TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

BAKALÁŘSKÁ PRÁCE

Liberec 2012 **Václav Bohata**

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie

Studijní obor: 1802R007 – Informační technologie

Webová aplikace pro centralizovanou správu uživatelských účtů

Web application for centralized management of user accounts

Bakalářská práce

Autor: Václav Bohata

Vedoucí práce: Mgr. Jiří Vraný, Ph. D.

V Liberci 15. 5. 2012

ZDE BUDE ORIGINÁL ZADÁNÍ JAKO STRANA Č. 2 (číslování se ale neuvádí)

1. Seznamte se s běžně používanými databázemi uživatelských účtů a existujícími nástroji pro jejich centralizovanou správu.

2. Navrhněte webovou aplikaci pro centrální správu uživatelských účtů, která bude schopna tyto databáze spravovat.

3. Návrh prakticky implementujte s využitím platformy LAMP.

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Touto cestou bych chtěl poděkovat mému vedoucímu práce Mgr. Jiřímu Vranému, Ph. D., za užitečné rady pro tvorbu mé bakalářské práce.

Abstrakt

Bakalářská práce popisuje návrh a implementaci aplikace sloužící pro centralizovanou správu uživatelských účtů. Úvod práce je věnován databázím, které se v současné době běžně používají jako úložiště především pro větší množství uživatelských účtů. Popisuje jejich vlastnosti a zabývá se jejich správou především z pohledu uživatelských účtů. Uvádí existující nástroje a postupy pro centrální správu uživatelů v těchto databázích. Hlavní část práce se zabývá návrhem aplikace pro centrální správu uživatelů. Závěrečná část práce je věnována implementaci aplikace pro centrální správu uživatelských účtů.

Klíčová slova

Identity Management, správa uživatelských účtů, koncový systém, centrální databáze, synchronizace uživatelů

Abstract

The thesis describes the design and implementation of application capable of centralized management of user accounts. The first part of the thesis describes characteristics of databases that are commonly used as a repository for multiple user accounts and describes their capability for remote management. It also lists existing tools and procedures for remote management of these databases. The design of application capable of centralized management of user accounts is described in the main part of the thesis. The implementation of design is described in the last part of the thesis.

Keywords

Identity Management, management of user accounts, end system, central database, synchronization of users

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů, zkratek a termínů

SŘBD – systém řízení báze dat

SQL – Structured Query Language

LAMP – Linux, Apache, Mysql, PHP

OID – object identifier

DN – distinguished name

RDN – relative distinguished name

LDIF – LDAP Data Interchange Format

DSML – Directory Services Markup Language

NOS – Network Operating System

OU – organizační jednotka

IdM – Identity Management

HR – Human resources

RBAC – Role-Based Access Control

Web API – webové aplikační rozhraní

HTTP – Hypertext Transfer Protocol

SSL – Secure Socket Layer

HTTPS – Hypertext Transfer Protocol Secure

ACL – Access Control List

PHP – hypertext preprocessor

EAV – Entity attribute value

CGI – Common Gateway Interface

CSS – Cascading Style Sheets

PECL – The PHP Extension Community Library

CSRF – Cross-site request forgery

1 Úvod

1.1 Úvod do problematiky

Počítačové systémy jsou nedílnou součástí současného civilizovaného světa. Mnoho lidí se dnes neobejde bez osobního počítače či chytrého telefonu. Elektronická pošta, elektronické obchody či připojení do firemní sítě aj. služby denně využívá vysoké procento populace.

Většina těchto služeb vyžaduje autentizaci (ověření identity) uživatelů a málokdy jsou všechny propojeny do jednoho systému. Uživatelé se převážně autentizují uživatelským jménem a heslem. Musejí si pamatovat několik hesel, a proto často volí do různých služeb stejné, převážně snadno odhadnutelné heslo. Při jeho změně lze snadno zapomenout změnit ho v nějaké z mnoha služeb, které uživatel používá.

Z tohoto důvodu se používají centrální databáze uživatelských účtů společné pro několik služeb. Na webu se často pro autentizaci používá třetí strana, která výsledek autentizace poskytuje samotné službě, aniž by služba znala uživatelské přihlašovací údaje. Tyto mechanismy sice poskytují elegantní řešení pro ověřování uživatelů, v praxi je ale občas potřeba uchovávat uživatelské údaje pro každou službu zvlášť, např. z důvodu možné nedostupnosti centrální databáze. Někteří uživatelé mohou z různých důvodů (např. kvůli zvýšení bezpečnosti) požadovat možnost nastavit různým službám různá hesla a centrální ověřovací mechanismy toto většinou neumožňují.

Řešením tohoto problému je používat pro každou službu vlastní databázi uživatelských účtů a tyto databáze nějakým způsobem propojit alespoň tak, aby si uživatel mohl měnit své heslo ve všech databázích najednou nebo si mohl vybrat, pro které databáze bude používat hesla stejná. Propojení databází je potřeba řešit nejen na uživatelské úrovni (změna hesla), ale i na úrovni správcovské (centrální administrace propojených databází).

1.2 Cíl práce

Cílem této práce bylo vytvořit webovou aplikaci pro centrální správu databází uživatelských účtů. Aplikace obsahuje svou vlastní hlavní databázi uživatelů. Uživatel

je spravován ve všech spravovaných databázích na základě shody jeho uživatelského jména s hlavní databází. Aplikace předpokládá, že všechny spravované databáze jsou používány pro služby, které autentizují uživatele pomocí uživatelského jména a hesla.

Všechny propojené uživatelské účty ve všech databázích (vč. hlavní databáze aplikace) jsou spravovány administrátorem přes webové rozhraní. Webové rozhraní je optimalizováno pro rychlou hromadnou správu většího množství uživatelů. Funkčnost webového rozhraní lze snadno rozšířit pomocí modulů. Součástí webové aplikace je i aplikační rozhraní pro komunikaci s uživatelskou částí aplikace. Uživatelská část aplikace umožňuje uživatelům změnu hesla ve vybraných databázích.

2 Běžné databáze uživatelských účtů

Jako úložiště uživatelských účtů se používají nejrůznější databáze – od prostých textových souborů až po komplexní adresářové systémy. Tato kapitola popisuje komplexnější databáze, které jsou v současnosti běžně používané pro ukládání většího množství uživatelských účtů. Zabývá se jejich architekturou a vlastnostmi.

2.1 Relační databáze

Některé aplikace používají pro ukládání uživatelských účtů relační databázi. Často se lze s tímto úložištěm setkat u aplikací webových, kde nalezneme uživatelské účty pohromadě s ostatními daty. Relační databázi je vhodné využívat jako úložiště uživatelských účtů tam, kde se používá neměnná struktura dat nebo kde je potřeba využívat některé specifické vlastnosti většiny používaných relačních databází (např. transakce). Z bezpečnostního hlediska obvykle není vhodné používat ji tam, kde je vyžadován přístup z mnoha systémů (např. síťové přihlašování uživatelů v operačním systému, kdy by se standardně musel systém nejdříve sám autentizovat nějakým speciálním účtem v databázi). Naopak ji lze použít v případě, kdy daný systém pracuje s celou databází (nebo tabulkami) uživatelských účtů a potřebuje k nim mít přímý přístup (např. některé databáze uživatelů poštovního serveru).

Relační databáze jsou založeny na relačním modelu. Ten je založen na matematické teorii množin a predikátové logice. Definuje způsob reprezentace dat a jejich ochranu, určuje nad nimi možné operace a odděluje je od jejich implementace. Jméno relace, názvy atributů a integritní omezení určuje tzv. schéma relace. Každá buňka dat by měla obsahovat pouze atomické hodnoty (nelze je dekomponovat do menších částí). S databázovou relací můžeme standardně pracovat jako s tabulkou. Relace je uspořádána pouze do řádků a sloupců, řádky nemají specifické pořadí, nelze je přímo identifikovat. Z tohoto důvodu se používá speciální sloupec (atribut) nebo soustava sloupců – primární klíč. Ten jednoznačně identifikuje řádek tabulky. Primární klíč je složen z klíčových atributů, ostatní atributy jsou neklíčové. [1]

V relačních databázích se separací dat do jednotlivých tabulek předchází jejich redundanci. Kvůli odstranění různých nedostatků se databáze normalizují. Při tomto procesu dochází např. k potlačení redundance a rizika vzniku aktualizační anomálie

(ztráta nebo nekonzistence dat po nějaké operaci). Po normalizaci se databáze nachází v tzn. normální formě.

Samotná práce s databázemi je řízena pomocí databázového systému – SŘBD (Systém řízení báze dat). Jedná se o aplikaci zapouzdřující ukládání dat. Pro zápis požadavků na SŘBD se používá dotazovací jazyk SQL (Structured Query Language).

Při správě uživatelských účtů je velikou výhodou relačních databází podpora transakcí. Transakce převádí databázi z jednoho konzistentního stavu do druhého. Všechny operace prováděně v transakci jsou vnímány jako jedna jednotka a buď jsou provedeny všechny nebo žádné. Nesmí nastat stav, kdy by se provedla jen část operací. Tato vlastnost je při manipulaci s uživatelskými daty uloženými v relační databázi nezastupitelná. Vytvoření uživatelského účtu, přidání uživatele do skupin apod. jsou typické operace vyžadující transakční zpracování. Jestliže během transakce dojde k chybě, lze transakci přerušit a data zůstanou v původním stavu. Externí přerušení transakcí (výpadek elektřiny, hardwarová porucha, …) ošetřuje samotný databázový systém.

Z pohledu správy uživatelských účtů jsou nejzajímavější databázové systémy MySQL a PostgreSQL. S oběma lze pracovat přes síť, lze je použít bezplatně a jsou podporované širokou škálou aplikací.

2.1.1 MySQL

Snad nejznámějším zástupcem relačních databází je databázový systém MySQL. Pro svou jednoduchost a snadnost používání se stal velice oblíbeným nejen u začínajících programátorů. Z tohoto důvodu je standardně nasazován komerčními i nekomerčními webovými hostingy. Velice často se používá při stavbě webových serverů na operačních systémech Linux v kombinaci s jazykem PHP a webovým serverem Apache HTTPD – tzv. LAMP (Linux, Apache, MySQL, PHP).

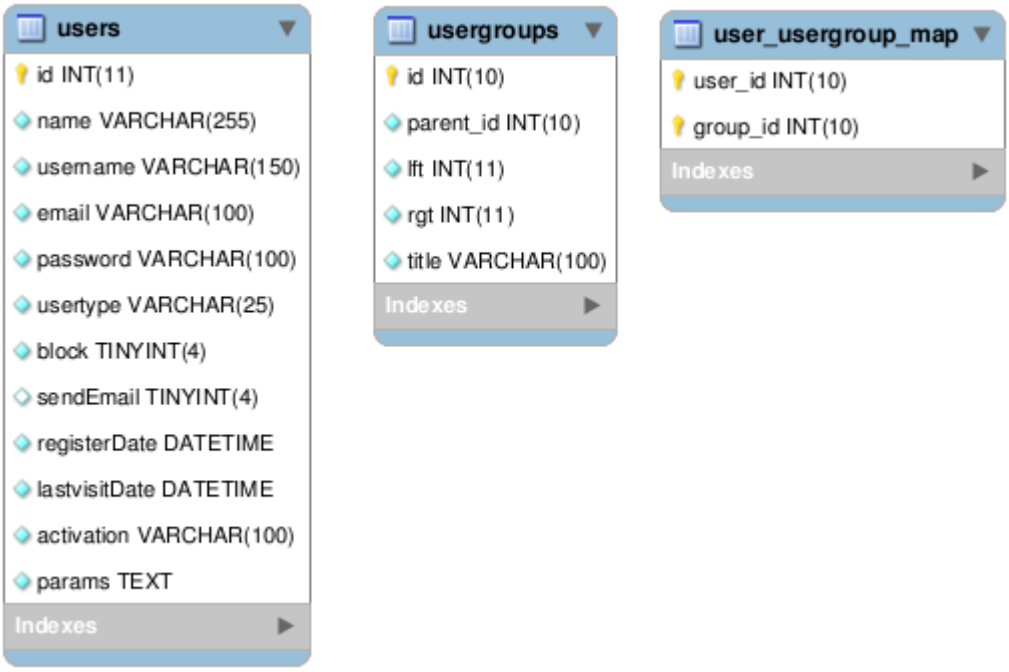
Databázový systém MySQL byl původně navržen jako jednoduchá a rychlá alternativa k ostatním databázím. Ačkoliv prošel značným vývojem, jeho původní návrh je na něm stále vidět. Nepodporuje některé standardní vlastnosti jako např. vícenásobné cizí klíče odkazující se na stejný sloupec, volání triggeru (spouštěče) přes cizí klíče apod. Na vytváření triggerů a procedur byla donedávna zapotřebí superuživatelská práva (oprávnění pro vytváření procedur bylo přidáno ve verzi 5.0.3 [2], oprávnění pro vytváření triggerů ve verzi 5.1.6 [3]). Kvůli tomu se příliš nepoužívají. MySQL je

rychlý převážně při jednoduchých operacích, složitější operace pro něj znamenají velice často podstatný výkonnostní propad.

Pro trvalé ukládání dat se v MySQL používají nejčastěji úložiště MyISAM a InnoDB. MyISAM je častěji používané převážně pro webové systémy, protože podporuje fulltextové vyhledávání. Nepodporuje však referenční integritu, o kterou se díky tomu musí starat koncová aplikace. Kvůli tomu se databáze snadno dostane do nekonzistentního stavu. Úložiště InnoDB na rozdíl od MyISAM podporuje transakce a cizí klíče. V poslední verzi MySQL (5.6) již InnoDB umí i fulltextové vyhledávání.

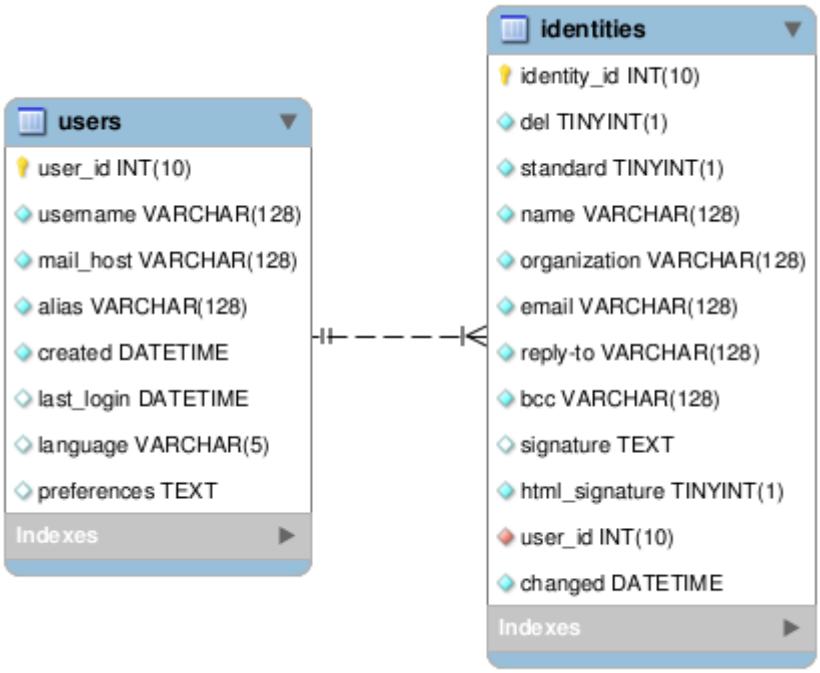
Vzhledem k tomu, že MySQL je nejčastěji dostupný databázový systém, používá jej převážná část známých open-source webových aplikací. Jsou mezi nimi např. redakční systém Joomla!, Roundcube webmail, e-learningový systém Moodle, MediaWiki (na této aplikaci běží Wikipedia) aj. Tyto a podobné aplikace podporují nebo dokonce vyžadují autentizaci třetí stranou (např. přes LDAP nebo IMAP v případě Roundube), informace o uživatelích ale ukládají lokálně do své databáze.

Při správě tabulek s uživatelskými účty v MySQL je potřeba dávat si pozor na návrh jednotlivých databází. Např. u úložiště MyISAM hrozí nebezpečí nekonzistentního stavu databáze (kvůli nepodpoře cizích klíčů a transakcí). Toto úložiště používá např. Joomla!, proto mezi tabulkami neexistují žádné relace (sloupec user_id tabulky user_usergroup_map by měl být cizí klíč sloupce id tabulky users a sloupec group_id tabulky user_usergroup_map by měl být cizí klíč sloupce id tabulky usergroups) viz. obrázek 2.1.



Obr. 2.1: EER diagram tabulek uživatelských účtů v redakčním systému Joomla!

Naopak Rouncube webmail používá InnoDB úložiště, tabulky jsou navzájem propojeny a referenční integritu kontroluje samotná databáze viz. obrázek 2.2.



Obr. 2.2: EER diagram vybraných tabulek webmailu Roundube

2.1.2 PostgreSQL

PostgreSQL je plnohodnotný relační databázový systém šířený pod BSD licencí. Jeho vývojáři se snaží o respektování a implementaci standardu ANSI SQL. Proto nepodporuje na rozdíl od MySQL neznaménkové číselné datové typy. Výkonnostně nezaostává za srovnatelnými komerčními systémy. Umožňuje běh uložených procedur napsaných v několika jazycích, podporuje funkcionální indexy a mnoho dalších pokročilých vlastností. [4]

Databázový systém PostgreSQL je méně populární než MySQL, který tvoří dominantní postavení u poskytovatelů webhostingových služeb. Jeho popularita ale roste a postupně začíná být podporován i aplikacemi dříve vyvíjenými speciálně pro MySQL (např. Joomla!). Tyto aplikace zároveň zachovávají podporu MySQL příp. jiných databázových systémů, a proto využívají PostgreSQL jen jako chytřejší úložiště dat.

2.2 Adresářové servery LDAP

Adresářové servery jsou nejčastěji používanou databází pro uchovávání uživatelských účtů. Vlastnosti adresářových serverů LDAP lze popsat a charakterizovat podle [5]. Na rozdíl od relačních databází jsou určeny především pro rychlé vyhledávání. Nepodporují složité transakce, nehodí se pro komplikované dotazy a časté změny dat. Adresáře lze velmi dobře zabezpečit, uživatelům lze vymezit, ke kterým atributům mají přístup. Přenosový protokol je standardizovaný, přenášená data lze šifrovat. V adresářových serverech lze ukládat i digitální certifikáty uživatelů. Adresářové služby lze použít jako autentizační zdroj pro poštovní servery, přihlašování uživatelů v operačním systému, webové aplikace aj. Adresářové servery nenahrazují relační databáze, nezvládají referenční integritu a transakce. Data uložená v adresářových serverech lze spravovat z jednoho místa. Adresářové služby se používají pro ukládání dat, která nejsou transakčního charakteru. Samotný adresářový server si data ukládá v nějaké jednoduché databázi (např. Berkeley DB), někdy dokonce i v relační databázi. V LDAP adresáři jsou data uspořádána v hierarchické stromové struktuře. Ta obvykle odráží nějaké organizační hranice.

Komunikace s adresářovým serverem probíhá protokolem LDAP. Klient odešle serveru dotaz a server vrátí výsledek. Protokol LDAP má devět základních operací:

- dotazovací (kladení dotazů LDAP serveru): search, compare;
- aktualizační (úprava záznamů v adresáři): add, delete, modify, modify RDN;
- autentizační a řídící: bind, unbind, abandon.

Datové typy, které lze v adresáři uložit definuje informační model. Objekty (objektové třídy, atributy, …) jsou jednoznačně identifikovány svým globální OID (object identifier). Základní jednotkou informace v adresářových službách je záznam (entry). Popisuje nějaký reálný objekt (např. počítač). Je složen z objektových tříd (objectclass). Objektová třída je množinou atributů, popisujících její vlastnosti. Atributy mají jednu nebo více hodnot (value). Na obrázku 2.3 je zobrazen ukázkový adresářový záznam.

Atribut	Hodnota
cn	Václav Bohata
givenName	Václav
mail	bohata@domena.tld

Obr. 2.3: Záznam

Adresářové servery většinou používají předdefinovaná schémata atributů a objektových tříd, lze ale dodefinovat vlastní.

Organizaci dat v adresáři definuje jmenný model. Cesta ke konkrétnímu záznamu ve stromě se vyjadřuje od kořene zprava doleva (cn=Václav Bohata, ou=Users, dc=domena, dc=tld). Adresářový strom lze rozdělit na více serverů. Jednotlivé větve stromu lze prohledávat samostatně. Každý záznam v adresáři lze identifikovat jedinečným rozlišovacím názvem DN (distinguished name), který je úplnou cestou k danému záznamu. V cestě jsou směrem zprava doleva uvedeny jednotlivé kontejnery oddělené čárkou. Název záznamu se skládá z názvu atributu, rovnítka a hodnoty atributu. V daném kontejneru lze jednotlivé záznamy identifikovat pomocí relativního rozlišovacího názvu RDN (relative distinguished name). Příklad DN a RDN:

DN: cn=Václav Bohata, ou=Users, dc=domena, dc=tld

RDN: cn=Václav Bohata

Pro autentizaci uživatele se používá operace bind. Po úspěšné autentizaci se provede autorizace a uživatel v závislosti na definovaném oprávnění dostane možnost

pracovat s určitými atributy. Operace bind umožňuje použít i anonymní autentizaci, při které se serveru nezasílají žádné informace o uživateli. Komunikace se serverem lze šifrovat pomocí SSL/TLS.

Pro import a export dat mezi adresářovými servery se používá standardní textový formát LDIF (LDAP Data Interchange Format). Existuje i formát založený na XML – DSML (Directory Services Markup Language).

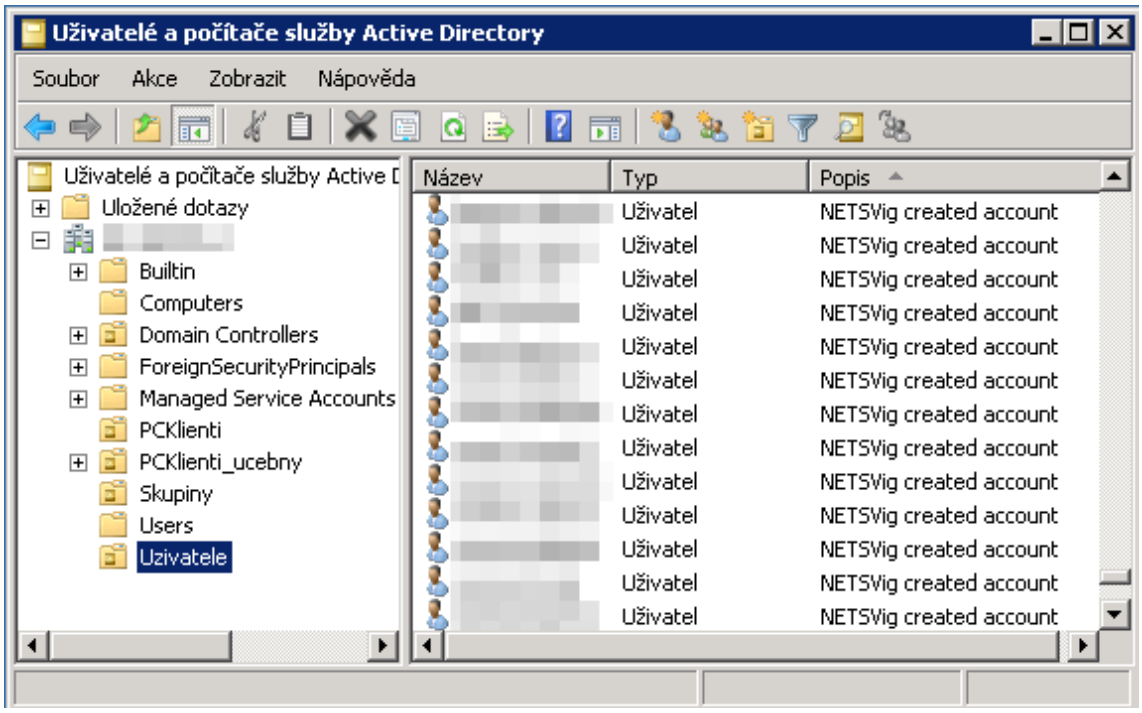
2.2.1 Microsoft Active Directory

Active Directory je NOS (Network Operating System) adresář používaný v sítích Windows, kde systémům poskytuje informace o různých zdrojích (sdílení, tiskárny, …). Ačkoliv to není typický LDAP server, lze s ním tímto protokolem komunikovat. V databázi Active Directory se nachází velká řada vzájemně propojených objektů. Jsou jimi např. servery, stanice, aplikace, skupiny, uživatelé aj.

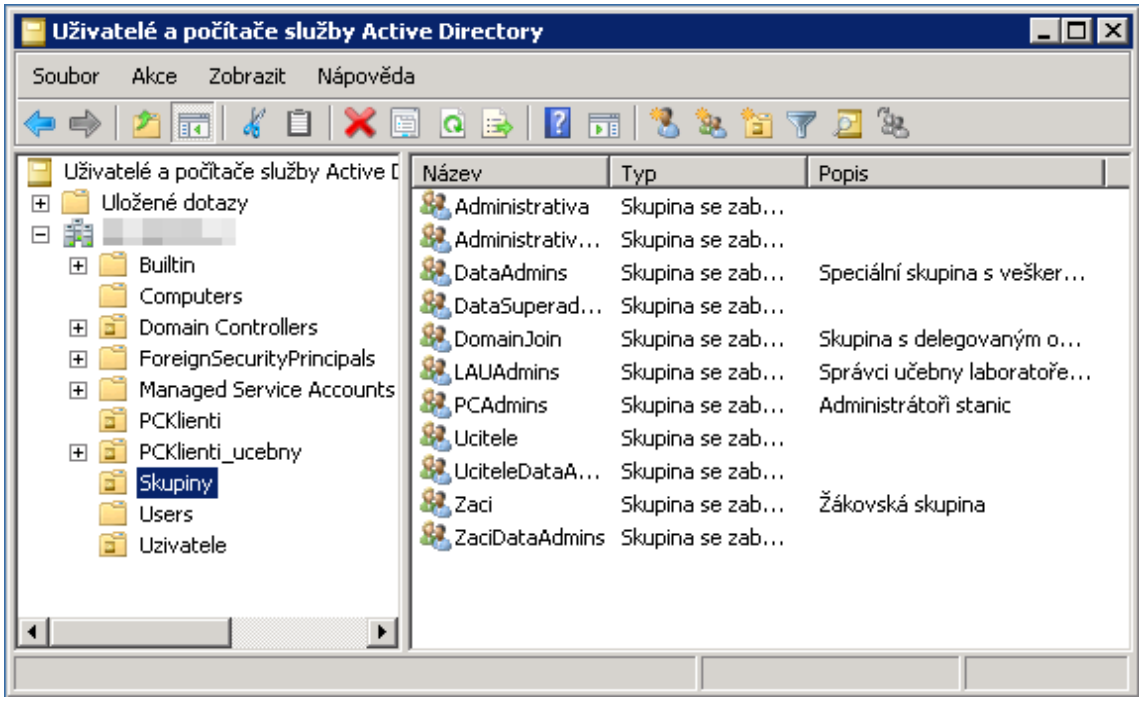
Logická struktura Active Directory je tvořena pomocí lesů, stromů, domén a organizačních jednotek (OU). Na vrcholu je les, ten může obsahovat několik stromů. Strom je tvořen doménami. Uvnitř domén jsou organizační jednotky, které obsahují jednotlivé objekty (počítače, uživatele, …). Fyzická struktura Active Directory je tvořena doménovými řadiči a sítěmi. [6]

Podle potřeb konkrétní organizace se uživatelé a skupiny nacházejí v jedné nebo více organizačních jednotkách příp. výchozích kontejnerech. Organizační jednotka je speciální kontejner, který umožňuje nastavovat skupinové politiky na všechny obsažené objekty. Na obrázcích 2.4 a 2.5 jsou zobrazeny[1] ukázky objektů v organizačních jednotkách tak, jak je vypisuje nástroj *Uživatelé a počítače služby Active Directory.*

1 Některé texty jsou rozostřené kvůli ochraně osobních údajů.



Obr. 2.4: Seznam uživatelů v organizační jednotce Uzivatele



Obr. 2.5: Seznam skupin v organizační jednotce Skupiny

2.2.2 Další významné adresářové servery

Vzhledem k převaze operačních systémů Microsoft Windows a jejich použití v nejrůznějších počítačových sítích mají ostatní adresářové servery obvykle jen minoritní zastoupení.

Konkurentem Active Directory se snaží být produkt Novell eDirectory. Adresářová služba eDirectory je založena na standardu X.500. Snaží se působit jako integrující prvek v multiplatformních prostředích.

Důležitým měřítkem při volbě adresářového serveru je jeho cena. Často se proto volí bezplatná řešení jako např. 389 Directory Server, který kromě mnoha jiných vlastností nabízí i možnost synchronizace uživatelů a skupin s Active Directory. Dalšími z bezplatných řešení jsou OpenLDAP a ApacheDS.

3 Nástroje pro centrální správu uživatelských účtů

V praxi mají uživatelé zpravidla přístup k několika informačním systémům. Pokud tyto systémy nepoužívají žádný systém centrálního ověřovaní, má uživatel několik různých přístupových účtů. Vzhledem k náročnosti správy takových systémů a potřeby znalosti toho, ke kterým systémům má uživatel přístup se především v podnicích používá software pro centrální správu uživatelů – identity management, jehož charakteristikou se zabývá [7].

Základním procesem identity managementu je životní cyklus identity uživatele. Ten většinou začíná nástupem zaměstnance do organizace. Jsou mu zřízeny základní přístupy, jež se v průběhu času mění a přibývají další. Často je potřeba měnit uživatelská jména nebo organizační zařazení. Po odchodu zaměstnance jsou mu účty vymazány.

Identity management (IdM) je informační systém, který z jednoho místa dokáže ovládat životní cyklus uživatelských účtů v organizaci a pomocí auditu sledovat jejich změny. Nový pracovník je zařazen do personálního systému Human resources (HR) a založen do systémů společnosti – tzv. provisioning účtů. Kdykoliv je možné zjistit aktuální účty a oprávnění, jimiž uživatel disponuje. Pomocí workflow IdM je možné zautomatizovat řadu procesů (např. synchronizaci identit ze systému HR).

Typická architektura IdM je centrálně orientovaná. Koncové systémy (systémy obsahující vlastní úložiště uživatelských účtů) jsou k centrálnímu uzlu napojeny pomocí tzv. konektorů. Konektor obstarává spojení k úložišti uživatelů v koncovém systému. Identity management funguje na principu dodání potřebných dat do koncových systémů na základě autoritativní informace. Koncové systémy si vlastními prostředky řídí autentizaci a autorizaci uživatele proti svému úložišti. Opačnou filozofií je access management, kde ověření uživatelů probíhá prostřednictvím centrální autority. V praxi se mohou obě architektury kombinovat do tzv. identity a access managementu.

Účty v koncových systémech je nutné spárovat s účty v IdM. Tomuto procesu se říká rekonciliace. Při tomto procesu se na základě logických podmínek přidělují účtům v koncových systémech identity IdM. Účty bez vlastníka jsou reportovány nebo rovnou

mazány z koncového systému. IdM je napojen na autoritativní zdroj identit (obvykle HR systém organizace).

IdM obvykle nabízí webové uživatelské rozhraní, které umožňuje změnu hesla nebo podání žádosti o zřízení přístupu do koncového systému. Rozhraní je možné rovněž integrovat do stávajícího firemního help desku.

Uživatelé mají přístup k určitému obsahu či funkcím na základě systémového oprávnění, tedy autorizace. K řízení autorizačních dat v koncových systémech lze s výhodou využít IdM. Rozšířeným principem v řízení oprávnění je Role-Based Access Control (RBAC), kde uživatelé získávají oprávnění nepřímo přes opakovaně přidělitelné objekty – role. Uživatelé mohou mít stovky aplikačních rolí, které se kvůli efektivní správě sdružují do skupin. Tyto skupiny se nazývají business role. Business role mohou odkazovat na více oprávnění v různých koncových systémech organizace.

4 Návrh aplikace NETSVig

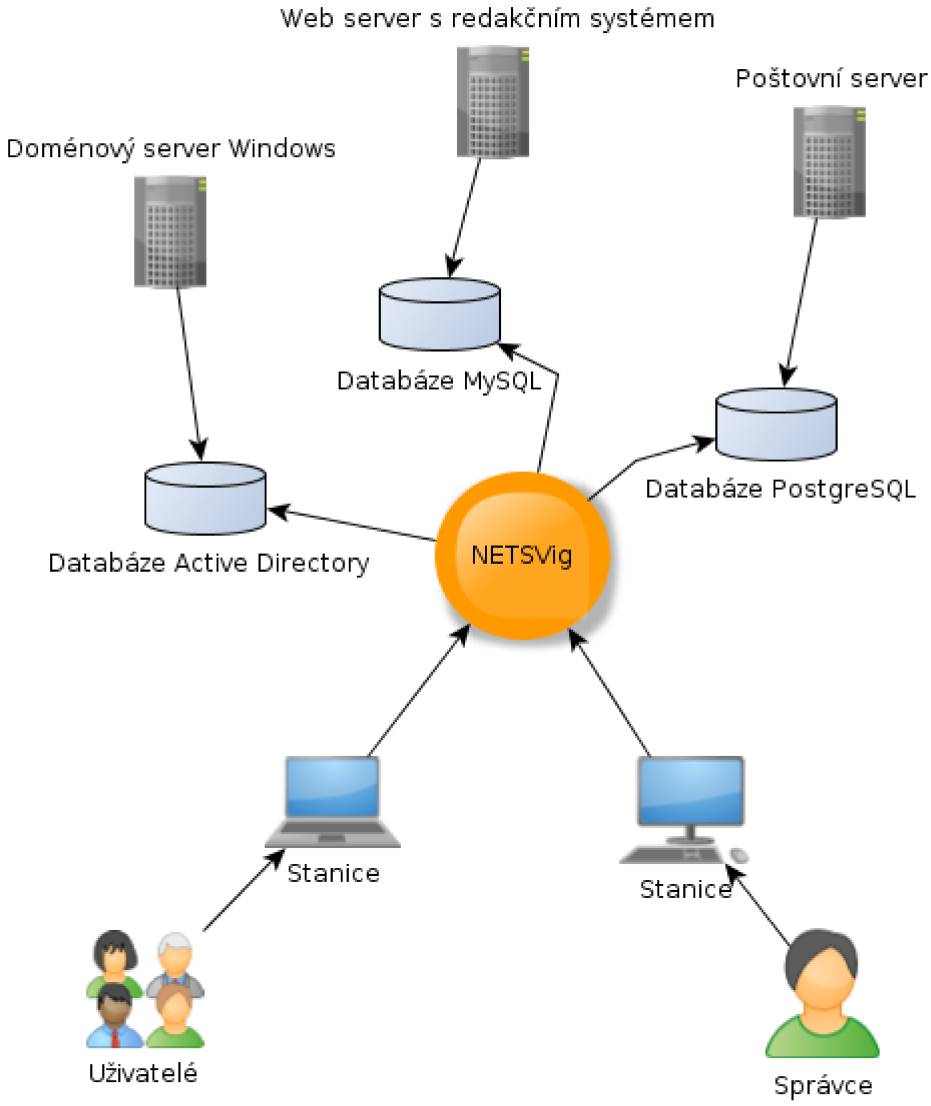
4.1 Cíle při návrhu a vlastnosti aplikace

Cílem návrhu bylo vytvořit aplikaci, která by umožňovala rychlým a efektivním způsobem hromadně spravovat větší množství uživatelů. Aplikaci jsem vyvíjel tak, aby nevyžadovala úpravy spravovaných databází a mohla být používána paralelně s již existujícími systémy centrální správy. Je navržena především s ohledem na bezpečnost na různých úrovních – od samotné manipulace s daty až po zabezpečení samotné aplikace. Nevyžaduje vždy plný přístup do spravovaných databází, automaticky nemění data ve spravovaných databázích a díky podpoře uživatelských oprávnění zamezuje neoprávněné manipulaci s daty. Komponenty aplikace jsou navrženy tak, aby mohly být rozmístěny v různě zabezpečených systémech a vlastní implementace se snaží zamezit útokům na samotnou aplikaci. Aplikace umožňuje velice snadné rozšíření pomocí modulů a pluginů. Je počítáno i s možností nasazení v hardwarově slabších systémech, proto je kladen důraz na minimalizaci paměťových nároků. Tato minimalizace však nesmí být na úkor rychlé manipulace s uživatelskými daty, protože právě manipulace s daty je klíčová činnost aplikace.

Aplikace se svým návrhem přibližuje identity management systémům. Na rozdíl od nich je ale založena na principu neexistence přímých autoritativních záznamů pro spravované databáze (systémy). Každá spravovaná databáze obsahuje jedinečnou autoritativní informaci o daném uživateli. Mezi databázemi neexistuje žádná závislost. Díky tomu může mít uživatel v jednotlivých systémech různé identifikační údaje. Hlavní výhodou tohoto přístupu je možnost spravovat vybrané databáze z různých aplikací najednou nezávisle na sobě. Synchronizovanou změnu identifikačních údajů uživatelů lze řešit vlastním rozšiřujícím modulem. Aplikace NETSVig neobsahuje mechanismy pro rekonciliaci. Spárování uživatelských identit probíhá automaticky na základě shody uživatelského jména, případně jeho části či jiného podobného atributu. To při dobré znalosti spravovaných databází umožní rychlé nasazení aplikace do již zavedených informačních systémů.

4.2 Architektura aplikace

Aplikace se skládá z frontendu, který tvoří webové rozhraní a backendu, který obstarává aktualizaci lokální cache dat spravovaných databází. Vnější pohled na aplikaci zobrazuje obrázek 4.1.



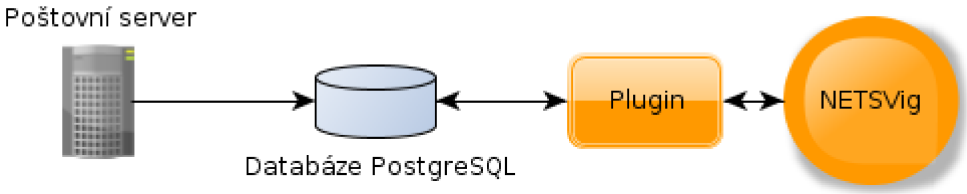
Obr. 4.1: Vnější pohled na aplikaci s ukázkou spravovaných databází

K aplikaci se připojují správci a běžní uživatelé. Správci spravují jednotlivé uživatele, přidávají a odebírají uživatelské účty. Běžní uživatelé si prostřednictvím aplikace NETSVig mění hesla do libovolných spravovaných systémů. Aplikace NETSVig je připojena ke spravovaným databázím koncových systémů a provádí v nich vlastní změny (změny hesel, správu uživatelů, …). Koncové systémy používají své lokální

databáze uživatelských účtů nezávisle na tom, že je spravuje aplikace NETSVig. Nijak s aplikací NETSVig nekomunikují. Veškerou komunikaci obstarává centrálně aplikace NETSVig (tzv. bezagentový přístup) prostřednictvím pluginů.

4.2.1 Pluginy

Pluginy plní funkci konektorů do koncových spravovaných databází. Tvoří vrstvu mezi těmito databázemi a jádrem aplikace NETSVig. Nabízejí funkce pro čtení dat z koncových databází, umožňují přidávat, mazat a upravovat uživatelské účty. Obrázek 4.2 znázorňuje komunikaci s koncovou databází.

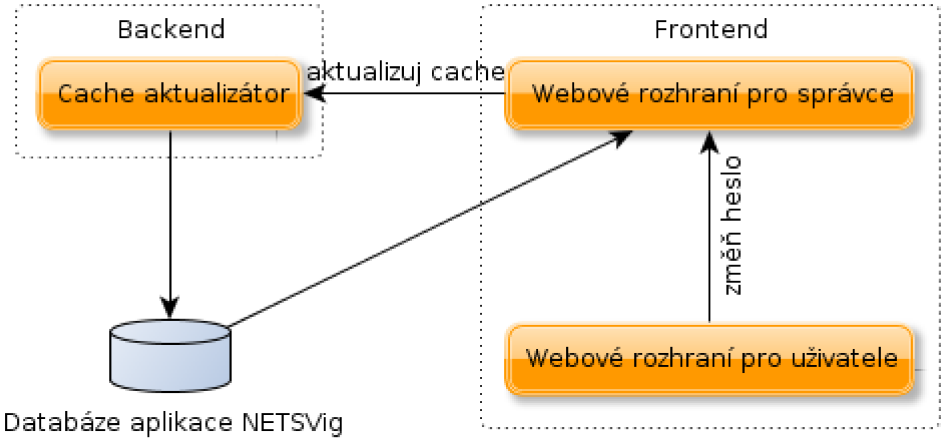


Obr. 4.2: Komunikace s koncovou databází

Každá spravovaná databáze je popsaná pomocí atributů. Atributy popisují jednotlivé vlastnosti spravované databáze a definují nakládání a způsob práce s daty.

4.3 Komponenty aplikace

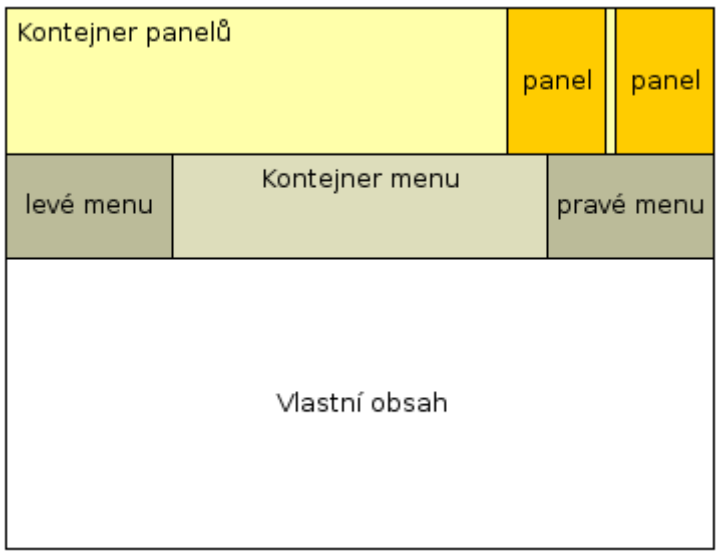
Frontend aplikace je tvořen hlavním webovým rozhraním pro správce a webovým rozhraním pro běžné uživatele. Backend tvoří aktualizační skript, který se pravidelně připojuje ke spravovaným databázím, stahuje z nich uživatelská data a ukládá je do lokálního úložiště cache. Každou komponentu aplikace lze provozovat samostatně na vlastním serveru. Propojení těchto komponent znázorňuje obrázek 4.3.



Obr. 4.3: Propojení komponent aplikace

4.3.1 Webové rozhraní pro správce

Jedná se o hlavní komponentu aplikace, která má na starost vlastní správu uživatelských účtů přes webový prohlížeč. Po úspěšné autentizaci nabízí správci na základě jeho oprávnění možnost spravovat určité množství uživatelských atributů v různých koncových databázích. Rozhraní je navrženo tak, aby umožňovalo v okně prohlížeče zobrazit najednou co největší množství dat a poskytovalo zároveň funkce potřebné pro manipulaci s nimi. Rozložení rozhraní je vidět na obrázku 4.4.



Obr. 4.4: Rozložení webového rozhraní pro správce

Panely poskytují nástroje pro manipulaci s daty (filtrování, řazení, …), menu slouží pro navigaci v aplikaci, volání funkcí pro manipulaci s daty apod. Funkční celky rozhraní, mezi nimiž probíhá navigace, se skládají z úloh (tasks) a akcí (actions). Úloha je např. administrace, akce editace ACL.

Webové rozhraní zobrazuje data uložená v lokální cache. Umožňuje je efektivně a rychle filtrovat a řadit podle několika kritérií zároveň. Každá úprava spravované databáze (vytvoření, odstranění nebo úprava uživatelského účtu) vyvolá aktualizaci lokální uživatelské cache. Během manipulace s daty může dojít k přerušení spojení s aplikací vlivem výpadku elektřiny, přerušení internetového spojení apod. Webové rozhraní pracuje při editaci s daty uloženými v lokální cache, zobrazuje je a nabízí jejich úpravu. Cache proto musí reflektovat aktuální stav spravované databáze. Proto probíhá úprava koncové databáze v následujících krocích:

1. Vytvoření plánu aktualizace – do databáze se zapíše informace, že je za určitý čas potřeba aktualizovat v lokální cache atributy právě spravované databáze pro aktuálně upravovaného uživatele.
2. Úprava spravované databáze – v této fázi se provede vlastní modifikace atributů nebo vytvoření či smazání účtu ve spravované databázi pro konkrétního uživatele.
3. Aktualizace lokální cache – aktualizují se změněná data tak, aby cache reflektovala data spravované databáze.
4. Smazání plánu aktualizace – v této fázi proběhly veškeré změny a je bezpečné smazat naplánovanou aktualizaci cache.

Pokud by během kroků 2-3 došlo k přerušení operace, pak komponenta cache aktualizátoru na základě vytvořeného plánu aktualizace provede aktualizaci cache sama.

Webové rozhraní kromě kompletního přehledu všech koncových databází nabízí i možnost hromadné úpravy dat pro větší množství uživatelů. Rozhraní lze pomocí modulů doplnit např. o možnost importu a exportu dat. Standardně je součástí aplikace modul pro import uživatelských účtů z textového souboru.

Webové rozhraní pro správce nabízí jednotlivým uživatelům nepřímo možnost změny jejich hesel. Poskytuje webové aplikační rozhraní (Web API), přes které je možné tyto změny provádět.

4.3.2 Webové rozhraní pro uživatele

V tomto rozhraní si může uživatel změnit hesla ve spravovaných systémech i centrální heslo do aplikace NETSVig. Každý spravovaný uživatel má přidělené kromě hesel do koncových systémů i heslo centrální pro aplikaci NETSVig. Pouze tímto heslem se autentizuje při změně hesel do spravovaných systémů. Uživatel si vybere koncové databáze, ve kterých si přeje nastavit nové heslo a provede změnu do všech vybraných databází najednou. Každá spravovaná databáze má vlastní požadavky na sílu hesla, a proto se nové heslo nastaví pouze v těch databázích, pro které nové heslo odpovídá požadavkům na sílu hesla.

Webové rozhraní pro uživatele je vlastní nezávislá komponenta aplikace, která komunikuje s jedním nebo více webovými rozhraními pro správce pomocí Web API. Pro každou změnu hesla odešle vlastní HTTP požadavek. V požadavku se odešle název klienta (zvolený název webového rozhraní), typ akce (změna hesla), uživatelské jméno, NETSVig heslo, nové heslo a název koncové databáze. Každý požadavek se posílá SSL vrstvou šifrovaným protokolem HTTPS. Vlastní data v požadavku jsou navíc šifrována 256bitovou šifrou Rijndael. Heslo je možné změnit pouze tehdy, když cílové webové rozhraní dokáže požadavek dešifrovat (zná šifrovací klíč klienta) a má dostatečné oprávnění k provedení této akce. Výhodou tohoto přístupu je kromě možnosti komunikace s několika webovými rozhraními pro správce i vyšší zabezpečení aplikační struktury. K tomuto rozhraní mají přímý přístup koncoví uživatelé a pokud by nějakým způsobem došlo k jeho napadení (např. nepřímo přes útok na vlastní server, kde je toto rozhraní provozováno), pak nedojde k ohrožení celé aplikace a dat.

4.3.3 Cache aktualizátor

Cache aktualizátor je spouštěn periodicky operačním systémem. Stahuje spravovaná uživatelská data koncových databází do lokální cache. V cache jsou uložena všechna data, se kterými aplikace NETSVig přímo pracuje. Nikdy se do ní ale neukládají uživatelská hesla. Cache aktualizátor může pracovat ve třech módech – static, full a tracking.

Mód static

Mód static slouží k jednorázovému načtení dat z koncových systémů. K aktualizaci cache poté dochází pouze po úpravě uživatele vyvolané aplikací

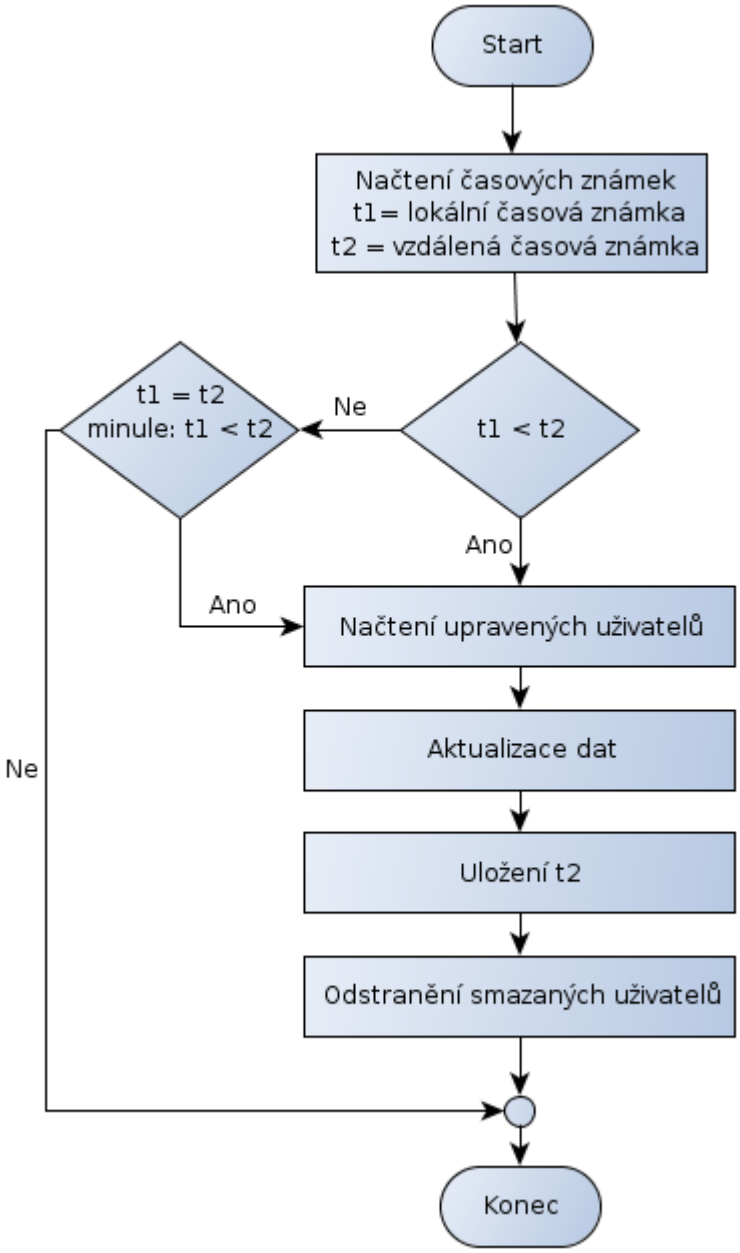
NETSVig. Tento mód je vhodný použít tam, kde je aplikace NETSVig jediným nástrojem, který spravuje koncovou databázi. Není vhodné použít jej pro nestatické databáze.

Mód full

V tomto módu dojde při každém periodickém spuštění cache aktualizátoru k úplnému stažení spravovaných dat do lokální cache. Uživatelská data, která nebyla stažena jsou z cache vymazána. Tento mód je vhodné použít u dynamických databází, ve kterých dochází ke změnám hodnot některých atributů (např. využití kvóty poštovní schránky uživatele). Rovněž je vhodné použít ho tehdy, není-li aplikace NETSVig jediným nástrojem pro správu cílové databáze. Mód full lze použít u databází s maximálně několika stovkami uživatelských účtů. Periodické stahování dat většího množství uživatelů by trvalo neúměrně dlouhý čas.

Mód tracking

Cache aktualizátor v módu tracking stahuje jen rozdílová data. Na základě časové známky zdrojových dat rozezná, u kterých uživatelů nastala změna hodnot atributů. Časová známka může být buď přímo časový údaj nebo jakákoliv jiná hodnota, která se po aktualizaci uživatele inkrementuje a lze ji porovnávat. V případě použití časového údaje je potřeba, aby frekvence spouštění cache aktualizátoru byla menší než jeden krok časového údaje. Např. pokud je časový údaj v minutách, pak perioda spouštění cache aktualizátoru musí být větší než jedna minuta. Časový údaj se musí pouze inkrementovat (použití UTC, nastavování hodin zpomalováním a zrychlováním systémového času). Nevýhodou módu tracking je nutná existence časových razítek v koncových databázích. Algoritmus módu tracking je na obrázku 4.5.



Obr. 4.5: Diagram aktualizace dat v módu tracking

Skládá se z následujících kroků:

1. Detekce změn zdrojových hodnot – z lokálních metadat se načte časová známka poslední aktualizace a porovná se z poslední časovou známkou ve spravované databázi. Jestliže je poslední časová známka ve spravované databází větší než lokální, pokračuje se bodem 2. Jestliže se časové známky rovnají a v předcházející aktualizaci byla vzdálená časová známka větší, pokračuje se rovněž bodem 2 (v opačném případě

algoritmus skončí). To sice způsobí vícenásobnou aktualizaci některých dat, ale zajistí dokonalou detekci všech změn i v případech, kdy se provádí načítání uživatelských dat právě ve stejném čase, kdy se upravují ostatní uživatelé.

2. Načtení upravených uživatelů – z koncové databáze se načtou všichni uživatelé s časovou známkou větší nebo rovnou lokální časové známce a zároveň menší nebo rovnou poslední časové známce v koncové databázi.
3. Aktualizace dat – pro všechny načtené uživatele se provede aktualizace hodnot atributů z koncové databáze do lokální cache. V případě neexistence uživatelského záznamu v cache se tento záznam automaticky vytvoří.
4. Uložení časové známky – do lokálních metadat se uloží nejvyšší časová známka z aktualizovaných dat.
5. Odstranění neplatných uživatelských záznamů z lokální cache – porovná se seznam uživatelů v cache se seznamem uživatelů v koncové databázi. Z cache jsou vymazáni uživatelé, kteří již neexistují v koncové databázi.

4.4 Modulární rozšiřitelnost aplikace

Aplikace nabízí možnost funkčního rozšíření nebo implementaci nových funkcí pomocí tzv. modulů. Je možné rozšiřovat jednotlivé aplikační třídy, přidávat novou funkcionalitu do webového rozhraní nebo definovat novou šablonu webového rozhraní. Modul obsahuje inicializační část a definici rozšíření. V inicializační části se registrují jednotlivá rozšíření. Zaregistrovat lze čtyři druhy rozšíření: task, action, handler a hook.

4.4.1 Task

Task registruje novou úlohu ve webovém rozhraní pro správce. Vytváří tak nový bod pro navigaci. Nedefinuje funkčnost nové úlohy, provádí pouze její registraci v aplikaci.

4.4.2 Action

Action registruje jednu nebo více akcí pro danou úlohu ve webovém rozhraní pro správce. Lze registrovat:

- univerzální akce, které se vyvolají při jakékoliv navigaci ve webovém rozhraní,
- akce vázané na jméno a libovolnou úlohu, které se vyvolají při navigaci na jakoukoliv úlohu a akci takto nazvanou,
- akce vázané na úlohu, které se vyvolají při jakékoliv navigaci na libovolnou akci v dané úloze,
- akce vázané na jméno a konkrétní úlohu, které se vyvolají pouze při navigaci na takto nazvanou akci pouze v konkrétní úloze.

V definici action se definuje reakce na danou akci. Tou může být přidání položky v menu webové aplikace, vytvoření nového obsahu, modifikace dat apod.

4.4.3 Handler

Handler registruje obslužnou rutinu pro daný kus kódu. Rozšiřuje možnost jednotlivých tříd. V aplikaci lze zaregistrovat pouze jeden handler daného jména. Typickým příkladem použití handleru může být rozšíření schémat podporovaných hesel ve třídě, která má na starosti manipulaci s hesly. Každá aplikační třída musí sama obstarávat volání daného handleru a předání argumentů. Handler pak vrátí požadovaný výstup.

4.4.4 Hook

Hook je funkčností podobný handleru. V aplikaci může být zaregistrováno více hooků stejného jména. Při spouštění daným aplikačním kódem jsou volány v pořadí, ve kterém byly registrovány. Pokud při spuštení hook vrátí pole, je spojeno s polem argumentů současně volaného hooku a tyto argumenty jsou předány dalšímu hooku v pořadí. Toto chování je inspirováno podobnou funkcionalitou v aplikaci Rouncube webmail [8].

4.5 Mechanismus oprávnění

Uživatelé získávají oprávnění nepřímo pomocí členství ve skupinách. Uživatel může být členem několika skupin. Každá skupina definuje dva druhy oprávnění: datová a ACL. Aplikace NETSVig rozeznává tři typy uživatelů: registered, admin, superadmin. Registered je běžný uživatel spravovaný aplikací NETSVig. Nemůže pracovat ve webovém rozhraní pro správce a nemůže být členem žádných skupin. Má jediné

implicitní oprávnění ke změně vlastního hesla. Uživatel typu admin je správce, který spravuje ostatní uživatele ve webovém rozhraní pro správce. Může nabývat oprávnění prostřednictvím členství ve skupinách. Má minimálně stejná oprávnění jako typ registered. Uživatel typu superadmin spravuje kromě ostatních uživatelů i uživatelské skupiny, definuje oprávnění a přiděluje správce do skupin. Má povoleny všechny akce ve webovém rozhraní, ale stejně jako typ admin nemá implicitně oprávnění k uživatelským datům. Toto oprávnění si ale může sám sobě přidělit.

4.5.1 ACL oprávnění

Oprávnění ACL určují, k jakým zdrojům smí skupina přistupovat a jaké operace s nimi může provádět. Zdrojem může být např. koncový systém a oprávněním možnost vytváření uživatelů. Skupina může definovat, zda je dané oprávnění povoleno, zakázáno nebo ponecháno ve výchozím stavu. ACL oprávnění jsou uložená ve speciální tabulce v databázi. Pokud je uživatel členem více skupin, může se stát, že některé skupiny oprávnění povolí, jiné zakážou. V takovém případě je rozhodující počet povolovacích a zakazovacích oprávnění. Pokud je počet povolovacích oprávnění větší než počet zakazovacích, je oprávnění uděleno. Je-li počet zakazovacích oprávnění roven nebo menší počtu povolovacích, je oprávnění odepřeno. Jestliže není dané oprávnění povoleno nebo zakázáno, je na aplikaci, jak se ke konkrétnímu oprávnění zachová. Některé operace mohou vyžadovat povolení dané akce, jiným může stačit nezakázání dané akce.

Příklad ACL oprávnění

V tomto příkladě je přihlášený uživatel typu admin členem skupin „skupina 1“ a „skupina 2“. Tabulka 4.1 zobrazuje definici oprávnění. Protože je uživatel členem prvních dvou skupin, má zakázané přidávání uživatelů do spravované databáze (jedna skupina oprávnění povoluje, druhá zakazuje). Uživatel není členem skupiny „skupina 3“, a proto nemá zakázané zobrazování uživatelů. Nemá ho však ani povolené. Jestliže by aplikace kontrolovala pouze přítomnost zákazových oprávnění, potom může uživatel zobrazovat ostatní uživatele zdroje NETSVig.

Tab. 4.1: Definice ACL oprávnění

Skupina	Typ oprávnění	Zdroj	Oprávnění
skupina 1	povolovací	spravovaná databáze	přidat uživatele
skupina 2	zakazovací	spravovaná databáze	přidat uživatele
skupina 3	zakazovací	NETSVig	zobrazit uživatele

4.5.2 Datová oprávnění

Datová oprávnění určují povolenou množinu operací nad definovanými atributy při aktuálním stavu hodnot atributů. Na základě porovnávání aktuální hodnoty cílového i ostatních atributů s definovanými hodnotami vyhodnocují povolené operace. Lze tak např. definovat, že atribut „datetime“ je možné zobrazovat pouze tehdy, když je jeho hodnota větší než definovaná. Datová oprávnění jsou na rozdíl od ACL oprávnění pouze povolující. Je zakázáno vše, co není explicitně povoleno. Datová oprávnění jsou mocný a složitý nástroj pro řízení uživatelských účtů. Jejich výpočet nelze provádět online. Proto se po každém přihlášení uživatele typu admin nebo superadmin do webového rozhraní pro správce přepočítávají oprávnění pro čtení dat. Rychlost přepočítávání závisí na množství dat a rozsáhlosti definic datových oprávnění. Při prvním přihlášení uživatele může trvat i několik sekund, po každém následujícím přihlášení se přepočítávají jen u dat, která jsou označena jako změněná. Označování změněných dat provádí automaticky databáze po změně hodnot spravovaných dat nebo po úpravě definice datového oprávnění jen pro uživatele, kterých se tato změna týká.

Příklad datového oprávnění

Následující příklad demonstruje chování datových oprávnění pro data nějakého uživatelského účtu, která jsou zobrazena v tabulce 4.2. V prvním sloupci je uveden systém, jehož se daný atribut týká. Systém NETSVig je centrální databází uživatelů. Všichni uživatelé jsou zde registrováni. Systém Active Directory je spravovaná koncová databáze. Přihlášený uživatel typu admin nebo superadmin bude měnit u nějakého uživatele hodnotu atributu *členem* systému Active Directory.

Tab. 4.2: Úprava dat

Systém	Atribut	Původní hodnota	Nová hodnota
NETSVig	uživatelské jméno	jan.novak	jan.novak
NETSVig	kategorie	Učitelé	Učitelé
NETSVig	podkategorie	dílny	dílny
Active Directory	zobrazované jméno	Jan Novák	Jan Novák
Active Directory	členem	**Ucitele**	**Ucitele** **PCAdmins**
Active Directory	poslední přihlášení	2010-02-02 12:12:12	2010-02-02 12:12:12
Active Directory	UNIX shell	/home/jan.novak	/home/jan.novak

Přihlášený uživatel je členem skupiny, která pro koncový systém Active Directory definuje dvě oprávnění, viz. tabulky 4.3 a 4.4. Aplikace NETSVig umožňuje v případech, kdy se kontroluje oprávnění ke spravovaným koncovým systémům porovnávat oprávnění i proti atributům samotného centrálního systému (NETSVig). Opačně to nefunguje. Nelze během kontroly přístupu k atributu systému NETSVig porovnávat hodnoty atributů ostatních systémů. Aplikace podporuje následující operátory porovnání:

- menší než: <,
- menší nebo rovno: <=,
- větší než: >,
- větší nebo rovno: >=,
- rovnost: =,
- rovnost bez ohledu na velikost písmen: ~=,
- nerovnost: <>,
- nerovnost bez ohledu na velikost písmen: ~<>,
- odpovídá Perl kompatibilnímu regulárnímu výrazu: regexp,
- neodpovídá Perl kompatibilnímu regulárnímu výrazu: notregexp.

Před úpravou atributu *členem* zkontroluje aplikace oprávnění pro modifikaci. Přihlášený uživatel musí mít oprávnění k modifikaci dat před i po úpravě hodnot. Oprávnění, které definuje skupina pro konkrétní systém se kontrolují postupně. Neodpovídá-li oprávnění požadovanému přístupu, pokračuje se dalším definovaným oprávněním. Procházejí se všechna oprávnění, která uživatel nabyl členstvím

v jednotlivých skupinách. Jestliže ani poslední oprávnění nepovoluje požadovanou operaci, je přístup k této operaci zamítnut. Jako první se prochází definice oprávnění v tabulce 4.3. V tomto případě se dá říci, že se kontrolují pouze části, které definují oprávnění k modifikaci atributu *členem*. Všechny tyto části musejí souhlasit – provádí se mezi nimi průnik (AND). Oprávnění je povoleno pouze tehdy, když je upravovaný uživatel zařazen v kategorii „Učitelé“ aplikace NETSVig a zároveň je atribut *členem* systému Active Directory roven (bez ohledu na velikost písmen) hodnotě „ucitele“. Oprávnění sice odpovídá původním datům, ne však cílovým. Po změně by totiž hodnota vícehodnotového atributu *členem* byla „ucitele“ a „pcadmins“. To ovšem oprávnění nepovoluje. Proto se pokračuje dalším oprávněním definovaném v tabulce 4.4. To definuje, že pro modifikaci dat musí být atribut *členem* systému Active Directory roven (bez ohledu na velikost písmen) hodnotě „ucitele“ a/nebo „pcadmins“. Žádné další omezující podmínky nejsou definovány. Oprávnění odpovídá jak původním, tak cílovým datům. Modifikace atributu *členem* je proto povolena a k vyhodnocení dalších oprávnění v pořadí již nedochází.

Tab. 4.3: Definice oprávnění 1 pro systém Active Directory

Definuje		**Porovnává**			
Oprávnění	**Atribut**	**Systém**	**Atribut**	**Porovnání**	**Hodnota**
číst	všechny	NETSVig	kategorie	=	Učitelé
modifikovat	členem	NETSVig	kategorie	=	Učitelé
modifikovat	členem	Active Directory	členem	~=	ucitele

Tab. 4.4: Definice oprávnění 2 pro systém Active Directory

Definuje		**Porovnává**			
Oprávnění	**Atribut**	**Systém**	**Atribut**	**Porovnání**	**Hodnota**
modifikovat	členem	Active Directory	členem	~=	ucitele pcadmins

V některých případech může nastat situace, kdy je definováno oprávnění pro atributy, které systém nezná (např. byly vymazány úpravou schématu koncového systému, ale zůstaly definované) nebo nemá k dispozici jejich data (např. definice koncového systému zakazuje ukládání hodnot některých atributů do lokální cache). V takovém případě jsou tyto atributy v definici oprávnění ignorovány. Problematické

atributy se dají identifikovat během editace oprávnění ve webovém rozhraní pro správce.

4.6 Manipulace s uživatelskými daty

Aplikace manipuluje s daty v různých databázích. Vzhledem k různým vlastnostem spravovaných databází a jejich specifickým datovým typům je nutné data během manipulace nějakým způsobem transformovat.

4.6.1 Datové typy

Každá spravovaná databáze může podporovat jinou množinu datových typů. Aby bylo možné s daty jednotně pracovat, je potřeba reprezentovat je jednotnými datovými typy. Proto aplikace NETSVig implementuje vlastní sadu datových typů. Všechna data, se kterými aplikace manipuluje se obalují objekty, definujícími jejich datové typy. Všechny datové typy koncových databází musejí být přeloženy do jednoho z následujících podporovaných datových typů aplikace NETSVig:

- Binary: obsahuje binární data o maximální velikosti 16 MiB,
- Bool: logický datový typ, jehož hodnota je TRUE nebo FALSE,
- Date: uložení data,
- Time: uložení času,
- Datetime: uložení data a času,
- Int8: 8bitové celé číslo,
- Int16: 16bitové celé číslo,
- Int32: 32bitové celé číslo,
- Int64: 64bitové celé číslo,
- Double: číslo s plovoucí desetinnou čárkou, velikost je závislá na platformě.

4.6.2 Filtrace dat

Při přenosu dat mezi databázemi (obvykle mezi spravovanou databází a lokální cache) se může vyskytnout potřeba úpravy přenášených dat. Obsahuje-li např. zdrojová databáze informaci o využití kvóty poštovní schránky v bajtech, mohou se podle potřeby data před uložením do lokální cache transformovat na megabajty. Filtrace

transformuje data do jiné podoby. Díky ní je možné lépe přizpůsobit spravovanou koncovou databázi potřebám aplikace NETSVig.

4.6.3 Kontrola dat

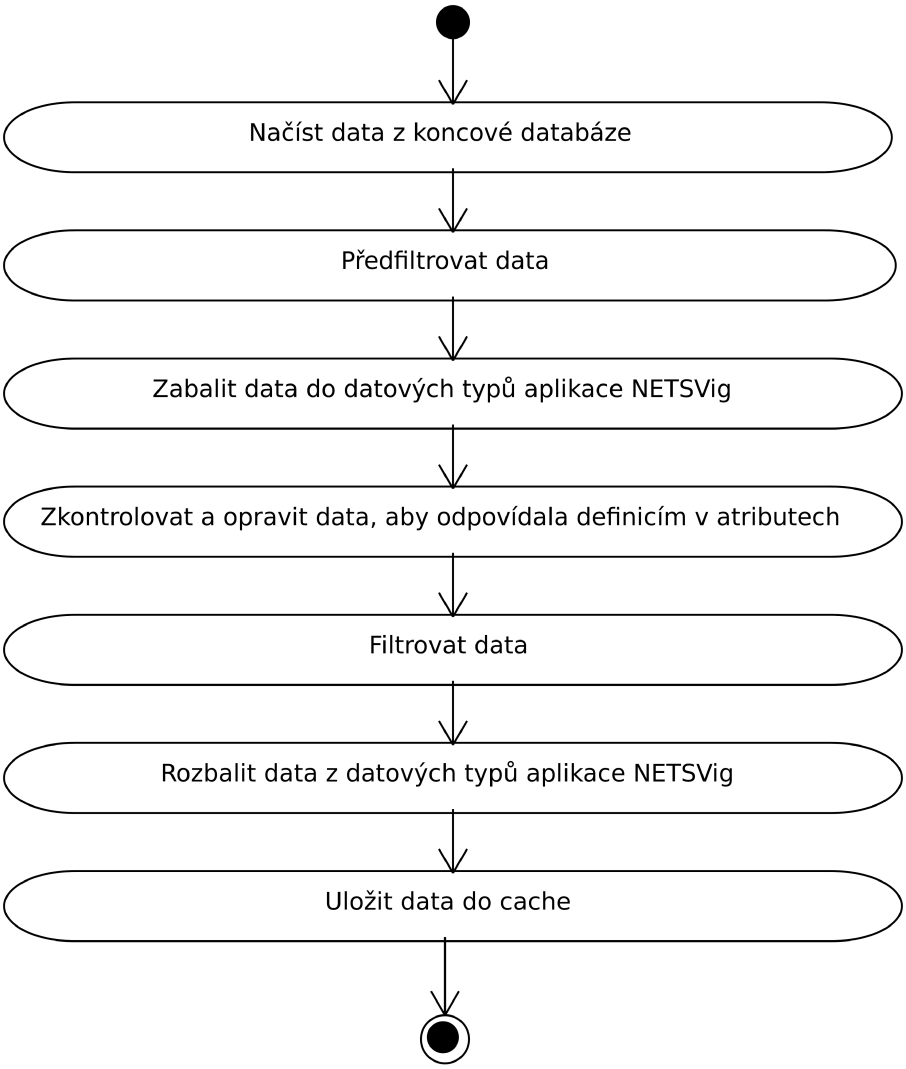
Kontrola dat zajišťuje ověření dat před jejich uložením do databáze. Umožňuje definovat řadu kontrol hodnot atributů a hesel.

4.6.4 Načítání uživatelů z koncové databáze do cache

Algoritmus načítání uživatelských dat z koncové databáze do cache je zobrazen na obrázku 4.6. Skládá se z následujících kroků:

1. Načtení dat z koncové databáze – plugin vrátí data ze spravované databáze v prostém formátu, kterému rozumí jazyk, v němž je napsaná aplikace NETSVig.
2. „Předfiltrace“ dat – v tomto kroku se provádí první filtrování. Je zde možné např. provést ořezání počátečních a koncových mezer textových řetězců.
3. Zabalení dat do datových typů aplikace NETSVig – každý plugin má definovaný tzv. wrapper (pouzdro). Je to objekt, který provádí obalování dat do datových typů aplikace NETSVig a jejich opětovné rozbalování na základě naprogramovaných pravidel. Tímto pravidlem může být např. převod hodnot 0 a 1 do datového typy Bool. Samotné operace provádí základní aplikace, plugin vrací pouze wrapper.
4. Kontrola a oprava dat – u vstupních dat se kontroluje, zda plugin vrátil všechny požadované hodnoty ve formátu, ve kterém je předepisuje atribut požadované hodnoty. Pokud data neprojdou kontrolou, aplikace se je pokusí opravit.
5. Filtrování dat – v tomto okamžiku jsou již data převedená do datových typů aplikace NETSVig. Tím je garantována správnost hodnoty atributu z pohledu datového typu. Během této filtrace je vhodné provést dodatečné úpravy dat jako převod jednotek (dělení a násobení čísla) apod.
6. Rozbalení dat z datových typů aplikace NETSVig – v tomto kroku se zavolá wrapper aplikace NETSVig, který provede rozbalení dat.

7. Uložení dat do cache – rozbalená data se předají driveru databáze a uloží se do cache.



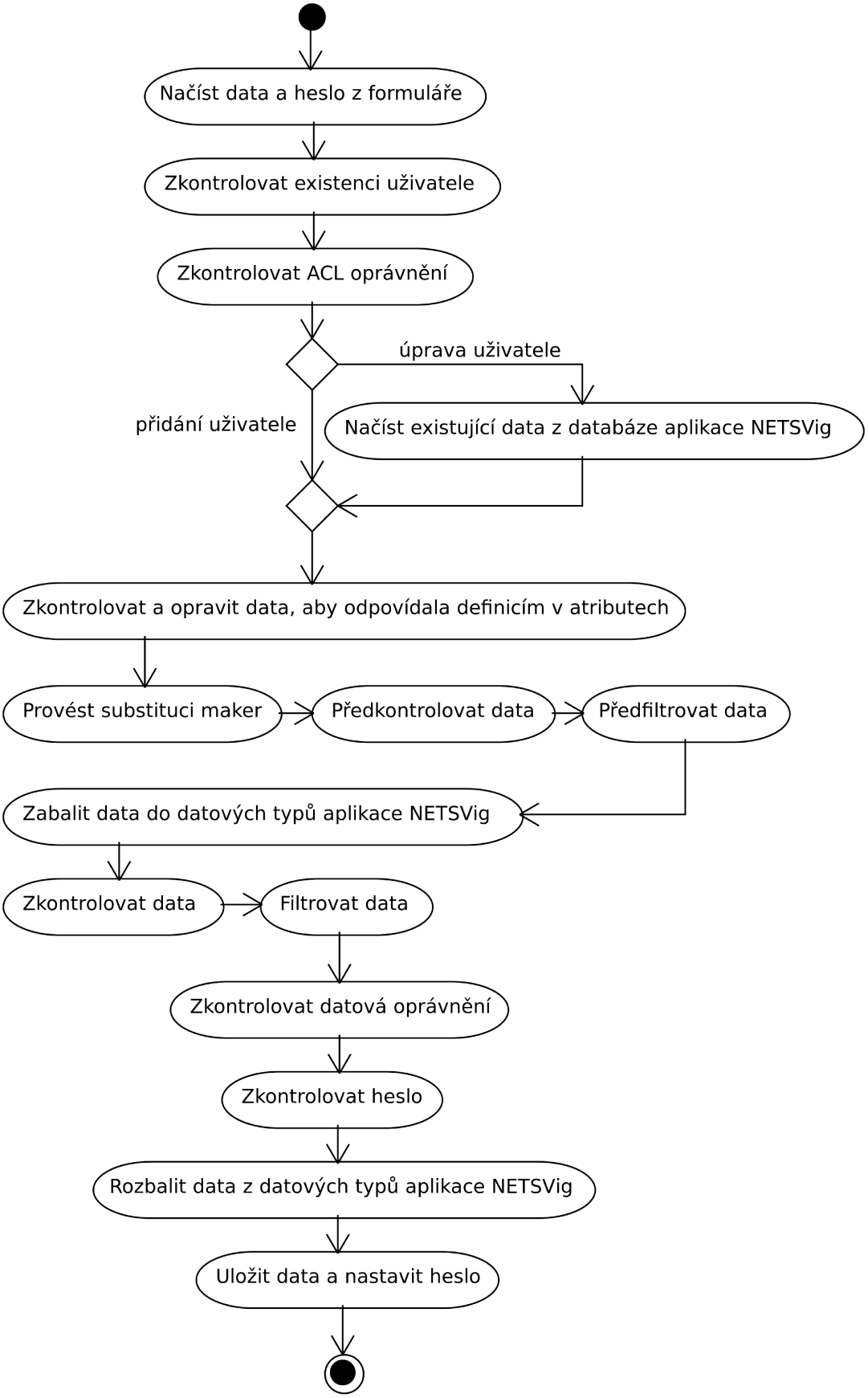
Obr. 4.6: Diagram aktivit: načítání dat z koncové databáze do cache

4.6.5 Vytváření a upravování uživatelů

Vytváření a upravování uživatelů je nejnáročnější datový proces v aplikaci. Jeho hlavní části jsou znázorněny na obrázku 4.7. Skládá se z následujících kroků:

1. Načtení dat a hesla z formuláře – uživatel (správce) skrze webové rozhraní vytvoří požadavek pro vytvoření nebo úpravu uživatelského účtu. Požadavek je nejčastěji vytvořen vyplněním a odesláním formuláře nebo importem z textového souboru. Při vytváření nového uživatelského účtu v centrální databázi aplikace NETSVig je nutné zadat heslo.
2. Kontrola existence uživatele – aplikace zkontroluje zda daný uživatel existuje. Uživatel musí existovat v centrální databázi aplikace NETSVig, jestliže je vytvářen nebo modifikován v koncové databázi. Přidávaný uživatel nesmí existovat v cílovém systému, upravovaný uživatel existovat v cílovém systému musí.
3. Kontrola ACL oprávnění – ověření, zda má přihlášený uživatel oprávnění provádět tento druh operace. Je zde vyžadováno povolovací oprávnění, tzn. že povolení k této operaci musí být explicitně uděleno.
4. Načtení existujících dat z databáze aplikace NETSVig – tento krok je nutné provést při modifikaci dat uživatele nebo při přidávání účtu do koncové databáze. Vedle dat načtených v bodě 1 se dočasně uloží data z databáze aplikace NETSVig. Díky tomu je v následujících krocích možné provádět kontrolu datových oprávnění i na základě těchto hodnot, využívat tato data pro substituci maker apod.
5. Kontrola a oprava dat – testuje se formát vstupních dat na základě definice atributů jednotlivých hodnot. Pokud data neprojdou kontrolou, aplikace se je pokusí opravit.
6. Substituce maker – všechna nalezená makra jsou nahrazena odpovídajícími hodnotami. Např. makro „%n“ je nahrazeno uživatelským jménem.
7. „Předkontrola“ dat – prvotní kontrola hodnot atributů vstupních dat. V tento okamžik ještě hodnoty nejsou omezeny svými datovými typy.
8. „Předfiltrace“ dat – v tomto kroku se provádí první filtrování. Zde rovněž nejsou data omezena svými datovými typy.

9. Zabalení dat do datových typů aplikace NETSVig – pro zabalení dat do datových typů se použije textový wrapper.
10. Kontrola dat – v tento okamžik se provádí kontrola hodnot atributů dat. Hodnoty jsou omezeny svými atributy, je zbytečné provádět kontrolu na takové hodnoty, které nemůže atribut daného datového typu nabývat.
11. Filtrace dat – poslední operace, kterou je možné modifikovat vstupní data. Je možné upravit hodnoty atributů pouze takovým způsobem, aby odpovídaly svým datovým typům.
12. Kontrola datových oprávnění – po předchozích úpravách s daty je možné až v tomto bodě provést kontrolu datových oprávnění. Pokud se uživatel upravuje, musí se zkontrolovat jak původní, tak konečné hodnoty uživatelských atributů.
13. Kontrola hesla – na základě vytvořených pravidel se prověří, zda zadané heslo odpovídá požadavkům v těchto pravidlech definovaným. Pravidlo může kontrolovat délku hesla, přítomnost nějakých znaků apod. Kontrola je prováděna stejným mechanismem, jako v bodě 7.
14. Rozbalení dat z datových typů aplikace NETSVig – zavolá se wrapper koncového systému. Ten rozbalí data tak, aby je bylo možné uložit.
15. Uložení dat a nastavení hesla – rozbalená data se předají pluginu koncové databáze. Plugin se postará o jejich uložení. V případě vytváření nebo modifikace centrálního účtu (účet uložený v databázi aplikace NETSVig) se o uložení dat postará samotná aplikace.

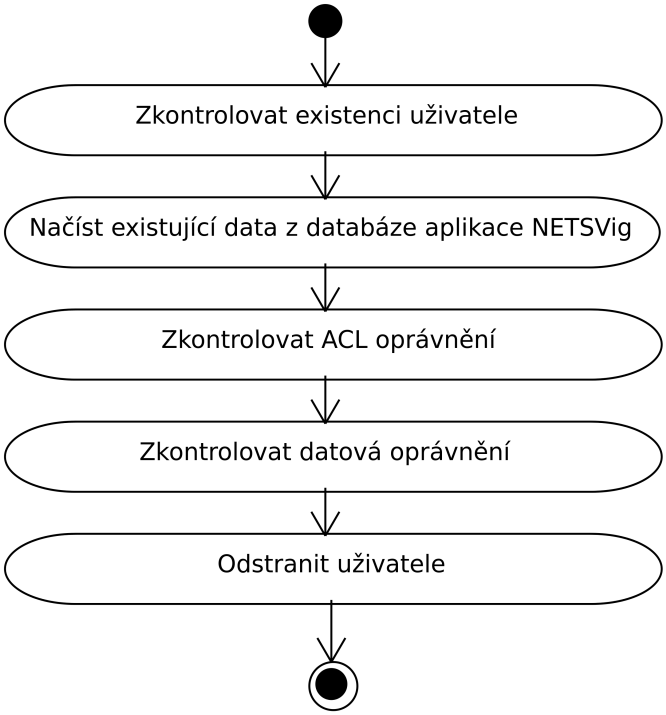


Obr. 4.7: Diagram aktivit: vytvoření a úprava uživatelů

4.6.6 Odstraňování uživatelů

Algoritmus odstraňování uživatelských účtů je znázorněn na obrázku 4.8. Skládá se z následujících kroků:

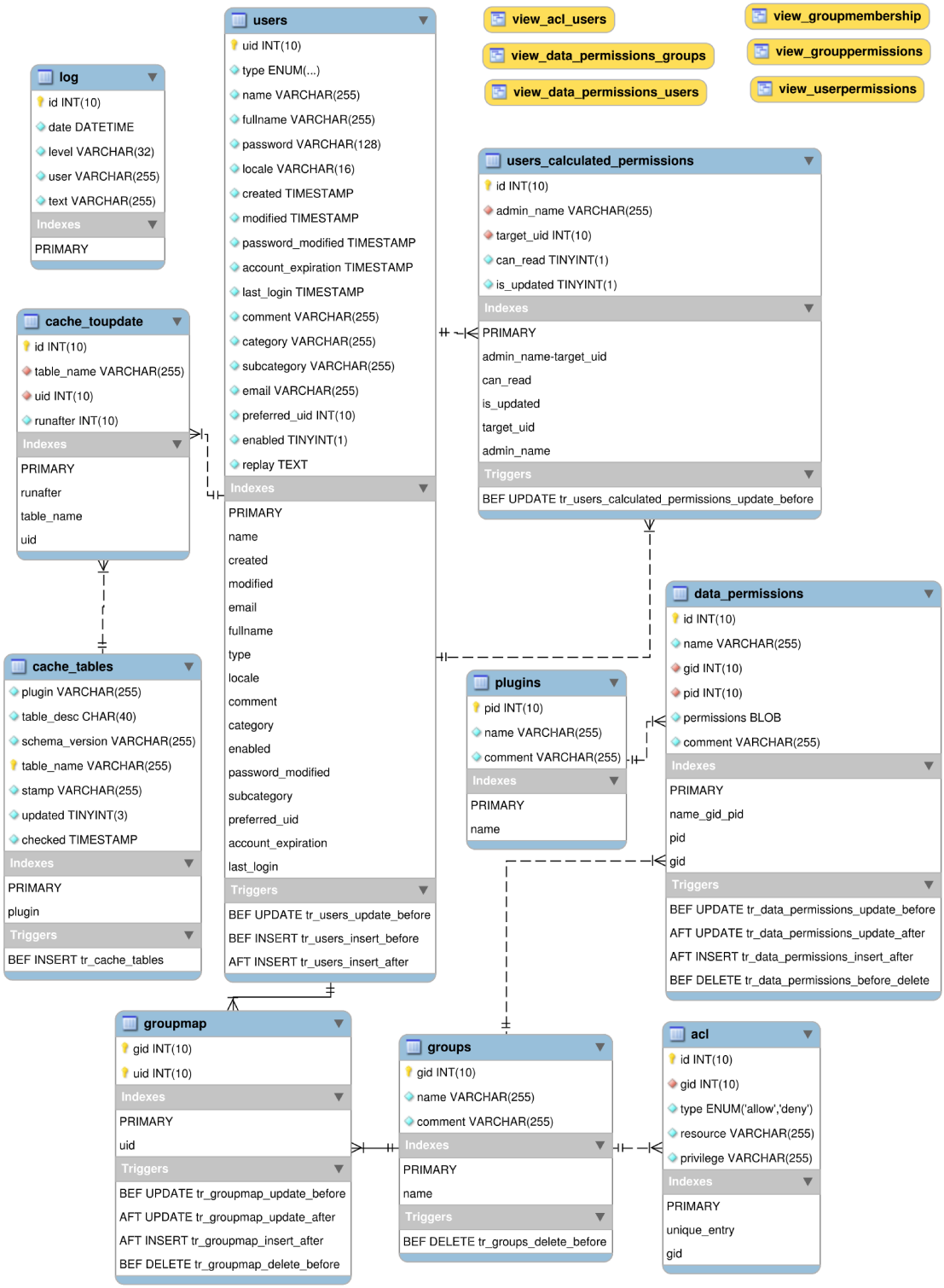
1. Kontrola existence uživatele – pouze existující uživatel může být odstraněn. Pokud se odstraňuje uživatel z koncové databáze, musí existovat v centrální databázi aplikace NETSVig.
2. Načtení existujících dat z databáze aplikace NETSVig – načtou se známé hodnoty atributů (vlastní uživatelská data aplikace NETSVig + cache).
3. Kontrola ACL oprávnění – kontrola, zda má uživatel udělené explicitní povolovací oprávnění k odstraňování uživatelů.
4. Kontrola datových oprávnění – na základě dat načtených v bodě 2 se provede kontrola oprávnění nad všemi atributy, které jsou součástí odstraňovaného uživatelského účtu.
5. Odstranění uživatelů – smazání uživatelského účtu z koncové databáze nebo z databáze aplikace NETSVig.



Obr. 4.8: Diagram aktivit: odstraňování uživatelů

4.7 Datový model

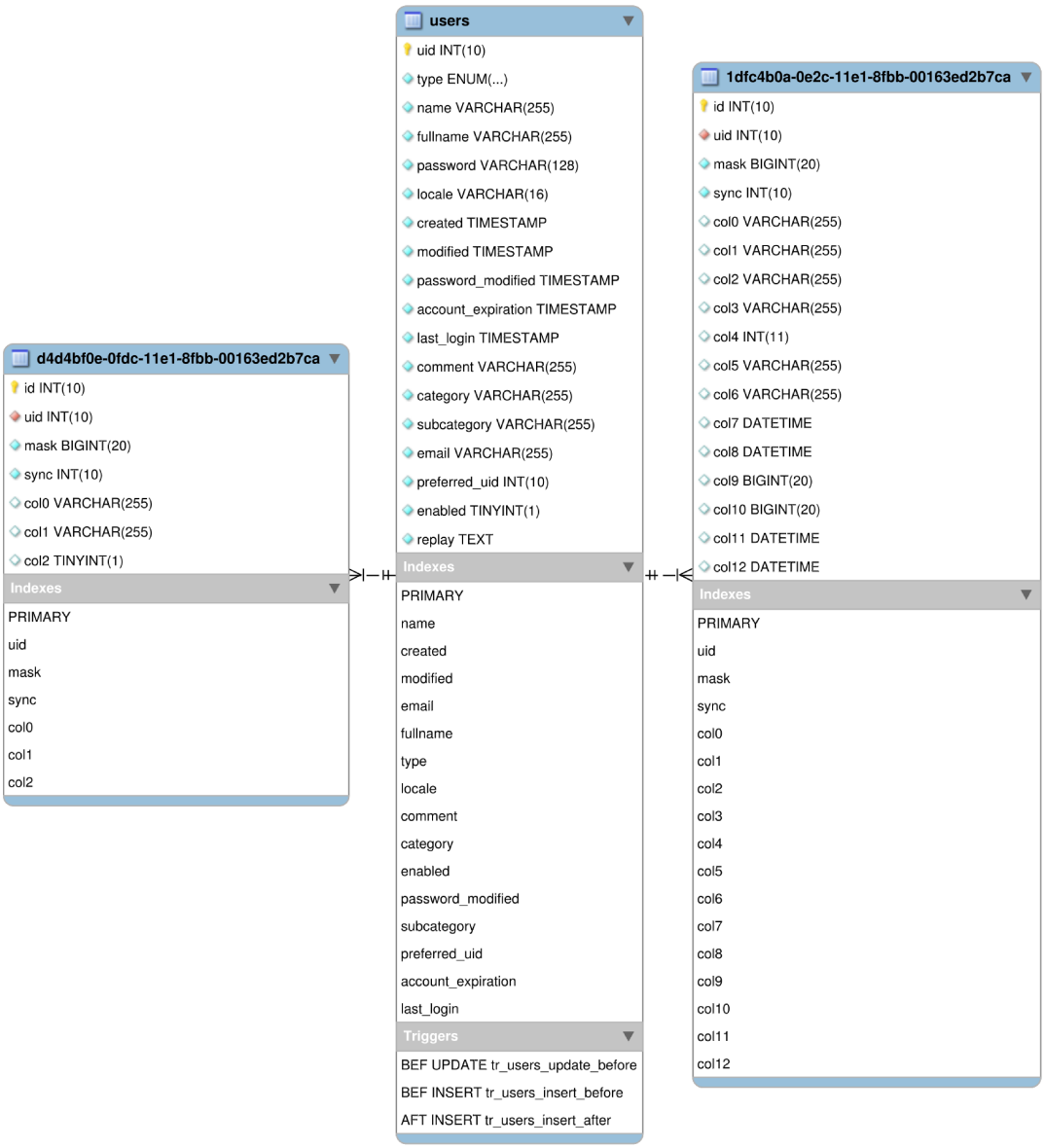
Aplikace používá pro ukládání svých dat relační databázi MySQL. Všechny tabulky používají úložiště InnoDB. V tabulce *users* jsou uložení všichni uživatelé, se kterými aplikace pracuje. Je to centrální úložiště uživatelů aplikace NETSVig. Tabulka *users_calculated_permissions* slouží pro ukládání spočítaných datových oprávnění pro čtení dat. Definuje zdrojového uživatele, cílového uživatele a oprávnění pro čtení. Vzhledem k tomu, že MySQL neumí definovat dva cizí klíče na jeden primární (respektive umí, ale nedokáže s nimi provádět kaskádové odstraňování a úpravy), odkazuje jeden cizí klíč na název uživatele, druhý na jeho unikátní identifikační číslo. Díky tomu, že jsou datová oprávnění pro čtení dat vypočítaná dopředu je možné provádět výběr dat s ohledem na tato oprávnění. Aplikace nemusí z výpisu odstraňovat celé uživatele, provede pouze odstranění nepovolených atributů. V tabulce *groups* jsou uložené skupiny. Jejich mapování na uživatele provádí tabulka *groupmap*. Ke každé skupině jsou připojena datová a ACL oprávnění. Datová oprávnění musejí kromě skupiny znát i plugin, na který se mají aplikovat (tabulka *plugins*). Všechny akce provedené v aplikaci se zaznamenávají do tabulky *log*. EER diagram databáze je zobrazen na obrázku 4.9.



Obr. 4.9: EER diagram databáze

V tomto schématu nejsou zobrazeny tabulky používané pro ukládání cache dat koncových databází.

Pro úložiště cache by bylo teoreticky nejoptimálnější (z pohledu čistoty návrhu) použít EAV (Entity attribute value) model. Aplikace však umožňuje vyhledávat a řadit data v různých koncových systémech najednou, proto by při jeho použití docházelo při výběru dat z databáze ke spojování velkého množství tabulek. Výběr dat by mohl trvat několik minut až hodin. Proto je cache implementována jako další tabulka v databázi. Obecně je chybou dynamicky měnit schéma databáze, nicméně v tomto případě tento postup přináší podstatné zrychlení, kdy výběr dat trvá podle velikosti databáze maximálně několik sekund (většinou zlomek sekundy). Pro každý plugin je automaticky založená nová tabulka pro ukládání cache. Název tabulky je vygenerován. Datové sloupce jsou pojmenovány jako *colN*, kde N je číslo datového sloupce. Nepoužívají názvy koncových systémů, ani názvy jejich atributů, protože názvy tabulek a sloupců jsou omezeny. Pro mapování koncového systému na tabulku cache se používá tabulka *cache_tables*. Data v cache tabulkách jsou uložena redundantně. Nepoužívá se dekompozice na několik dílčích tabulek. Dotaz nad jednou velkou tabulkou je rychlejší než dotaz nad spojením několika tabulek. Aplikace podporuje vícehodnotové atributy. Pro uložení dat se používá kartézský součin. Aby byla aplikace schopná detekovat jednotlivé hodnoty, obsahuje cache tabulka speciální sloupec, ve kterém se ukládá maska užitečných dat. Cache aktualizátor cache tabulky pravidelně automaticky aktualizuje. Tabulka *cache_toupdate* se používá pro dodatečné plánování aktualizací vybraných uživatelů. Diagram znázorňující vytvořené cache tabulky je zobrazen na obrázku 4.10.



Obr. 4.10: EER diagram cache tabulek

5 Implementace aplikace

5.1 Použité technologie

Aplikace je napsaná v jazyku PHP. Jedná se o skriptovací programovací jazyk, který se kompiluje automaticky během spuštění aplikace. Jazyk splňuje požadavky aplikace na snadnou rozšiřitelnost. Základní práci s ním zvládne i začínající programátor. Kromě toho aplikace ulehčuje vývoj rozšiřujících modulů a pluginů tak, aby ho zvládl i méně pokročilý vývojář. Jazyk PHP je dynamicky typovaný – datový typ proměnné se určí až po přiřazení hodnoty. To do značné míry zjednodušuje kód aplikace zejména v částech, v nichž se manipuluje s daty koncových systémů. Nevýhodou tohoto chování je, že kontrolu datových typů musí provádět sám programátor. Jazyk PHP prochází aktivním vývojem. Kromě přibývání nových vlastností se někdy mění i vlastnosti stávající. Proto je při vývoji aplikace důležité pracovat s oficiální dokumentací jazyka, kterou je možné nalézt na oficiálním webu PHP [9].

Pro ukládání dat používá aplikace databází MySQL. Je to poměrně rychlá databáze, která ale nenabízí příliš mnoho funkcí. V mnoha důležitých aspektech nabízejí konkurenční databáze minimálně stejné vlastnosti jako MySQL. Z tohoto pohledu je MySQL ideální databází pro vývoj aplikace. Aplikace pracující s MySQL půjde snadno portovat i na ostatní databáze.

Webové části aplikace jsou napsány tak, aby fungovaly dobře pod webovým serverem Apache httpd. Tento webový server implementuje podporu jazyka PHP buď přímo modulem mod_php nebo prostřednictvím CGI. Aplikace podporuje oba způsoby implementace.

Webové stránky webového rozhraní jsou generovány jako XHTML Basic 1.1. Na stránkách jsou použity kaskádové styly CSS. Stránky neobsahují vždy validní kód. Někdy jsou použité některé vlastnosti z HTML 5 (např. zakázání automatického vyplňování formuláře) a CSS3 (zaoblení okrajů). Webové prohlížeče, které tyto vlastnosti nepodporují je nezobrazí. Stránky jsou generovány pomocí PHP tak, aby bylo webové rozhraní použitelné i v prohlížečích, které mají zakázané použití JavaScriptu. JavaScript se sice v generovaných stránkách značně používá, základní funkcionalita webu je však dostupná i s vypnutým JavaScriptem. Pro generování a práci s některými

formuláři je použitý PHP framework Nette [10], pro snadnější tvorbu JavaScript kódu je v omezené míře použit JavaScript framework jQuery [11].

Backend aplikace vyžaduje periodické spouštění PHP skriptu pro aktualizaci cache. Lze jej provozovat ve všech systémech, které umožňují periodické spouštění úloh a mají nainstalovaný PHP interpret. Optimální je použít nějakou distribuci operačního systému Linux. Aplikace byla testována v distribucích Ubuntu, Debian a CentOS.

5.2 Softwarové požadavky

Aplikace používá některé nové vlastnosti jazyka PHP. Jedná se především o jmenné prostory. Vyžaduje minimální verzi PHP 5.3.2. V linuxových distribucích je PHP obvykle dostupné jako balíček. Distribuce vydává pravidelně bezpečnostní záplaty tohoto balíčku. Po aktualizaci záplatovaného balíčku by měl správce zkontrolovat funkčnost aplikace. Občas se vlivem záplatování bezpečností chyby zanese do interpretu PHP nová chyba. Např. záplata *php_crypt_revamped.patch* v distribuci Debian squeeze částečně rozbila funkčnost algoritmu extended DES ve funkci crypt, která se často používá pro vytváření a kontrolu hesel. Aplikace vyžaduje vypnutí některých zastaralých, ale donedávna stále podporovaných vlastností PHP. Jedná se především o safe mode (omezí PHP, ale nezvyšuje zabezpečení serveru) a magic quotes (automatické escapovaní vstupních řetězců). Obě tyto vlastnosti byly odstraněny až v PHP 5.4.0. PHP interpret musí mít nastavené následující parametry:

- maximální množství přidělené paměti (memory_limit): alespoň 82 MiB,
- maximální velikost POST requestu (post_max_size): alespoň 80 MiB,
- maximální velikost nahrávaného souboru (upload_max_filesize): alespoň 64 MiB,
- povolit práci s URL jako se soubory (allow_url_fopen).

PHP musí být zkompilováno s podporou iconv (převod znakových sad), MySQL a dalších databází, ke kterým se bude aplikace připojovat jako ke vzdáleným systémům. Webové rozhraní pro uživatele navíc vyžaduje nainstalované rozšíření PECL http (kvůli komunikaci s webovým rozhraním pro správce).

Databázový server MySQL ve verzi 5.1 a vyšší by měl běžet na stejném fyzickém stroji jako webová aplikace pro správce. Práce s lokálním databázovým

strojem je rychlejší a bezpečnější. Aplikace nepodporuje šifrování komunikace s databázovým strojem, protože tato funkčnost bývá v PHP pravidelně rozbitá.

Webový server Apache httpd by měl být alespoň verze 2.2. Musí být zkompilován s podporou SSL (kvůli podpoře HTTPS). Webové aplikace vyžadují použití protokolu HTTPS. Pokud se k aplikaci přistupuje nezabezpečeným protokolem HTTP, aplikace se pokusí přesměrovat na zabezpečenou verzi protokolu. Jestliže tuto zabezpečenou verzi webový server nepodporuje, aplikace nebude fungovat. Kdyby aplikace podporovala nezabezpečenou verzi HTTP, umožňovala by nešifrovaný přenos hesel.

5.3 Struktura aplikace

5.3.1 Struktura kódu

Aplikace je napsaná stylem kombinujícím procedurální a objektové programování. Jádro aplikace je procedurální a inicializuje jednotlivé objekty. Všechny části aplikace (aktualizátor cache a obě webové rozhraní) používají stejnou strukturu kódu. Webové rozhraní pro uživatele používá jiné konfigurační soubory, některé jiné aplikační soubory a neobsahuje většinu z aplikačních tříd webového rozhraní pro správce. U webových rozhraní provádí načítání ostatních částí aplikace soubor index.php, u aktualizátoru cache je hlavním aplikačním souborem cache_update.php. Struktura aplikace se odráží i v adresářové struktuře aplikačních souborů. Podle potřeby se nachází v jednotlivých adresářích soubor .htaccess, který určuje chování webového serveru. Konkrétně se zde používá pro zakázání přímého přístupu do daného adresáře např. přes URL webového prohlížeče. Následující text popisuje strukturu webového rozhraní pro správce a aktualizátoru cache (používají totožné aplikační soubory). Webové rozhraní pro uživatele je jen zjednodušenou verzí webového rozhraní pro správce.

Nejvyšší adresář

Soubor cache_update.php inicializuje aplikaci cache aktualizátoru, index.php inicializuje webové rozhraní, webapi.php poskytuje Web API webovému rozhraní pro uživatele. Aplikace podporuje hezká URL pomocí jejich přepisování. Automaticky detekuje jejich podporu ze strany webového serveru. Pro tuto detekci používá soubor

mod_rewrite.test. V souboru .htaccess je nadefinováno přesměrování adresy „system/nonexisting.file“ na soubor mod_rewrite.test. Pokud se aplikaci na této adrese podaří soubor otevřít, je podpora přepisování URL na straně webového serveru zapnuta a aplikace jí použije.

Adresář config

Obsahuje konfigurační soubory aplikace.

Adresář css

Obsahuje soubory kaskádových stylů webové aplikace.

Adresář images

Obsahuje obrázky používané webovým rozhraním aplikace.

Adresář includes

V tomto adresáři se nacházejí všechny ostatní neobjektové části aplikace. V souboru sql_connection.php dochází ke spojení s databází aplikace. Soubor login_screen.php generuje přihlašovací obrazovku webové aplikace pro správce. Po úspěšném přihlášení načte soubor logged_base.php celé uživatelské rozhraní. Webové rozhraní pro uživatele obsahuje pouze soubor main_screen.php, který kromě generování webové stránky vykonává i aplikační logiku.

Adresář includes/tasks

Obsahuje soubory základních úloh ve webovém rozhraní. Každá úloha si sama řídí navigaci na koncové akce (např. úloha administrace a akce editace ACL). Cílovou URL získává ze statické třídy Route. Soubory s definicí úlohy a/nebo akce částečně nebo úplně generují HTML kód aplikace. U částečného generování předávají HTML obsah šabloně. Při plném generování kód přímo vracejí. Zároveň je v těchto souborech umístěna hlavní aplikační logika webového rozhraní.

Adresář includes/templates

Obsahuje soubory šablon. Šablona obsahuje základní HTML kód a dotazem na instanci třídy Template získává vlastní HTML obsah (vč. titulku stránky). Pokud instance třídy Template nevrátí žádný obsah, je provedeno přesměrování na hlavní zobrazení webového rozhraní. Hlavním zobrazením je seznam uživatelských účtů. Tento

seznam nepoužívá kvůli úspoře paměti šablony a vygenerovaný obsah přímo vrací webovému serveru. Díky tomu je možné vygenerovat tisíce řádků HTML kódu s využitím malého množství operační paměti.

Adresář libs

V adresáři libs se nacházejí aplikační třídy objektové části aplikace. Jednotlivé podadresáře adresáře libs odpovídají jmenným prostorům vlastních tříd.

Adresář locale

Aplikace obsahuje v kódu anglické texty. Pro lokalizaci prostředí se používá projekt gettext [12]. Funkce pro překlad jsou dostupné přímo v PHP. Soubory jednotlivých překladů jsou v adresáři locale. Nyní je k dispozici vedle původních anglických textů pouze jejich česká lokalizace.

Adresář logs

Do adresáře logs se ukládají soubory logů. Aplikace upřednostňuje logování do databáze, je ale možné ukládat logy i do textových souborů. Textové soubory neumožňují paralelní zápis, proto může být dočasně relace jednoho uživatele blokována jinou relací. Adresář logs se používá hlavně pro ukládání ladících výstupů (debug) aplikace. Každý ladící výstup se zapisuje do samostatného souboru, proto nedochází k vzájemnému blokování uživatelských relací.

Adresář script

Obsahuje JavaScriptové soubory webového rozhraní.

5.3.2 Konfigurační soubory

Konfigurační soubory aplikace se nacházejí v adresáři config.

Konfigurace webového rozhraní pro uživatele

Hlavním konfiguračním souborem je config.main.php. Konfiguruje spojení k webovému rozhraní pro správce a zapíná ladící výstupy.

Konfigurace webového rozhraní pro správce a cache aktualizátoru

I zde je hlavním konfiguračním souborem config.main.php. Konfiguruje spojení k databázi, zapíná ladící výstupy, nastavuje parametry uživatelského prostředí

webového rozhraní, konfiguruje Web API a provádí přizpůsobení konfigurace pluginů. V souboru logging.php se nastavuje logování. Statické třídě Logging se předají objekty úložišť jednotlivých logů. Pokud se třídě Logging nepředají žádné objekty, je logování vypnuto. V souboru modules.php se inicializují rozšiřující moduly. V souboru plugins.php se provádí inicializace a konfigurace pluginů.

5.4 Hlavní aplikační třídy

Základní kód aplikace je rozdělen do 83 hlavních tříd. Zde budou uvedeny ty nejpodstatnější. Třídy jsou uvedeny se jmennými prostory (namespace) ve formátu namespace\třída.

5.4.1 Základní třídy

Netsvig\Object

Abstraktní třída, od které jsou odvozeny všechny ostatní.

Netsvig\Singleton

Umožňuje volat objekty typu singleton. Singleton objekty tuto třídu rozšiřují.

Netsvig\Autoloader

Soubor s definicí této třídy musí být vnořen (include) v inicializačním souboru aplikace. Autoloader slouží k dynamickému načítání deklarací tříd. Autoloader při inicializaci načte mapu umístění ostatních tříd. Při prvním použití nějaké třídy provede PHP pomocí Autoloaderu lokalizaci a nahrání souboru, ve kterém je volaná třída deklarovaná. Pokud se soubor se třídou nenachází v mapě, Autoloader se pokusí třídu vyhledat mezi moduly a pluginy.

Netsvig\Configuration

Načítá a obaluje konfigurační soubor config.main.php. Aplikace se na konfiguraci dotazuje třídy Configuration. Při volání funkce třídy Configuration může být předána i výchozí hodnota konfiguračního parametru. Pokud nebyl konfigurační parametr deklarování v souboru config.main.php, funkce vrátí výchozí hodnotu.

Netsvig\Environment

Inicializuje uživatelské prostředí webového rozhraní. Testuje konfigurační parametry PHP, některé parametry nastavuje, provádí přesměrování HTTP na HTTPS a nastavuje lokalizaci.

Netsvig\Sessions

Definuje parametry sessions a obaluje je.

5.4.2 Třídy pro deklaraci atributů a manipulaci s daty spravovaných systémů

Netsvig\Data\Attributes

Obsahuje instance objektů, které představují jednotlivé atributy spravovaných systémů. Instance třídy Attributes důkladně popisuje koncový systém. Určuje, které atributy jsou pro čtení, které pro zápis (neukládají se do lokální cache) a které se používají pouze jako pomocné (ukládají se jen do cache, jsou užitečné při kontrole datových oprávnění). Určuje, zda koncová databáze umožňuje spravovat hesla. Deklaruje, které atributy jsou povinné, které mohou být vícehodnotové, nastavuje filtrační a validační pravidla.

Netsvig\Data\DataTypes\DataType

Abstraktní třída, kterou rozšiřují třídy definující datové typy (např. třída Int8).

Netsvig\Data\Rules\Filtering

Přidává filtry dat konkrétního atributu a provádí filtrování.

Netsvig\Data\Rules\Validation

Přidává pravidla pro kontrolu dat konkrétního atributu a provádí kontrolu.

Netsvig\Data\Wrapper\Common

Univerzální wrapper dat. Zabaluje data do datových typů aplikace NETSVig a provádí jejich rozbalení. Aplikace obsahuje několik wrapperů.

5.4.3 Třídy pro deklaraci modulů

Netsvig\Modules\StdModule

Abstraktní třída, kterou rozšiřují třídy vlastních modulů. Implementuje funkce pro registraci modulů v instanci třídy Module.

Netsvig\Module

Registruje různé funkce modulů a provádí jejich volání.

5.4.4 Třídy pro deklaraci pluginů

Netsvig\Plugins\StdPlugin

Abstraktní třída, kterou rozšiřují třídy vlastních pluginů.

Netsvig\Plugin

Inicializuje pluginy, vytváří pro ně nové cache tabulky v databázi a provádí překlad jmen pluginů a atributů na názvy tabulek a sloupců.

5.4.5 Třídy pro zabezpečení aplikace

Netsvig\Security\User

Zajišťuje přihlašování uživatelů typu admin nebo superadmin do uživatelského rozhraní pro správce a umožňuje jim změnu hesla. Automaticky ukládá uživatelskou relaci v případě nečinnosti. Instance objektu User obsahuje reference na objekty ACL a datových oprávnění.

Netsvig\Security\User\Permission

Instance této třídy udržuje ACL oprávnění nahraná z databáze. Obsahuje dvě funkce: jePovoleno, jeZakázáno.

Netsvig\Security\User\DataPermission

Instance této třídy obsahuje datová oprávnění nahraná z databáze. Oprávnění jsou uložena jako prostá vícedimenzionální pole. Do databáze se ukládají jako serializovaná. Při načítání oprávnění se kontroluje jejich formát. Oprávnění, která nejsou v platném formátu jsou zahozena. Před ověřováním oprávnění se instanci třídy

DataPermission nastaví kontrolovaná data a provede se dotaz na oprávnění pro jednotlivé atributy.

Netsvig\Security\Password

Třída Password slouží pro manipulaci s hesly. Podporuje celou řadu běžně používaných formátů hesel, kterou lze rozšířit pomocí modulů. Umožňuje pracovat s hesly, která jsou v různých znakových sadách.

Netsvig\Security\Token

Generuje a ověřuje bezpečnostní token. Jedná se o bezpečnostní prvek webového rozhraní. Znemožňuje provedení útoku CSRF (Cross-site request forgery) vygenerováním náhodného řetězce, který je po každém dotazu GET nebo POST ověřován proti hodnotě uložené v sessions.

5.4.6 Třídy pro deklaraci úložišť

Netsvig\Storage

Pomocí zvoleného ovladače inicializuje datové úložiště. Aplikace obsahuje ovladače pro textový soubor a databázový systém. Ovladač pro databázový systém může použít generátor základních dotazů, který na základě [13] podporuje databáze: MySQL, PostgreSQL, Firebird, Oracle, IBM DB2, MS SQL, SQLite.

Netsvig\Storages\Driver

Abstraktní třída, která implementuje rozhraní IDriver. Toto rozhraní musí implementovat všechny ovladače.

5.4.7 Třídy pro manipulaci se spravovanými uživateli

Netsvig\Users

Deklaruje atributy centrální databáze. Provádí dotazy nad daty uloženými v cache. Obsahuje funkce pro vytváření, odstraňování a úpravu uživatelů v centrální databázi i koncových systémech. Pro vytváření, odstraňování a úpravu uživatelů v koncových systémech používá funkcí pluginů. Pluginy implementují jen funkce pro manipulaci s daty. Kontrolu dat a oprávnění provádí instance třídy Users.

Netsvig\CacheUpdater

Nastavuje plány aktualizací cache a provádí vlastní aktualizaci.

5.4.8 Třídy pro generování webového rozhraní

Netsvig\Web\Route

Třída pro navigaci v rozhraní. Detekuje podporu hezkých URL. Na základě toho umožňuje vytvářet webové odkazy. Analyzuje aktuální URL a vrací cestu rozdělenou na úlohu, akci a další parametry.

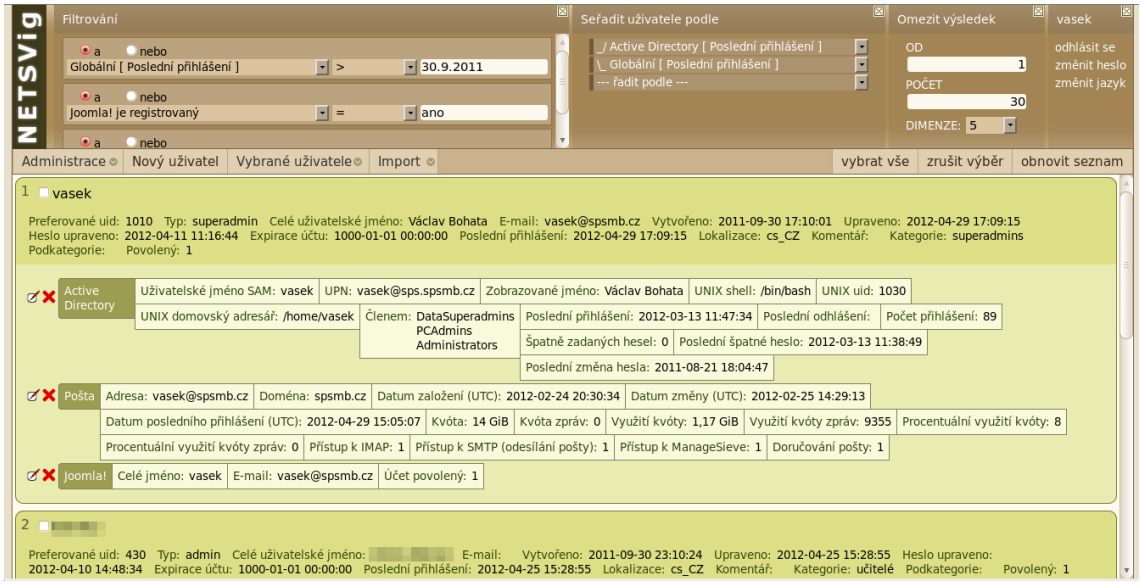
Netsvig\Web\Template

Udržuje deklaraci panelů a menu ve webovém rozhraní. Obsahuje funkce pro přidávání meta tagů, JavaScript a CSS odkazů. Udržuje vlastní HTML obsah. Na základě nastavených hodnot v instanci třídy Template provede šablona vygenerování stránky.

5.5 Uživatelské rozhraní

5.5.1 Uživatelské rozhraní aplikace pro správce

Po přihlášení se správce ocitne ve výchozí úloze (zobrazení) aplikace. Ta nabídne správci výpis všech uživatelů spravovaných aplikací NESVig. V této úloze je možné provádět filtrování a řazení uživatelů a volit počet zobrazených atributů. Počet zobrazených atributů se nastavuje pomocí tzv. dimenze. Každý atribut má deklarovanou dimenzi. Zobrazují se atributy, jejichž dimenze je rovna nebo menší nastavené dimenzi. Ukázka[1] zobrazení výchozí úlohy je na obrázku 5.1.



Obr. 5.1: Výchozí zobrazení aplikace

V tomto případě se zobrazují uživatelé, kteří se přihlásili do webového rozhraní po 30. 9. 2011 a mají účet v koncovém systému Joomla!. Takto zobrazení uživatelé se seřadí vzestupně podle data posledního přihlášení v koncovém systému Active Directory. Uživatelé se stejným datem se seřadí sestupně podle data posledního přihlášení do webového rozhraní. Zobrazují se uživatelé 1-30, dimenze zobrazení je nastavena na 5.

V menu si správce vybírá požadovanou akci. Může založit nového uživatele, smazat vybrané z koncového systému, databáze aplikace NETSVig nebo všech systémů. Může přidat vybrané uživatele do koncového systému nebo jim hromadně nastavit hesla.

1 Některé texty jsou rozostřené kvůli ochraně osobních údajů.

Na obrázku 5.2 je ukázka přidávání uživatele do systému Active Directory.

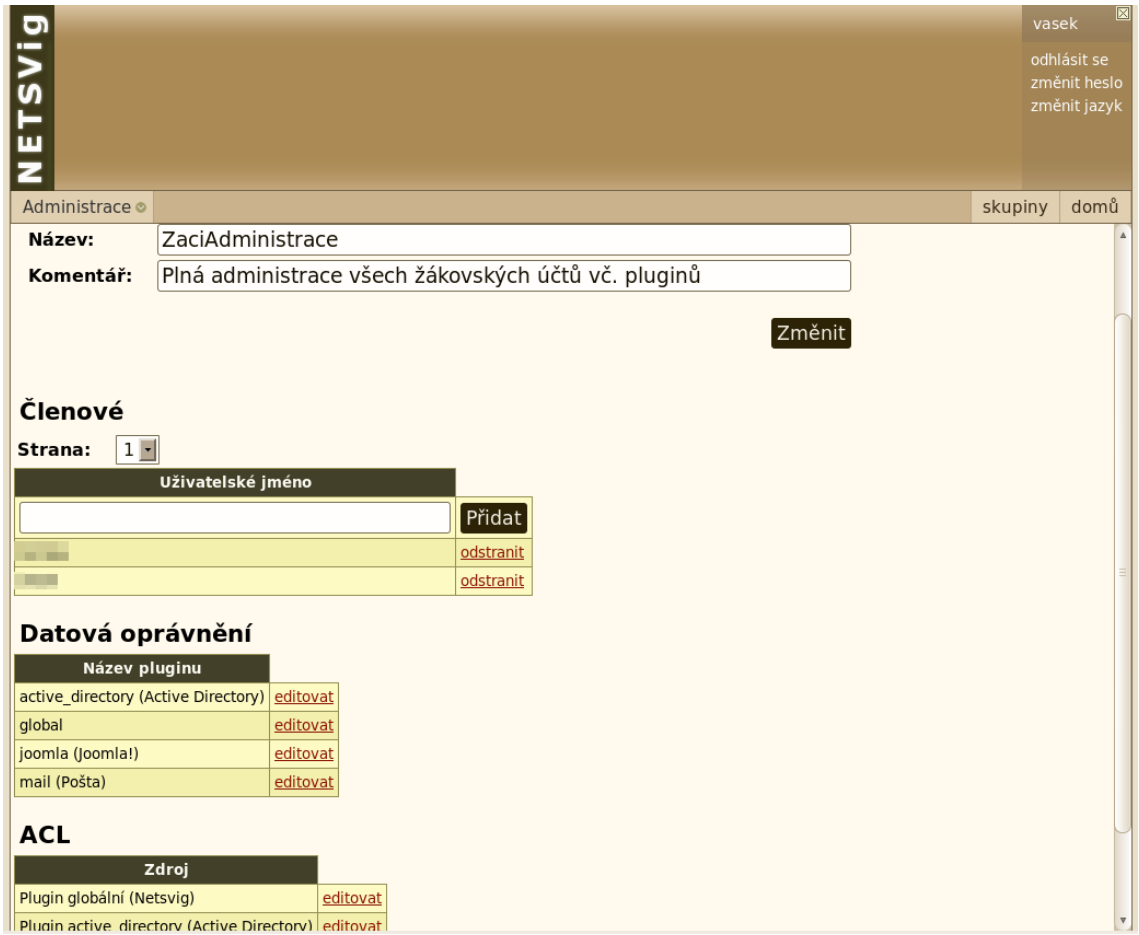
Obr. 5.2: Přidávání uživatele do systému Active Directory

Prostředí aplikace NETSVig se snaží akci co nejvíce urychlit. Na základě definovaných atributů jsou hodnoty předvyplněny makry, která se přeloží na odpovídající hodnoty atributů systému NETSVig. Atributy označené hvězdičkou jsou povinné. Nabídka „Použít pro uživatele“ má za cíl ještě zrychlit vytváření uživatelů v koncových systémech. Správce vybere v hlavním zobrazení několik uživatelů, které chce založit v koncových systémech. V nabídce „Použít pro uživatele“ vybere několik z nich, kterým vyplní požadované hodnoty a poté je založí. Založení uživatelé zmizí z nabídky. Správce upraví nějakou hodnotu, vybere cílové uživatele v „Použít pro uživatele“ a ty opět založí. Díky tomu je možné rychle přidat vysoké množství uživatelů a různým vybraným skupinám rychle pozměnit nějaké hodnoty. Vedle hodnot jednotlivých atributů se zobrazují pravidla pro jejich vytváření. Tato pravidla kontroluje aplikace při zakládání uživatele.

Podobné uživatelské rozhraní nabízí aplikace i pro úpravu uživatelů. Odstraňování uživatelů je možné rovněž provádět hromadně. Zakládání nových uživatelů do aplikace NETSVig lze urychlit jejich importem z textového souboru.

Uživatel typu superadmin může provádět administraci uživatelských skupin. Každé skupině nastavuje ACL oprávnění, datová oprávnění a seznam členů viz. obrázek 5.3[1].

1 Některé texty jsou rozostřené kvůli ochraně osobních údajů.



Obr. 5.3: Editace skupiny

Editace ACL oprávnění je zobrazena na obrázku 5.4

Obr. 5.4: Editace ACL oprávnění

Položka „GLOBÁLNÍ PŘÍSTUP“ definuje přístup k celému zdroji. Jednotlivá oprávnění definují povolené operace se zdrojem. Editace jednoho z datových oprávnění pro koncovou databázi je znázorněna na obrázku 5.5.

Obr. 5.5: Editace datového oprávnění

Oprávnění je definováno pro vybrané atributy. U jednotlivých atributů je znázorněno, zda se jejich hodnoty ukládají do cache (C), zda je atribut pro čtení (R) a jestli je zapisovatelný (W). Porovnávání hodnot se nastavuje v textových polích ve sloupci „Odpovídá“. Každé nové porovnávání je na novém řádku. Deklaruje se ve tvaru: operátor_porovnání porovnávaná_hodnota (např. < 200). Pokud je textové pole prázdné, neprovádí se žádné porovnání a vybrané operace nad hodnotami zvolených atributů jsou povoleny pro všechny hodnoty.

5.5.2 Uživatelské rozhraní aplikace pro koncové uživatele

Jediný účel tohoto rozhraní je povolit uživatelům změnu hesel ve spravovaných systémech. Uživatelské rozhraní pro koncové uživatele je zobrazeno na obrázku 5.6.

Obr. 5.6: Uživatelské rozhraní pro koncové uživatele

6 Závěr

Cílem práce bylo navrhnout webovou aplikaci pro centralizovanou správu uživatelských účtů v různých koncových databázích. Aplikace se zaměřuje především na rychlou hromadnou správu většího množství uživatelů. Je plně rozšiřitelná pomocí modulů a pluginů. Pluginy zastávají roli konektorů do spravovaných databází (koncových systémů). Moduly nabízejí rozšiřitelnost již existujících částí aplikace nebo umožňují vytvářet nové funkční celky. Aplikace podporuje komplexní systém oprávnění. Definuje ACL a datová oprávnění. ACL oprávnění kontrolují přístup ke zdrojům. Datová oprávnění omezují přístup k atributům uživatelů na základě jejich aktuálních hodnot. Všichni uživatelé, které aplikace spravuje, musejí být založeni v centrální databázi, tj. v databázi aplikace. Identity uživatelů v koncových databázích jsou považovány za stejné, pokud se shodují jejich uživatelská jména (případně jiné podobné atributy) s centrální databází.

Aplikace se skládá ze tří samostatných komponent. Jsou jimi webové rozhraní pro správce, webové rozhraní pro uživatele a cache aktualizátor. Cache je tabulka v databázi aplikace. Používá se pro ukládání hodnot atributů jednotlivých uživatelů z koncových systémů. Aplikace tak má k dispozici částečnou kopii spravovaných uživatelských databází. Může s nimi pracovat najednou, vyhledávat v nich a řadit je i pokud by byla spravovaná databáze dočasně nedostupná. Webové rozhraní pro správce je hlavním rozhraním aplikace. Provádí se v něm samotná správa uživatelů ve všech spravovaných databázích. Webové rozhraní pro uživatele pouze nabízí koncovým uživatelům možnost změnu vlastního hesla.

Aplikace je implementovaná v jazyce PHP, jako centrální databázi používá MySQL a webová rozhraní fungují pod webovým serverem Apache httpd.

Aplikace je plně funkční a splňuje všechny požadavky určené při návrhu. V současné době je nasazená na Střední průmyslové škole v Mladé Boleslavi, kde spravuje několik reálných uživatelských databází.

Po osobních zkušenostech z reálného nasazení bych do webové aplikace integroval více JavaScript a použil AJAX. Tím by se zpříjemnilo ovládání za cenu nefunkčnosti ve webových prohlížečích s vypnutým JavaScriptem. Rovněž použití relační databáze MySQL se ukázalo jako nevhodné. Pro běžné ukládání dat tato

databáze vyhovuje, ale u složitější aplikace, kdy je potřeba část aplikační logiky přenést na databázi, je tato databáze pomalá a spoustu věcí nezvládá vůbec. Kritický problém představuje neschopnost spouštění triggerů po změně hodnoty cizího klíče. Tuto funkčnost bylo nutné nahradit kaskádou triggerů.

Při dalším vývoji aplikace bych nahradil MySQL databází PostgreSQL. Pro generování HTML stránek bych zvolil HTML 5. Webová rozhraní by byla závislá na JavaScriptu. Rovněž bych implementoval automatickou synchronizaci vybraných atributů mezi zvolenými koncovými systémy.

Seznam použité literatury

[1] GROFF, James R. a Paul N. WEINBERG. *SQL*: *kompletní průvodce*. Vyd. 1. Brno: CP Books, 2005, 936 s. ISBN 80-251-0369-2.

[2] MySQL 5.0 Reference Manual: Privileges Provided by MySQL. *MySQL Documentation*: *MySQL Reference Manuals* [online]. 29957. vyd. 7.4.2012 [cit. 2012-04-07]. Dostupné z: http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html

[3] MySQL 5.1 Reference Manual: Privileges Provided by MySQL. *MySQL Documentation*: *MySQL Reference Manuals* [online]. 29938. vyd. 6.4.2012 [cit. 2012-04-07]. Dostupné z: http://dev.mysql.com/doc/refman/5.1/en/privileges-provided.html

[4] STĚHULE, Pavel. PostgreSQL. In: *PostgreSQL* [online]. 27.2.2012 [cit. 2012-04-09]. Dostupné z: http://postgres.cz/wiki/PostgreSQL

[5] BENÁK, Karel. *Použití adresářových služeb v informačních systémech*. Praha, 2004. Dostupné z: http://ldap.benak.net/diplom.pdf. Diplomová práce. České vysoké učení technické v Praze, Fakulta stavební, Katedra systémového inženýrství.

[6] BOUŠKA, Petr. Active Directory komponenty - domain, tree, forest, site. *LDAP a Active Directory* [online]. [cit. 2012-04-14]. Dostupné z: http://www.samuraj-cz.com/clanek/active-directory-komponenty-domain-tree-forest-site

[7] LÍZNER, Martin. Identity management – centrální správa uživatelských účtů. *Security World.cz | Deník o bezpečnosti pro IT profesionály* [online]. 24. 5. 2010[cit. 2012-04-22]. Dostupné z: http://securityworld.cz/securityworld/identity-management-centralni-sprava-uzivatelskych-uctu-2780

[8] *Roundcube webmail* [online]. © 2012 [cit. 2012-04-25]. Dostupné z: http://www.roundcube.net

[9] *PHP: Hypertext Preprocessor* [online]. © 2001-2012 [cit. 2012-04-29]. Dostupné z: http://www.php.net

[10] *Quick 'n' Comfortable Web Development in PHP | Nette Framework* [online]. © 2008, 2012 [cit. 2012-04-29]. Dostupné z: http://nette.org/en

[11] *JQuery: The Write Less, Do More, JavaScript Library* [online]. © 2012 [cit. 2012-04-29]. Dostupné z: http://jquery.com

[12] *Gettext - GNU Project - Free Software Foundation (FSF)* [online]. © 1998, 2010 [cit. 2012-04-29]. Dostupné z: http://www.gnu.org/software/gettext

[13] ARVIN, Troels. Comparison of different SQL implementations. *Troels Arvin's home page* [online]. 8. 3. 2012 [cit. 2012-05-09]. Dostupné z: http://troels.arvin.dk/db/rdbms