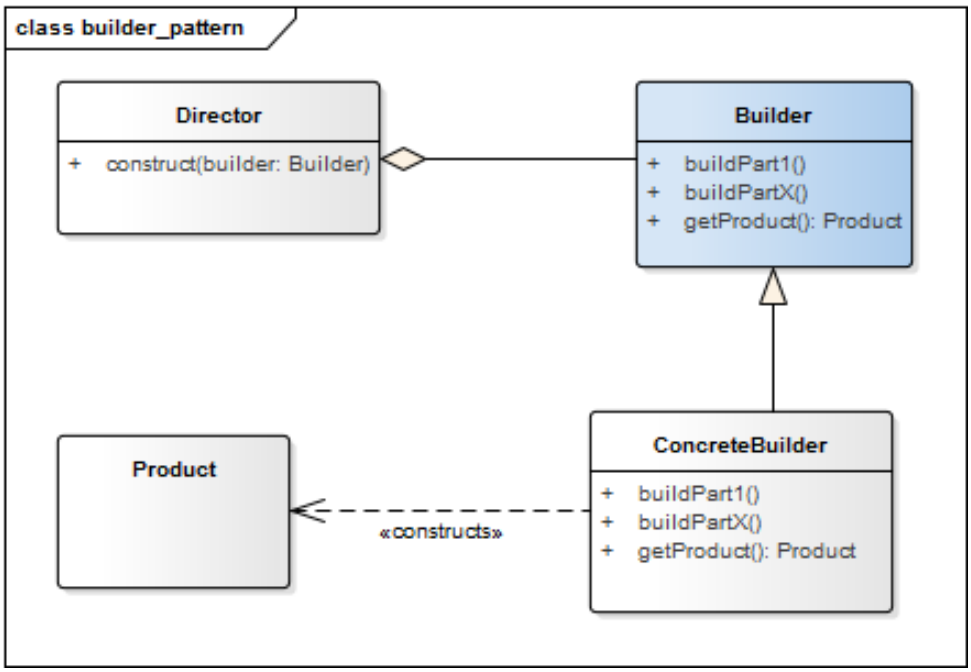
Naneštěstí JavaBeans vzor má vážné nevýhody sám o sobě. Protože konstrukce objektu je rozdělena přes několik volání, JavaBean může být během konstrukce v nekonzistentním stavu. Třída nemá volbu vynucení konzistence pouhou kontrolou validity parametrů konstruktoru. Pokusit se použít objekt v nekonzistentním stavu může způsobit chyby, které se sice dají poté odstranit z kódu, ale je obtížné je nalézt a ladit. Další nevýhodou je, že JavaBeans vzor přímo vylučuje možnost třídy být neměnitelnou (angl. *immutable*), vyžaduje přidanou snahu na straně programátora k zajištění *thread safety*.[2]

Naštěstí existuje třetí alternativa, která kombinuje bezpečnost vzoru teleskopického konstruktoru se snadnou čitelností vzoru JavaBeans. Jedná se o formu vzoru `Builder` (obrázek 6.2). Namísto přímého vytváření žádoucího objektu, klient volá konstruktor (nebo statickou továrnu) se všemi požadovanými parametry a získá objekt `builder`. Následně klient volá metody podobající se `setter` metodám na `builder` objektu k nastavení všech požadovaných parametrů. Nakonec klient volá bezparametrickou metodu `build` k vygenerování objektu, který je neměnný (*immutable*). `Builder` je statickým členem třídy, jejíž objekty vytváří. [2]

Vzor `Builder` (digram 6.2) poskytuje rozhraní pro vytváření částí tvořící `Product`, `ConcreteBuilder` je implementací tohoto rozhraní. `ConcreteBuilder` si zaznamenává reprezentaci, kterou vytváří, a zajišťuje možnost, jak získat výsledek (`Product`), stejně jako konstruovat produkt. `Director` konstruuje objekt skrz `Builder` rozhraní. Produkt je objekt, který se konstruuje. Často bývá velmi komplexní.[29]

Návrhový vzor `Builder` je použit mimo jiné k vytváření objektů při mapování databázových sloupců z aplikace Bakaláři na objekty v aplikaci INUM - viz ukázka kódu 9. Je použita jeho zjednodušená verze tak, jak ji navrhl Bloch v *Effective Java (2nd Edition) (The Java Series)*. Každá entita z databáze aplikace Bakaláři představuje v aplikaci INUM jednu třídu (např. BakalariPerson, BakalariGroup, BakalariClazz, BakalariLesson). Pomocí JDBCTemplate jsou načteny požadované záznamy z databáze a za využití rozhraní `RowMapper` (konkrétní implementací v ukázce kódu 9 je BakalariPersonMapper) se jednotlivé řádky záznamů návrhovým vzorem `Builder` transformují na příslušné Java objekty.

```
1 public class BakalariPerson implements Serializable {
```



Obrázek 6.2: Návrhový vzor Builder (inspirováno v [29]).

```
2       private String source;
3     private EPersonType personType;
4     private String surname;
5     // ...
6
7     public static class Builder {
8       private String source;
9       private EPersonType personType;
10      private String surname;
11      // ...
12
13      public Builder(String source, EPersonType personType) {
14        this.source = source;
15        this.personType = personType;
16      }
17
18      public Builder surname(String surname) {
19        this.surname = surname;
20        return this;
21      }
22
23      public BakalariPerson build() {
24        return new BakalariPerson(this);
25      }
26    }
27
28    private BakalariPerson(Builder builder) {
29      this.source = builder.source;
30      this.personType = builder.personType;
31      this.surname = builder.surname;
32      // ...
```

```
33    }
34  }
35
36  public class BakalariPersonMapper
37      extends GenericBakalariMapper implements RowMapper<BakalariPerson> {
38
39    private EPersonType personType;
40
41    @Override
42    public BakalariPerson mapRow(ResultSet resultSet, int rowNum)
43        throws SQLException {
44      init(resultSet);
45
46      return new BakalariPerson.Builder(getSource(), getPersonType())
47          .surname(getStringFromField("surname", resultSet))
48          // ...
49          .build();
50    }
51  }
```

Ukázka kódu 9: Návrhový vzor Builder při mapování atributů z Bakalářů.

6.5 PowerShell konzole

V sekci *5.2.2 Synchronizační služba pro Active Directory* byly uvedeny důvody, proč se vyplatí nadále používat PowerShell příkazy a skripty při synchronizaci mezi aplikací INUM a Active Directory.

Neustálé spouštění, provádění a ukončování jednotlivých oddělených PowerShell skriptů, jako tomu bylo v původní verzi aplikace na škole SZŠ Třebíč, způsobuje příliš velké zpoždění v rámci jednotek sekund. Navíc, protože volání není asynchronní, ale čeká se na dokončení provedení skriptu a uzavření konzole, čekací doba se promítne do času synchronizace již u jednoho objektu, nemluvě o kompletní synchronizaci všech objektů, pak se čekání sčítá.

Byla proto implementována služba, která se nazvala PowerShell konzole. V podstatě jde o proces operačního systému Windows, jehož životní cyklus je kontrolován z aplikace INUM za pomoci objektu třídy `Process`. Při prvním využití konzole, tj. při první synchronizaci, která vyžaduje volání PowerShell skriptů, se konzole nastartuje a žije až do ukončení celé aplikace, kdy se volá její ukončovací metoda. Tím pádem odpadá časově náročné spouštění při každém volání. Čas volání se tak smrskl prakticky na čas provedení samotného skriptu operačním systémem.

PowerShell konzoli se dají posílat jednotlivé příkazy, stejně jako celé soubory se skriptem, které se mají provést.

Protože využití PowerShell konzole není čistě limitované na aplikaci INUM, ale mohlo by být využitelné v kterémkoliv projektu pracujícím s prostředím Windows či Windows Server, byla konzole publikována na **GitLab** [4].

[4] `https://gitlab.com/davidak09/powershell-console`

V aplikaci INUM funguje konzole jako `Bean` v aplikačním kontextu. Je tak zaručeno, že bude vždy spuštěna maximálně jedna její instance (viz ukázka kódu 10).

```
@Bean(destroyMethod = "close")
@Lazy
public PowershellConsole powershellConsole() {
    PowershellConsole powershellConsole = PowershellConsole
        .getPowershellConsole();
    // needed for executing not-signed scripts
    powershellConsole.execute(
        "Set-ExecutionPolicy UnRestricted", "Set-ExecutionPolicy");
    return powershellConsole;
}

@Service
public class PowerShellService{

    @Autowired
    private PowershellConsole powershellConsole;

    public void executeExample(){
        // volani jednotliveho prikazu
        powershellConsole.invokeExpression("Get-Date", "Get-Date");

        // volani souboru se skriptem
        Path scriptPath = Paths.get("<CESTA_K_SOUBORU>");
        powershellConsole.invokeExpression(scriptPath, "muj ukazkovy kod");
    }
}
```

Ukázka kódu 10: PowerShell konzole.

Kapitola 7

Testování

7.1 Testování s uživateli

V rámci nasazení aplikace INUM proběhlo uživatelské testování ve 2 kolech, nejdříve s vybranými osobami, později se všemi aktivními učiteli. Testovala se převážně funkčnost procesu aktivace konta, tzn. učitelé zadávali své osobní emailové adresy, telefony a hlavně hesla. Většina testovaných uživatelů již měla funkční konta v integrovaných aplikacích, ale bez záruky stejného hesla, proto se účet nedal považovat za jednotný.

Důvod, proč byli zvoleni zrovna učitelé, je prostý. V kapitole *6 Implementace* je uvedeno, že v původním stavu komunikace mezi aplikacemi PowerShell skripty přenášely hesla z ručně generovaného výstupu aplikace Bakaláři. Tento soubor mohl být vygenerován pouze v době generování hesel k účtům, zpětně nešlo kvůli jednostranné hashovací funkci heslo získat v čitelné podobě. Administrátor pak vždy vygenerovaný soubor vložil do domluvené složky na serveru a skripty hesla dále zpracovávaly. Protože se některé soubory dochovaly i z historie, bylo možné naimportovat tato hesla do aplikace INUM pomocí dodatečně doprogramované utility. Soubory se také tiskly a dále předávaly uživatelů, aby věděli svůj login a heslo. Tento proces platil pro žáky a studenty, pro učitele žádný export neprobíhal, takže se ani nemohl zachovat soubor. Učitelé tak měli sice nastavena hesla, ale aplikace INUM o nich nevěděla. Proto je nově zavedený proces aktivace konta ten správný postup, jak hesla učitelů získat.

První kolo testování proběhlo na 2 vybraných učitelích a 2 studentech, kteří nově nastoupili do školy, jejich konta tak nebyla ještě vytvořena. Po drobných úpravách kódu aplikace a odchytání nedostatků úspěšně uživatelé aktivovali svá konta. Tito uživatelé simulují standardní použití procesu aktivace konta, neboť přesně stejným způsobem bude probíhat pro všechny nově příchozí osoby do organizace.

Druhé kolo testování bylo určeno pro všechny učitele, lépe řečeno zaměstnance školy, z pohledu Bakalářů jsou všichni umístěni v tabulce `ucitele`, liší se dále pracovní pozicí (učitel, školník, ředitel, administrativní pracovník apod.). Výsledek testování je zobrazen v tabulce 7.1. Všichni učitelé, kteří se testování zúčastnili, úspěšně proces aktivace konta dokončili, čímž aplikace

INUM získala jejich data. Účastnilo se sice pouze 63 ze 115 učitelů (cca 55%), které aplikace Bakaláři považuje za aktivní, to lze ale vysvětlit velkým počtem externistů na škole SZŠ ČB, kteří nevyužívají integrované aplikace. Každý aktivní učitel si procesem prošel, do té doby měl zablokované konto, takže bez něj mu není umožněn přístup do žádné aplikace.

testování	počet účastníků	počet úspěšných účastníků	celkem aktivních uživatelů
aktivace konta učitelů	63	63	115
aktivace konta žáků a studentů	2	2	621
změna hesla všech osob	30	30	735

Tabulka 7.1: Výsledky uživatelského testování.

Nutit všechny žáky a studenty projít procesem aktivace nedávalo smysl, proto byl určen pouze pro ty, jejichž hesla z určitých důvodů nebyla známá. Nakonec se tak testování týkalo pouze jednoho žáka a jednoho studenta z celkového počtu 621 aktivních, oba dva bez problémů aktivaci dokončili.

Dále se ještě testovala změna hesla osob. Zatím si změnilo (či zapomnělo a muselo změnit) heslo 30 uživatelů, z toho všichni, kteří proces započali, jej i úspěšně dokončili. Výsledek je vyhodnocen na základě přítomnosti 2 atributů v databázi INUM, první zaznamenává čas, kdy uživatel o změnu hesla zažádal, druhý pak kdy uživatel změnu potvrdil, tedy moment, ve který aplikace INUM považuje uživatelem nově zvolené heslo za platné a původní heslo za neplatné. Podobný princip 2 časových razítek je použit i při vyhodnocení aktivace konta.

7.2 Testování jednotek

V aplikace INUM lze identifikovat několik jednotek, jež je zapotřebí testovat. Mezi takové jednotky patří hlavně synchronizační služby, tedy služby pro synchronizaci s Active Directory, aplikací Bakaláři a Google Apps. Testování AD a aplikace Bakaláři bylo vysvětleno v bakalářské práci [17], Google Apps jsou nově integrované. Vzhledem ke komplexnosti všech tří aplikací je velmi složité je nahradit „mock objekty“ jako tomu bývá při klasickém unit testování. Proto neprobíhalo automatické testování jednotek, ale ruční.

Synchronizace s Bakaláři probíhá ve formě přímého přístupu do databáze, celou aplikaci tak není možné simulovat jinak než druhou testovací instancí nebo kopií databáze. Testovací instanci kvůli nedostatku další licence programu použít nelze, zbývá tak kopie databáze. Tento postup byl zvolen, plnohodnotně však nesimuluje běh aplikace a hlavně způsoby, jak se systém po přímém zápisu do databáze aplikací INUM zachová, zda nenastane neočekávaný stav, chyba či kompletní pád aplikace. Proto po fázi testování s kopií

databáze byla na velmi malém vzorku uživatelů testována synchronizace vybraných osob s produkční databází, aby případné změny zasáhly a ovlivnily co nejmenší část systému.

Pro Active Directory neexistují „mocky“ (pro LDAP ano, například JXplorer [1]), testování tak probíhalo nejdříve za využití virtuálního serveru s operačním systémem Windows Server nastaveným na doménový kontroler fiktivní domény, co nejvíce odpovídající nastavením doméně v produkčním prostředí. Do testování zasahuje i exekuce PowerShell skriptů, které nelze pustit v jiném prostředí než Windows. Po této fázi přišel na řadu režim testování v produkčním prostředí, opět na skupině vybraných uživatelů. Vytvářely se jim účty přesně odpovídající reálnému účtu s jedinou změnou - přidáním přípony `_test` za login. Účty se soustředily do speciální organizační jednotky (OU), aby se nepletly s produkčními daty.

Obdobně probíhalo testování jednotky pro synchronizaci s Google Apps. I zde chybí „mock“, je proto daleko lepší simulovat požadované chování na malém počtu objektů přímo v produkčním prostředí, než se složitě prostředí snažit nasimulovat. Pro Google Apps byly použity již zmíněné účty 2 učitelů z testování s uživateli. Nejdříve se vytvářely účty v doméně GA s příponou `_test`, následně přímo finální účty a jejich změny - změna hesla, zneplatnění, změna jména apod. Až po důkladném zkoušení se aplikaci INUM povolilo spravovat všechny účty.

7.3 Logování

Jeden z využitých způsobů testování při implementaci a nasazení aplikace INUM představuje i zkoumání a analyzování logů a chybových zpráv aplikace. V případě pádu, chyby nebo výjimky aplikace zasílá emailem na definované adresy úryvek posledních záznamů v logu a zároveň chybu, která vše způsobila.

Zároveň aplikace udržuje vlastní log v databázi, kde se uchovávají hlášení definovaná programátorem. Nejčastěji se jedná o zprávy o průběhu synchronizace, jako například kolik záznamů bylo synchronizováno, které úspěšně, které neúspěšně a případně, proč se synchronizace nezdařila.

[1] `http://jxplorer.org/downloads/jndimock.html`

Kapitola 8

Závěr

Zadání diplomové práce vytyčilo dva hlavní cíle. Prvním cílem bylo navrhnout, implementovat a nasadit aplikaci, která zajistí fungovaní jednotného uživatelského konta v prostředí střední školy. Druhým cílem bylo vybrat a nasadit jednotnou ověřovací službu, která bude pracovat na principu Single Sign-On (SSO) a bude v rámci školní organizace působit jako poskytovatel identity.

První cíl byl úspěšně splněn nasazením nástroje INUM, který byl vytvořen autorem této práce. INUM zajišťuje integraci aplikací Bakaláři, Active Directory, Moodle a Google Apps. Napojení Google Apps bylo jedním z podřazených úkolů první části. INUM přenáší převážně data o uživatelích mezi integrovanými aplikacemi a udržuje tak konzistenci a integritu uživatelského konta, tj. hlavně jeho přihlašovacího jména a hesla. Uživatel udržuje pouze jedny přihlašovací údaje a je schopen se jejich pomocí přihlašovat do zmíněných aplikací.

INUM je webová aplikace napsaná v Java frameworku Spring a funguje jako integrační architektura Hub-and-Spoke (viz kapitola *3 Analýza systémové integrace*). Dovoluje uživatelům přes webové rozhraní spravovat své údaje, například heslo, email a jiné. Data následně distribuuje do všech aplikací, ve kterých má podle definovaných podmínek konto existovat. INUM nesynchronizuje jen uživatelská data, ale i informace o skupinách, třídách, rozvrhu hodin a další. Tyto informace jsou získány z aplikace Bakaláři, která tvoří primární zdroj dat.

Byly definovány a zavedeny procesy jako aktivace a deaktivace uživatelského účtu, změna hesla a podobně, napomáhající aplikaci INUM, administrátorům i koncovým uživatelům řídit životní cyklus uživatelských účtů a dalších dat v prostředí.

Druhý cíl byl splněn nasazením federační služby ADFS. Poskytovateli identity, ADFS serveru, důvěřují a jsou nastaveny jako předávající strany (angl. *relying parties*) aplikace INUM, Google Apps, Moodle i Active Directory. Aplikaci Bakaláři z vysvětlených důvodů není možné do SSO zapojit.

Po přihlášení z kterékoliv aplikace k poskytovateli identity přes webový formulář není po definovanou dobu po uživateli dále požadováno opětovné zadávání přihlašovacích údajů. Místo toho je považován za autentizovaného a za pomoci výměny SAML zpráv mezi poskytovatelem identity a příslušnou

předávající stranou je automaticky přihlášen i do dalších aplikací, pokud k nim požaduje přístup.

Nasazené SSO řešení využívá jednotný uživatelský účet z prvního cíle, protože ověřuje uživatelské údaje vůči informacím obsaženým v doméně Active Directory, o jejichž synchronizaci se stará integrační nástroj INUM.

Jedním z úkolů, z části patřící pod oba cíle, bylo zavést možnost autentizace uživatelů ke správě jednotného účtu pomocí účtů třetích stran. Byli zvoleni tři poskytovatelé tzv. Social Login na základě statistik o jejich použití - Facebook, Google a Twitter. Všechny tři služby používají k autorizaci i autentizaci protokol OAuth. Uživatel nejdříve ve webové správě svého účtu autorizuje aplikaci INUM k provádění akcí u poskytovatele služby. To znamená, že propojí svůj účet v organizaci se svým osobním účtem u aplikace třetí strany, a následně je mu umožněno se ke správě svého účtu přihlašovat použitím identity účtu třetí strany.

Autor práce se seznámil s integračními principy a architekturami důležitými pro návrh integrace systémů a zároveň prozkoumal důkladně zejména protokol LDAP a API pro Google Apps, díky kterým probíhá synchronizace s Active Directory, respektive Google Apps. Při návrhu SSO řešení si autor osvojil federační, autentizační a autorizační mechanismy, prozkoumal nabízená řešení a podrobně probádal protokol OAuth a jazyk SAML.

V dalším vývoji aplikace INUM by bylo vhodné rozvinout potenciál propojení s účty třetích stran, v současné době se využívají pouze pro autentizační účely. Bez větších potíží by šlo například k jednotnému účtu připojit foto ze sociální sítě nebo třeba zakládat facebookové skupiny pro jednotlivé třídy studentů, které by mohly podpořit soudržnost a spolupráci mezi žáky a studenty.

Rozvoj by mohl pokračovat nejen na samotné aplikaci, ale i jejím rozšířením na další školy, pokud by byly schopné akceptovat navržené procesy, zejména vzniku a zániku uživatelského účtu.

Literatura

[1] C. A. Binildas, *Service Oriented Java Business Integration*, 1. vyd. Packt Publishing Ltd., 2008, ISBN: 978-1-847194-40-4.

[2] J. Bloch, *Effective Java (2nd Edition) (The Java Series)*, 2. vyd. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2008, ISBN: 0321356683, 9780321356680.

[3] P. Bouška. (14. zář. 2007). Adresářové služby a LDAP, WWW: `http://www.samuraj-cz.com/clanek/adresarove-sluzby-a-ldap/` (cit. 16. 03. 2016).

[4] R. Boyd, *Getting Started with OAuth 2.0*, 1. vyd. O'Reilly Media, Inc., 2012, ISBN: 978-1-449-31160-5.

[5] P. A. Cabarcos, F. A. Mendoza, A. Marín-López a D. Díaz-Sánchez, „Wireless and Mobile Networking: Second IFIP WG 6.8 Joint Conference, WMNC 2009, Gdańsk, Poland, September 9-11, 2009. Proceedings“, in, J. Wozniak, J. Konorski, R. Katulski a A. R. Pach, ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, kap. Enabling SAML for Dynamic Identity Federation Management, s. 173–184, ISBN: 978-3-642-03841-9. DOI: `10.1007/978-3-642-03841-9_16`. WWW: `http://dx.doi.org/10.1007/978-3-642-03841-9_16`.

[6] N. Conner, *Google Apps: The Missing Manual*, 1. vyd. O'Reilly Media, Inc., 2008, ISBN: 978-0-596-51579-9.

[7] Dana Gardner. (28. čvc 2009). Don't use an ESB unless you absolutely, positively need one, Mule CTO warns, WWW: `http://www.zdnet.com/article/dont-use-an-esb-unless-you-absolutely-positively-need-one-mule-cto-warns/` (cit. 19. 04. 2016).

[8] Z. Dennis. (9. květ. 2013). Choosing an SSO Strategy: SAML vs OAuth2, WWW: `https://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/` (cit. 19. 04. 2016).

[9] B. Desmond, J. Richards, R. Allen a A. G. Lowe-Norris, *Active Directory*, 5. vyd. O'Reilly Media, Inc., 2013, ISBN: 9781449320027.

[10] M. Fischer, J. Partner, M. Bogoevici a I. Fuld, *Spring Integration in Action*. Manning Publications Co., 2012, ISBN: 9781935182436.

[11] Google Developers. (22. ún. 2016). GData API Directory, WWW: https://developers.google.com/gdata/docs/directory (cit. 07. 04. 2016).

[12] Google Inc. (23. břez. 2015). Final Reminder: Deprecated Google Apps Admin APIs to be discontinued on April 20, 2015, WWW: http://googleappsupdates.blogspot.cz/2015/03/final-reminder-deprecated-google-apps.html (cit. 24. 03. 2016).

[13] Google Inc. (24. břez. 2016). Google Apps For Education, WWW: https://www.google.com/edu/products/productivity-tools/ (cit. 24. 03. 2016).

[14] D. R. Herrick, „Google This!: Using Google Apps for Collaboration and Productivity“, in *Proceedings of the 37th Annual ACM SIGUCCS Fall Conference: Communication and Collaboration*, ř. SIGUCCS '09, St. Louis, Missouri, USA: ACM, 2009, s. 55–64, ISBN: 978-1-60558-477-5. DOI: 10.1145/1629501.1629513. WWW: http://doi.acm.org/10.1145/1629501.1629513.

[15] G. Hohpe a B. Woolf, *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Pearson Education, Inc., 2004, ISBN: 0-321-20068-3.

[16] InfoSec Institute. (30. zář. 2014). SAML, OAuth, OpenID, WWW: http://resources.infosecinstitute.com/saml-oauth-openid/ (cit. 25. 04. 2016).

[17] D. Janíček, „Nástroj pro integraci uživatelských účtů“, bakalářská práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, květ. 2014.

[18] L. Jelínek, *Školní agenda*, Projekt OP VK „Inovace studijních oborů zajišťovaných katedrami PřF UHK“, Univerzita Hradec Králové, 2014. WWW: https://www.bakalari.cz/ucebnice/zakladniPrirucka.pdf.

[19] G. Kaur a D. Aggarwal, „A Survey Paper on Social Sign-On Protocol OAuth 2.0“, *Journal of Engineering, Computers & Applied Sciences (JEC&AS)*, sv. 2, č. 6, s. 93–96, 2013.

[20] S. Koussa. (16. čvc 2013). Federated Identities: OpenID vs SAML vs OAuth, WWW: http://www.softwaresecured.com/2013/07/16/federated-identities-openid-vs-saml-vs-oauth/ (cit. 25. 04. 2016).

[21] A. Larralde. (3. dub. 2015). Social Login Trends Across the Web: Q1 2015, WWW: http://janrain.com/blog/social-login-trends-across-the-web-q1-2015/ (cit. 29. 04. 2016).

[22] K. D. Lewis a J. E. Lewis, „Web Single Sign-On Authentication using SAML“, *CoRR*, sv. abs/0909.2368, 2009. WWW: http://arxiv.org/abs/0909.2368.

[23] Microsoft, *Active Directory Cmdlets in Windows PowerShell*, 2016. WWW: https://technet.microsoft.com/en-us/library/ee617195.aspx (cit. 16. 03. 2016).

[24] Moodle Project. (12. ún. 2016). About Moodle, WWW: https://docs.moodle.org/30/en/About_Moodle (cit. 23. 03. 2016).

[25] L. Palíšek, „Integrace Google Apps“, bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Katedra počítačových systémů, květ. 2013.

[26] V. Radha a D. H. Reddy, „A Survey on Single Sign-On Techniques“, *Procedia Technology*, sv. 4, s. 134–139, 2012, 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012, ISSN: 2212-0173. DOI: http://dx.doi.org/10.1016/j.protcy.2012.05.019. WWW: http://www.sciencedirect.com/science/article/pii/S2212017312002988.

[27] sqlexpress. (21. dub. 2010). SQL Server 2008 R2 Express Database Size Limit Increased to 10GB, WWW: https://blogs.msdn.microsoft.com/sqlexpress/2010/04/21/sql-server-2008-r2-express-database-size-limit-increased-to-10gb/ (cit. 15. 04. 2016).

[28] J. Sugrue. (9. břez. 2010). Visitor Pattern Tutorial with Java Examples, WWW: https://dzone.com/articles/design-patterns-visitor (cit. 02. 05. 2016).

[29] J. Sugrue. (15. červ. 2010). Builder Pattern Tutorial with Java Examples, WWW: https://dzone.com/articles/design-patterns-visitor (cit. 02. 05. 2016).

[30] S.-T. Sun, „Simple But Not Secure: An Empirical Security Analysis of OAuth 2.0-Based Single Sign-On Systems“, 2012. WWW: http://blogs.ubc.ca/computersecurity/files/2012/04/San-Tsai.pdf.

[31] S. Tuttle, A. Ehlenberger, R. Gorthi, J. Leiserson, R. Macbeth, N. Owen, S. Ranahandola, M. Storrs a C. Yang, *Understanding LDAP Design and Implementation*, 2. vyd. IBM, 2004.

[32] Úřad pro ochranu osobních údajů. (23. čvc 2014). Je rodné číslo v ČR citlivým údajem podle § 4 písm. b) zákona o ochraně osobních údajů? Zodpovídá: PhDr. David Pavlát, WWW: https://www.uoou.cz/je-rodne-cislo-v-cr-citlivym-udajem-podle-4-pism-b-zakona-o-ochrane-osobnich-udaju/d-11350/p1=2619 (cit. 28. 04. 2016).

Seznam zkratek

AD Active Directory. viii, 3–5, 8, 10–12, 19, 27, 28, 31, 33, 49, 51, 52, 54, 55, 57, 61, 63–65, 69, 71, 73, 77, 80, 81, 83, 84

ADFS Active Directory Federation Services. 4, 63–67, 83

API Application Programming Interface. viii, 3, 4, 6, 7, 16–20, 27, 28, 35, 36, 50–53, 55, 69, 84

CSV Comma Separated Values. 4

DBF dBase database file. 4

DIT Directory Information Tree. 10, 11, 13

DN Distinguished Name. 11, 12

GA Google Apps. viii, 2, 3, 6, 7, 13–19, 27, 28, 31, 35, 51, 52, 54, 55, 57, 61, 63, 66, 67, 69, 71, 80, 81, 83, 84

HTTP Hypertext Transfer Protocol. 16, 17, 19, 25, 38

IdP Identity Provider. viii, 36, 37, 39, 40, 64, 67

INUM Nástroj pro integraci uživatelských účtů (Integrated User Manager) [17]. vii, viii, 1, 4–7, 10, 16, 18, 19, 21, 27, 31, 32, 35, 36, 39–41, 45, 46, 49, 51–63, 66, 67, 69–75, 77–81, 83, 84

JDBC Java Database Connectivity. 53

JSON JavaScript Object Notation. viii, 16, 17, 19, 72

LDAP Lightweight Directory Access Protocol. viii, 3, 5, 8–13, 33, 54, 73, 81, 84

MSSQL Microsoft SQL Server. 4, 53

OAuth Open standard for authorization. vii, 16, 20, 33, 36, 37, 43, 45, 46, 58, 67, 69, 84

RDN Relative Distinguished Name. viii, 11, 12

REST Representational State Transfer. 6, 16, 18

RFC Request for Comments. 5, 8

RP Relying Party. 64–66

RPC Remote Procedure Call. 25, 27

SAML Security Assertion Markup Language. viii, 33, 35–41, 63–66, 69, 83, 84

SQL Structured Query Language. 53, 63

SSL Secure Sockets Layer. 15

SSO Single Sign-On. vii, 2, 5, 35, 36, 38, 39, 54, 57, 58, 63–67, 83, 84

SZŠ Třebíč Vyšší odborná škola a Střední škola veterinární, zemědělská a zdravotnická Třebíč. 1, 7, 77

SZŠ ČB Střední zdravotnická škola a Vyšší odborná škola zdravotnická České Budějovice. 1, 3–7, 16, 18, 27, 31, 35, 39, 51, 53, 55, 63, 66, 69, 80

URL Uniform Resource Locator. 19, 51

XML eXtensible Markup Language. viii, 19, 22, 30, 37–39, 41

Příloha A

Příklad SAML odpovědi od IdP

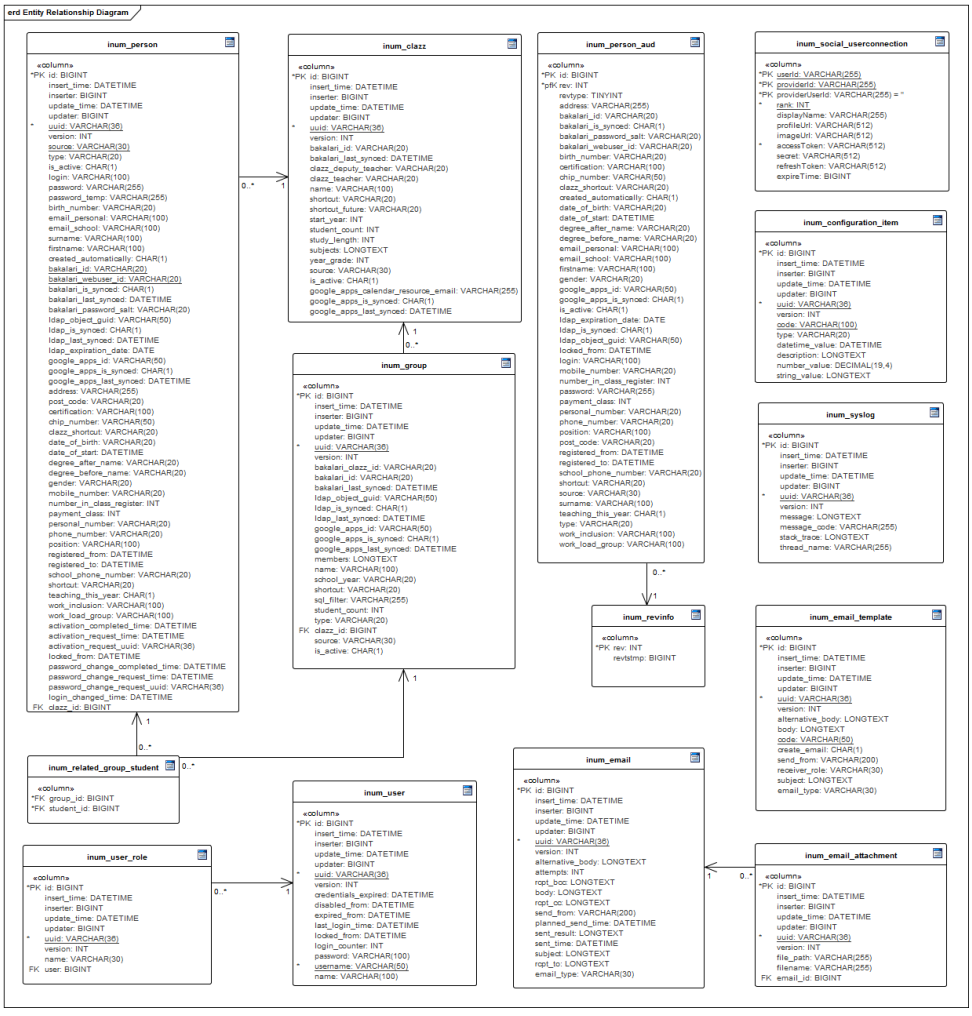
```
<?xml version="1.0" encoding="UTF-8"?>
<samlp:Response
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
Destination="https://davidak.inum.cz/saml/SSO"
ID="id..."
InResponseTo="response_to..."
IssueInstant="2016-04-21T12:47:56.650Z" Version="2.0">
  <Issuer xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
    http://idp.szscb.cz/adfs/services/trust
  </Issuer>
  <samlp:Status>
    <samlp:StatusCode
    Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
  </samlp:Status>
  <EncryptedAssertion
  xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
    <xenc:EncryptedData
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    Type="http://www.w3.org/2001/04/xmlenc#Element">
      <xenc:EncryptionMethod
      Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
      <KeyInfo
      xmlns="http://www.w3.org/2000/09/xmldsig#">
        <e:EncryptedKey
        xmlns:e="http://www.w3.org/2001/04/xmlenc#">
          <e:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
            <DigestMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
          </e:EncryptionMethod>
          <KeyInfo>
            <ds:X509Data
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
              <ds:X509IssuerSerial>
                <ds:X509IssuerName>
                    CN=mujucet.szscb.cz
                </ds:X509IssuerName>
                <ds:X509SerialNumber>
                    seriove cislo...
                </ds:X509SerialNumber>
              </ds:X509IssuerSerial>
```

```
43              </ds:X509Data>
44            </KeyInfo>
45            <e:CipherData>
46              <e:CipherValue>hodnota sifry...</e:CipherValue>
47            </e:CipherData>
48          </e:EncryptedKey>
49        </KeyInfo>
50      </xenc:EncryptedData>
51    </EncryptedAssertion>
52  </samlp:Response>
```

Ukázka kódu 11: SAML odpověď od ADFS IdP aplikaci INUM.

Příloha B

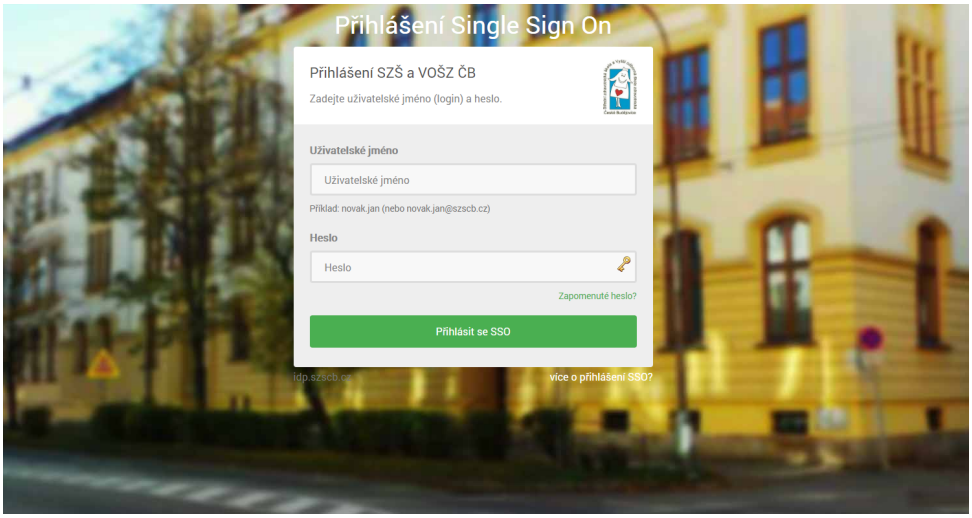
ER diagram



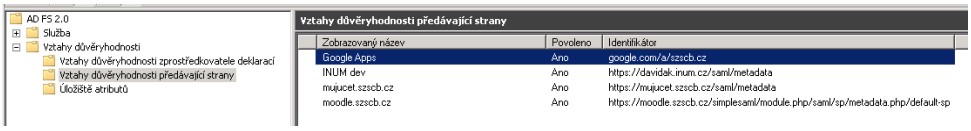
Obrázek B.1: Diagram atributů a závislostí databázových entit aplikace INUM.

Příloha C

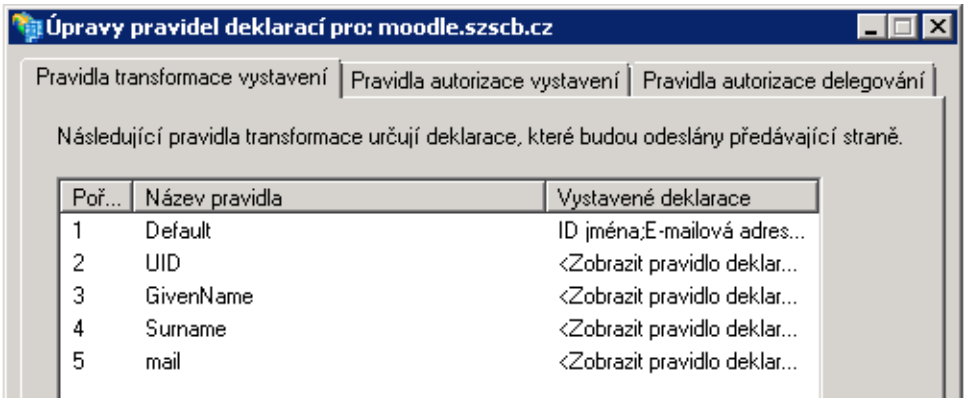
Konfigurace ADFS



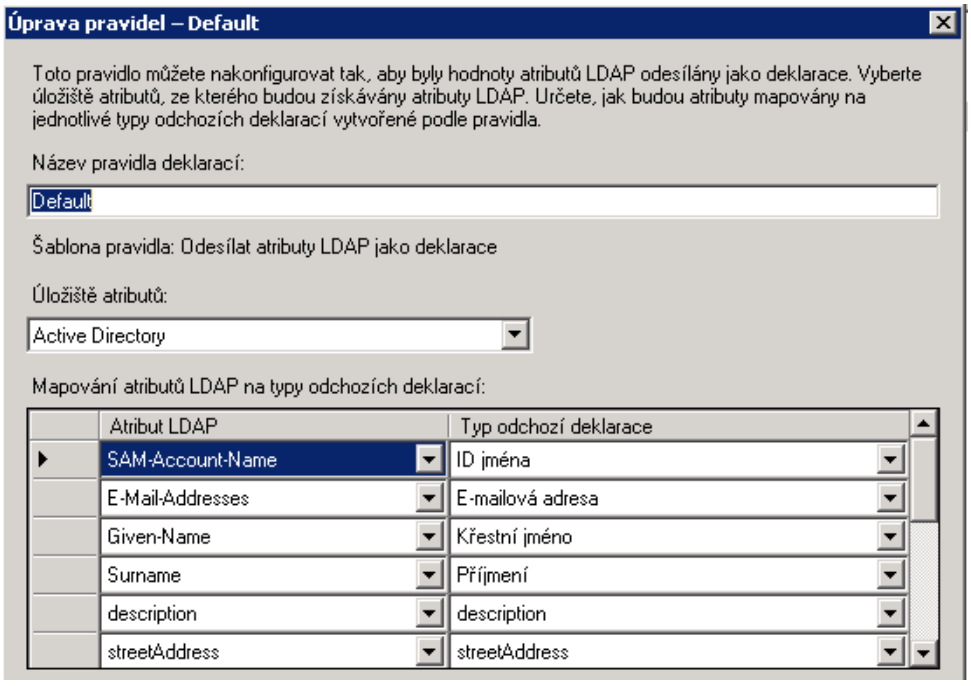
Obrázek C.1: Navržený vzhled přihlašovací stránky SSO ADFS federačního proxy serveru.



Obrázek C.2: Snímek obrazovky obecného nastavení důvěryhodných předávajících stran v ADFS serveru.



Obrázek C.3: Snímek obrazovky nastavení pravidel deklarací pro důvěryhodnou předávající stranu, v tomto případě aplikaci Moodle, respektive SimpleSAMLphp.



Obrázek C.4: Snímek obrazovky úpravy konkrétního SAML deklaračního pravidla v ADFS serveru.

Příloha D

Konfigurace SSO v INUM

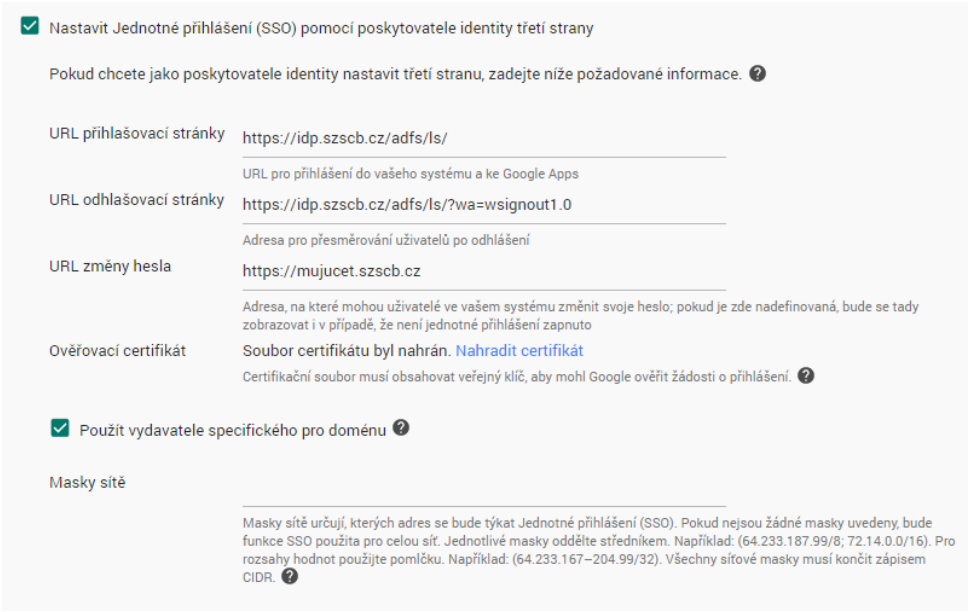
```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    // konfigurace prihlaseni k~administraci aplikace
  @Configuration
  @Order(1)
  public static class AdminSecurityConfig extends
      WebSecurityConfigurerAdapter {

      @Override
      protected void configure(HttpSecurity http) throws Exception {
        // ... predchozi volani

        // do administrace muze pristoupit jen uzivatel s~roli administratora
        http
          .antMatcher("/admin/**")
          .authorizeRequests()
          .antMatchers("/admin/**")
          .hasAnyRole(EUserRoleType.ROLE_ADMIN.getShort())
          .antMatchers("/**")
          .denyAll();
      // ... pokracovani
    }
    // dalsi metody konfigurace
  }

    // konfigurace prihlaseni k~uzivatelskemu uctu
  @Configuration
  @Order(2)
  public static class AccountSecurityConfig extends
      InumSAMLWebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
      // ... predchozi volani
      // SAML SSO konfigurace
      samlizedConfig(http);

      http
          .authorizeRequests()
          .antMatchers(
```

```
43                  "/admin/**"
44              )
45              .denyAll()
46              .antMatchers(
47                  "/ucet",
48                  "/ucet/**"
49              )
50              .hasAnyRole(EUserRoleType.ROLE_USER.getShort())
51              .antMatchers("/**")
52              .permitAll()
53
54              // standardni login a~heslo
55              .and()
56              .formLogin()
57              .failureHandler(authenticationFailureHandler())
58              .successHandler(authenticationSuccessHandler)
59              .loginPage("/signin")
60              .loginProcessingUrl("/signin/authenticate")
61              .failureUrl("/signin?failed=1")
62              .usernameParameter("j_username")
63              .passwordParameter("j_password")
64              .permitAll()
65
66              // prihlaseni pomoci uctu treti strany
67              .and()
68              .apply(new SpringSocialConfigurer()
69                  .postLoginUrl("/ucet")
70                  .alwaysUsePostLoginUrl(true)
71                  .defaultFailureUrl("/signin?social_failed=1")
72                  .signupUrl("/signin?social_failed=1")
73              );
74      }
75
76      // Nacteni konfigurace SAML z~databaze INUM
77      @Override
78      protected SAMLConfigBean samlConfigBean() {
79        return new SAMLConfigBeanBuilder()
80              .setIdpServerName(ConfigurationSAML.getIdpServerName())
81              .setSpServerName(ConfigurationSAML.getSpServerName())
82              .setKeystoreResource(new DefaultResourceLoader()
83                  .getResource(ConfigurationSAML.getKeystoreResource()))
84              .setKeystorePassword(ConfigurationSAML.getKeystorePassword())
85              .setKeystoreAlias(ConfigurationSAML.getKeystoreAlias())
86              .setKeystorePrivateKeyPassword(
87                  ConfigurationSAML.getKeystorePrivateKeyPassword())
88              .setSuccessLoginDefaultUrl(
89                  ConfigurationSAML.getSuccessLoginDefaultUrl())
90              .setSuccessLogoutUrl(ConfigurationSAML.getSuccessLogoutUrl())
91              .setSamlUserDetailsService(personDao)
92              .createSAMLConfigBean();
93      }
94
95      // dalsi metody konfigurace
96    }
97  }
```

Ukázka kódu 12: Část kódu zabezpečení aplikace INUM přes Spring Security.

Příloha E

Konfigurace SSO v Google Apps



Obrázek E.1: Snímek obrazovky konfigurace SSO v Google Apps.

Příloha F

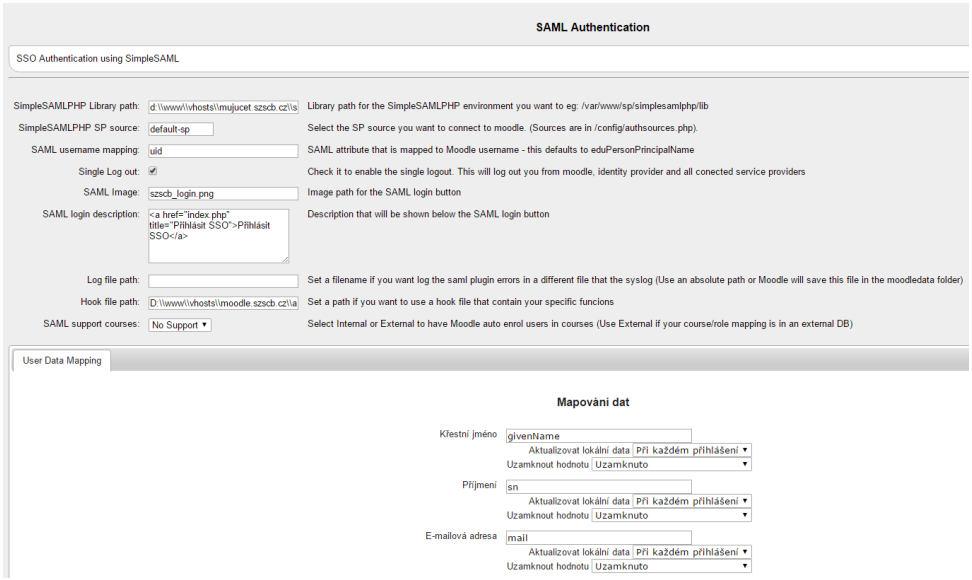
Konfigurace SSO v Moodle

```
// SimpleSAMLphp - Cast konfigurace souboru config/authsources.php
<?php
'default-sp' => array(
    'saml:SP',
    'privatekey' => 'saml.pem',
    'certificate' => 'saml.crt',
    'NameIDPolicy' => null,

    // The entity ID of this SP.
    // Can be NULL/unset, in which case an entity ID
    // is generated based on the metadata URL.
    'entityID' => null,

    // The entity ID of the IdP this should SP should contact.
    // Can be NULL/unset, in which case the user will
    // be shown a list of available IdPs.
    'idp' => 'http://idp.szscb.cz/adfs/services/trust',

    // The URL to the discovery service.
    // Can be NULL/unset, in which case
a// builtin discovery service will be used.
    'discoURL' => null,
    // ...
)
// ...
```

Ukázka kódu 13: Úryvek z konfigurace SimpleSAMLphp.

```
// SimpleSAMLphp - Cast konfigurace souboru metadata/saml20-idp-remote.php
<?php
\$metadata['http://idp.szscb.cz/adfs/services/trust'] = array(
        'name' => array(
                'en' => 'ADFS IdP',
        ),
        'SingleSignOnService'  =>
        'https://idp.szscb.cz/adfs/ls',
        'SingleLogoutService'  =>
        'https://idp.szscb.cz/adfs/ls/?wa=wsignout1.0&
            wreply=https://idp.szscb.cz/adfs/ls/?wa=wsignoutcleanup1.0',
        'certFingerprint'      =>
        '123456789abcdefghijklmnopqrstuvwxyz0123456789'
);
```

Ukázka kódu 14: Konfigurace poskytovatele identity v SimpleSAMLphp.



Obrázek F.1: Snímek obrazovky konfigurace modulu pro SAML v aplikaci Moodle.

Příloha G

Struktura přiloženého CD

/
- `ea` ..... soubory z Enterprise Architect
- `sso` ..... nastavení SSO v jednotlivých integrovaných aplikacích
  - `adfs` ..... nastavení SSO v ADFS
  - `ga` ..... nastavení SSO v Google Apps
  - `inum` ..... nastavení SSO v INUM
  - `moodle` ..... nastavení SSO v Moodle (simpleSAMLphp)
- `text` ..... složka s textem diplomové práce
  - `latex` ..... složka se zdrojovými soubory ve formátu LaTeX
  - `DP_Janicek_David_2016.pdf` ..... text práce ve formátu PDF
- `tool` ..... složka s implementovaným nástrojem
  - `inum` ..... zdrojové kódy integrační aplikace
  - `install.txt` ..... návod na instalaci nástroje