Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Jednotná administrace IT zdrojů KIV

Plzeň 2015 Martin Kučera

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 24. června 2015

Martin Kučera

Abstract

This thesis deals with possibilities of unified management of IT resources for Department of Computer Science and Engineering at University of West Bohemia. It describes managed resources by the department and possible automatization of frequently performed tasks. It compares existing solutions and analyses their suitability. Next part of this thesis proposes own solution and describes it's implementation.

Abstrakt

Tato práce se zabývá možnostmi správy IT zdrojů Katedry informatiky a výpočetní techniky (KIV) Západočeské univerzity v Plzni. Popisuje spravované zdroje katedrou a možnosti automatizace často vykonávaných úkonů. Srovnává existující řešení a analyzuje jejich vhodnost použití. V další části je navrženo vlastní řešení a popis jeho následné implementace.

Obsah

1 Úvod

Na katedře informatiky a výpočetní techniky (KIV) se často setkáváme s požadavky z oblasti IT zdrojů. Většina těchto požadavků je vyřízena manuálně odpovědnou osobou, z čehož plynou určité nedostatky. Jelikož neexistuje jednotná evidence prováděných či plánovaných operací, může docházet k duplicitám, kdy se dva či více lidí snaží splnit stejný úkol. Takto provozovaný systém se následně stává těžce udržovatelným a spravovatelným.

Dále k těžké udržovatelnosti systému přispívá množství a rozmanitost zdrojů katedry. Z důvodu velikosti katedry je vhodné, aby bylo možné vykonávat stejný typ úkolu opakovaně stejným postupem. Bez uceleného systému vykonávání požadavků a následné evidence mohou být některé úkoly provedeny více způsoby a tím vznikají rozdíly jednotlivých provedení.

Uvedené nedostatky je možné, alespoň částečně, eliminovat vhodným systémem pro evidenci a automatizaci vybraných úkolů. Automatizace požadavků zajistí jednotné provedení všech požadavků stejného typu, bez ohledu na to, kdo tento požadavek zadal. Díky následné evidenci všech požadavků jsme schopni zabránit případným duplicitám.

Cílem této práce je nastudovat možnosti a potřeby pro jednotou administraci IT zdrojů v rámci KIV a následné navržení vhodného řešení, které usnadní a sjednotí správu s důrazem na bezpečnost a efektivitu vybraného řešení.

2 Administrace zdrojů

IT zdroje představují fyzické nebo virtuální komponenty počítačového systému či infrastruktury. Administrace zdrojů představuje kritickou činnost pro zajištění bezproblémového, bezpečného a dlouhodobě udržitelného systému.

2.1 Možnosti administrace

Jedno z možných rozdělení řešení pro administraci je na ruční a automatizovanou správu.

2.1.1 Ruční správa

Při ruční správě vykonává pověřená osoba jednotlivé úkoly manuálně s žádnou či malou nápomocí podpůrného softwaru. Setkáme se s příkazovou řádkou, vestavěnými systémovými nástroji, ruční úpravou konfiguračních souborů pomocí textového editoru nebo s jednoúčelovými skripty. Úkoly se často opakují a tak je snadné dopustit se chyb či nepřeností.

Evidence požadavků a jejich stavů většinou neexistuje, což vede k problémům u vícečlených týmů při rozdělování a vykonávání jednotlivých úkolů.

Metoda je vhodná pro menší organizace, kde se o administraci stará jedna až dvě osoby. Výhodou tohoto přístupu je schopnost člověka reagovat na případné odchylky při vykonávání úkolů či pozorování neočekávaného chování systému.

2.1.2 Automatizovaná správa

Jedná se o typ správy infrastruktury, kdy je instalace a konfigurace jednotlivých služeb automatizována. V této oblasti převážně vyniká tzv. Configuration management, což je jedna z možností řízení systému. Configuration management umožňuje definovat stav jednotlivých služeb a následně tento

stav automaticky vynutí, díky čemuž eliminuje nutnost ruční instalace a konfigurace.

Configuration management často využívá centrálního serveru, nebo-li masteru a agentů, tedy klientských serverů. Stavy jednotlivých zdrojů jsou většinou centrálně uloženy pomocí konfiguračního souborů nebo databáze, díky čemuž jsme v případě selhání hardware schopni stav obnovit.

Configuration management je vhodný pro velké množství zdrojů, u kterých jsme schopni přesně definovat jejich konfiguraci. Pokud je mnoho zdrojů unikátních nebo se jedná o jednorázovou konfiguraci, definování konfiguračních souborů může zabrat stejné množství času jako ruční konfigurace. Některá řešení navíc nabízejí automatickou aktualizaci instalovaného softwaru, což ovšem může způsobit problémy v případě aktualizace softwaru, u kterého je nová verze nekompatibilní s verzí původní.

Mezi zástupce řešení typu configuration management patří například `Puppet` [1], `Ansible` [2], `Saltstack` [3] a `System Center 2012 R2 Configuration Manager` [4]. Dále budou popsáni dva populární zástupci Configuration Management a to `Puppet` a `Ansible`.

Puppet

Puppet je open-source[1] program sloužící k hromadné konfiguraci Unixových a Microsoft Windows systémů a jejich služeb. Administrátor definuje stav služeb pomocí tzv. manifestů, které představují konfigurační soubory agentů tvořené v jazyce Ruby DSL(Domain Specific Language)[2] nebo ve vlastním programovacím jazyce softwaru Puppet. Puppet umožňuje distribuci manifestů dvěma způsoby a to s použitím master serveru a bez použití master serveru.

Při použití master serveru je na tomto serveru spuštěn Puppet démon[3]. Na agentech běží Puppet agent, který se hlásí v určitém intervalu, klasicky 30 minut, master serveru a požaduje od něj svojí konfiguraci. Agent vyhledá rozdíly nalezené při porovnání s předchozí konfigurací a případné změny aplikuje.

[1] Počítačový program s veřejně přístupným zdrojovým kódem
[2] Programovací jazyk specifický pro konkrétní aplikační doménu
[3] Program, který je spuštěn na pozadí a běží výhradně bez interakce s uživatelem

V případě použití bez master serveru jsou manifesty ručně přenášeny na agenty a pomocí Puppet agent démonu aplikovány. Vhodné řešení při velkém počtu agentů z důvodu lepší škálovatelnosti nebo pro agenty, u kterých nelze proces automatizovat.

Existuje široké množství modulů, které dále rozšiřují základní možnosti a zjednoduší nám konfiguraci konkrétního softwaru. Moduly jsou převážně vyvíjeny a vystaveny ke stažení uživateli Puppetu.

Hlavní výhodou softwaru Puppet je poskytnutí abstraktní vrstvy, která nám umožní správu různých operačních systémů. Díky abstraktní vrstvě jsme schopni psát univerzální manifesty, které si za nás poradí například s odlišnými jmény softwaru či různými cestami k důležitým souborům.

Ansible

Ansible je další obdobou nástroje umožňujícího hromadnou konfiguraci. Ke konfiguraci agentů využívá SSH[4] a PowerShell[5]. K popisu konfigurace využívá formátu YAML[6]. Ke konfiguraci využívá jeden či více kontrolních serverů, které pomocí SSH nebo PowerShellu ovládají cílové klienty. Seznam klientů má kontrolní server lokálně uložený na rozdíl od Puppetu, kde master o agentech nemusí vědět.

Dostupná rozšíření poskytnou dodatečnou funkcionalitu, která dále zjednodušší správu konkrétních zdrojů. Rozšiřující moduly umožní ovládání systémových zdrojů jako jsou služby, softwarové balíky nebo soubory.

Výhodou Ansible je řešení bez nutnosti klientů cyklicky se dotazovat kontrolního serveru na aktuální nakonfigurovaný stav. Jelikož si kontrolní servery u sebe uchovávají seznam klientů a jejich síťovou adresu, dokáží klienty sami při změně kontaktovat.

[4] Zašifrovaný síťový protokol pro bezpečnou komunikaci
[5] Interpret příkazů specifický operačním systémům Microsoft Windows
[6] Koncept umožňující serializaci dat ve formě dobře čitelné člověku

2.1.3 Shrnutí

Výběr typu administrace záleží na velikosti administrované infrastruktury, počtu administrátorů a požadovaných vlastnostech serverů. Ruční administrace nabídne větší kontrolu v případě menšího systému a menšího počtu administrátorů. V případě snadného popisu požadovaných stavů nebo větších podobností spravovaných serverů může mít i u menšího systému smysl nasazení programu pro automatickou správu, například zmíněný typ Configuration management, který díky uloženým konfiguračním souborům umožní snadné nastavení a případné drobné úpravy jednotlivých klientů.

Velkou výhodou většiny řešení pro automatizovanou administraci typu Configuration management je existence dodatečných modulů, které umožní konfiguraci konkrétního software a tím pádem podstatně zjednoduší výsledný zápis konfiguračního souboru.

3 Analýza současného prostředí

Na katedře KIV se ke správě zdrojů využívá převážně ruční správy, tedy existují oprávněné osoby, které úkoly provádějí ručně bez podpůrného softwaru. Evidence požadavků neexistuje, a tak se spoléhá na pouhou domluvu mezi administrátory. Kvůli vlivům nedostatků ruční správy je třeba předcházet vzniku duplicit či nedorozumnění mezi jednotlivými administrátory při rozdělování a vykonávání úkolů. Velikost celkového systému a uvedené nedostatky vybraného typu správy stěžují jeho administraci.

3.1 Spravované zdroje

Mezi často spravované služby patří webové stránky, databáze, virtuální stroje, doménová jména, mailové schránky a mailové konference, soubory AFS systém, seznamy softwarových vybavení laboratoří a verzovací systémy SVN a GIT.

3.1.1 WWW

WWW patří mezi hojně využívané služby a tak i jejich správa patří mezi často vykonávané činnosti. Mezi základní operace patří vytváření, rušení či modifikace virtuálních hostů, které umožňují překlad jednoho či více doménových jmen na jednu IP adresu, čímž umožňují provozování více virtuálních hostů na jednom serveru. Jako webové servery jsou používány Apache HTTP Server[5], Internet Information Services[6], Apache Tomcat[7] a Jetty[8].

3.1.2 DNS záznamy

Samotnou správu DNS záznamů univerzity zajišťuje organizace CIV [1]. Požadavky jsou zadávány a náschledně schvalovány, zároveň je vhodné tyto požadavky evidovat.

[1] Centrum informatizace a výpočetní techniky Západočeské univerzity v Plzni

3.1.3 Databáze

Na katedře jsou využívány databáze MySQL[9], PostgreSQL[10], Oracle[11] a Microsoft SQL Server[12]. Častými požadavky jsou vytvoření nové databáze a založení uživatelského účtu s přístupem do nové databáze pomocí hesla. Pro každý typ databáze je jiný postup pro vytvoření nové databáze a druh přístupu závisí na použitém operačním systému. Správa databáze je umožněna z konzole cílového serveru či pomocí API databázového serveru.

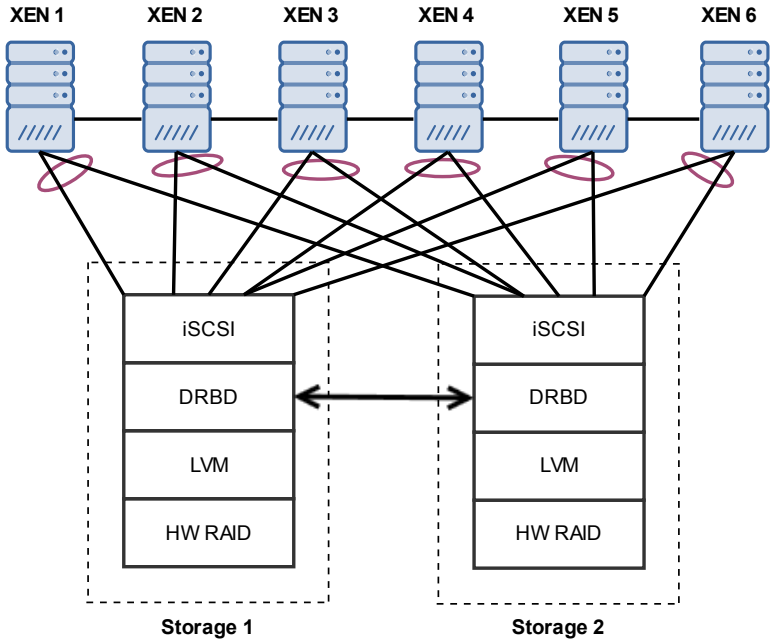
3.1.4 Virtualizace

Virtualizace strojů je zajišťována pomocí Xen[13] hypervizoru, což je rozhraní umožňující virtualizaci hardwaru a díky tomu možnost běhu více operačních systémů na jednom fyzickém zařízení. Mezi výhody virtualizace patří lepší využití hardwaru, možnost migrace virtuálních strojů na jiný hardware, bezpečnostní výhody díky oddělení jednotlivých operačních systémů a lepší možnosti správy.

Před založením virtuálního serveru je třeba zajistit vytvoření úložiště. Nejprve je vytvořen logický diskový oddíl pomocí LVM[14] na obou úložištích. Následně se zajistí synchronizace LVM oddílů pomocí DRBD[15]. Po dokončení synchronizace úložišť se na obou úložištích vytvoří iSCSI[16] target, nebo-li cílové zařízení. Nad iSCSI targety je ještě použit DM-Multipath[17], který slouží pro přístup k úložišti. DM-Multipath nabízí větší propustnost a v případě výpadku některého z úložišť redundanci, díky využití jiné cesty pro přístup k datům. Schéma zapojení viz obr. 3.1.

Po vytvoření úložiště jsou na všechny Xen Hypervizory nahrány konfigurační soubory nově vytvářeného virtuálního serveru. Úložiště serveru bude přístupné přes DM-Multipath. Posledním krokem je spuštění virtuálního serveru na vybraném Xen Hypervizoru a následné nastavení automatického spuštění virtuálu po startu hypervizoru.

U existujících virtuálních strojů lze provést migraci, neboli převést virtuální stroj z jednoho hypervizoru na jiný. Xen nabízí dva typy migrace a to offline a online. U offline migrace je běh virtuálu pozastaven, přesunuta potřebná data na cílový hypervizor a poté obnoven stav virtuálního stroje. Při online migraci je stroj ponechán spuštěný na originálním hypervizoru zatímco se migrují potřebná data, po dokončení migrace je virtuální stroj na



Obrázek 3.1: Schéma zapojení úložišť a XEN hypervizorů

původním hypervizoru pozastaven a spuštěn na novém.

3.1.5 Mailové konference

Mailové konference slouží pro odesílání emailových zpráv přihlášeným uživatelům do konference. Jsou zřizovány pomocí programu Mailman[18], který poskytuje pro každou konferenci několik emailových adres, sloužících například k odesílání emailových zpráv autorizovaným uživatelem a přihlášení nebo odhlášení z konference. Pro vytvoření nové konference je nutná adresa nové konference, adresa administrátorského emailu a heslo pro správu konference. Správa programu mailman je dostupná pomocí konzole, na kterou se můžeme připojit například SSH protokolem, nebo pomocí webové stránky.

3.1.6 Emailové účty

Správa emailových účtů a jejich aliasů. Emailové účty jsou provázány se systémovými uživateli a autentizace je zajištěna protokolem Kerberos. Lze zároveň vytvářet aliasy a virtuální aliasy. Aliasy slouží ke směrování více emailových adres do emailové schránky systémového uživatele nebo i na cizí emailovou adresu v případě virtuálního aliasu. Účty jsou přístupné pomocí protokolů POP[2] a IMAP[3] zabezpečených pomocí SSL[4].

3.1.7 OpenAFS svazky

OpenAFS[19] je implementací AFS, což je distribuovaný souborový systém, využíván univerzitou mimo jiné pro úložný prostor studentů. Systém je rozdělen na svazky, což jsou základní jednotky, které jsou dále exportovány uživatelům. Pro vytvoření svazku je třeba určit cílovou cestu, kvótu dat a přístupová práva pro přístup.

3.1.8 Seznam softwaru laboratoří

Evidence softwaru jednotlivých laboratří a to pro OS Windows i Linux. Seznam platí na jeden semestr a je vyplněn vyučujícím před začátkem semestru a to pro operační systémy Linux a Windows. Díky seznamu je lepší přehled nainstalovaných programů, který umožňuje například upřednostněnou aktualizaci v případě výskytu bezpečnostní chyby.

3.1.9 SVN a GIT repozitáře

Pro verzovací systémy, které jsou využívány pro projekty studentů a zaměstnanců univerzity, jsou použity programy SVN a GIT. Při vytváření nového repozitáře potřebujeme znát název nového repozitáře, cílové umístění a případné uživatele, kteří mají mít do nového repozitáře přístup.

[2] Post Office Protocol slouží pro stahování emailových zpráv ze serveru na klienta
[3] Internet Message Access Protocol slouží pro přístup k emailové schránce
[4] Secure Sockets Layer je protokol zabezpečující komunikaci

Přístup do SVN repozitáře je zajištěn pomocí HTTPS protokolu[5] a modulu do Apache web serveru nazvaného mod_dav_svn, který dovolí web serveru Apache zobrazit obsah SVN repozitáře. Autentizace je zajištěna pomocí protokolu Kerberos.

U GIT repozitáře je přístup spravován pomocí programu Gitolite, který umožňuje přístup do repozitářů autorizovaných uživatelů pomocí SSH protokolů a SSH klíče.

3.2 Požadované vlastnosti

Ze současného prostředí a spravovaných zdrojů vyplívají následující požadavky na systém pro jednotnou správu zdrojů.

3.2.1 Ověření uživatelů

Aplikace umožní ověřování uživatelů pomocí protokolu Kerberos, který slouží jako centrální autentizační služba univerzity a LDAP [6](Lightweight Directory Access Protocol) skupin, které poskytnou třídění uživatelů podle předmětů a kateder.

3.2.2 Zadávání žádostí

Veškeré úkoly jsou zadávány uživateli jako žádosti a podle jejich práv bude případně nutné schválení autorizovanou osobou, kterou zastupuje například vedoucí laboratoře, předmětu nebo katedry.

3.2.3 Hierarchie práv

V systému existuje hierarchie práv, kde každý vidí své možnosti a možnosti nižších úrovní. Existují práva pro přístup k jednotlivým prvkům systému,

[5] Zabezpečená verze HTTP protokolu
[6] Protokol sloužící pro práci s daty uložených na adresářovém serveru

možnost zadat a zobrazit existující požadavky, možnost schválení požadavků a možnost změnit oprávnění uživatele.

3.2.4 Skupiny

Možnost vytvářet skupiny uživatelů. Každý uživatel může být přiřazen do jedné nebo více skupin.

3.2.5 Administrátorská práva

Administrátor může provádět veškeré operace bez nutnosti jejich schvalování a zároveň může přiřadit administrátorská práva ostatním uživatelům.

3.2.6 Logování žádostí

Všechny žádosti jsou důsledně zaznamenávány. Zároveň se žádostmi jsou zaznamenáváni jejich zadavatelé, případně uživatel, který žádost schválil a stav žádosti.

3.2.7 Fronta žádostí

Přehled aktuálních žádostí ve frontě a jejich stav.

3.2.8 Notifikace

Možnost zapnout notifikace emailem pro informování o stavu úkolu.

3.2.9 Přepnutí účtu

Administrátor si může dočasně přepnout svůj účet na jiného uživatele, což administrátorovi umožní přístup k systému jako jiný uživatel bez nutnosti zadání uživatelova hesla.

3.3 Existující nástroje

Byly ověřeny dostupné programy, které by mohly nejlépe splňovat požadavky na správu zdrojů. Dříve uvedené nástroje typu Configuration Management nebyly dále uvedeny, jelikož jejich architektura obecně nesplňuje požadované vlastnosti řešení. Zde je uveden přehled nejvhodnějších kandidátů.

3.3.1 Webmin

Webmin [20] je nástroj pro správu Unixových a Windowsových systémů a populárních serverových aplikací, mezi které patří například Apache HTTP Server, PHP[7] nebo MySQL. Program je vyvíjen v jazyce Perl a založen na webovém uživatelském rozhraní, které umožní dostupnost na širokém množství zařízení a operačních systému.

Hlavní výhodou programu je jeho modularita, díky které je možné pomocí dodatečných modulů rozšířit funkčnost nad standardní možnosti. Webové rozhraní poskytuje mimo jiné i obecné informace o systému, jako je aktuální jádro systému, architektura procesoru, jak dlouho systém běží, počet procesů a další. Mezi dva větší a populárnější moduly patří **Cloudmin** a **Virtualmin**.

Cloudmin modul

Cloudmin je modul Webminu, který poskytuje rozhraní pro správů virtuálních serverů na bázi Linux XEN, KVM[8], OpenVZ[9] a další. Pomocí Cloudminu je možné vytvářet, mazat a dále spravovat více instancí virtuálních serverů založených na libovolných podporovaných technologií. K ovládání je mimo webového prostředí dostupné ještě ovládání pomocí API[10] a příkazové řádky.

[7] PHP: Hypertext Preprocessor, skriptovací programovací jazyk
[8] Virtualizační technologie umožňující použít linuxové jádro jako hypervizor
[9] Virtualizační technologie umožňující běh několika samostatných instancí, nebo-li kontejnerů, na jednom fyzickém zařízení
[10] Programátorské rozhraní, pomocí kterého lze využívat procedury, funkce či protokoly programu

Virtualmin modul

Virtualmin je rozšiřující modul, pomocí kterého jsme schopni spravovat libovolné množství virtuálních domén, mailových schránek, databází a dalších serverových aplikací. Stejně jako Cloudmin, lze Virtualmin ovládat pomocí webového rozhraní, příkazové řádky či API.

I přes to, že je Webmin primárně navržen na OS typu Unix/Linux, s omezenou funkčností zvládne správu i OS Windows. Mezi zápory by se dalo zařadit nemožnost doinstalovat spravované aplikace přímo z rozhraní Webminu.

3.3.2 cPanel

cPanel [21] je linuxový nástroj pro usnadnění správy webových stránek a základních systémových služeb pomocí webového rozhraní. Po instalaci je zároveň nainstalována součást cPanelu WHM (Web Host Manager), který slouží pro administrátory systému. cPanel je rozdělen na tři základní části určené pro administrátory, přeprodejce [11] a koncové uživatele. Oddělení jednotlivých částí systému podle účelu nabídne větší přehlednost a bezpečnost.

Mimo ovládání pomocí webového prohlížeče nabízí ještě příkazovou řádku a API. Mezi spravovatelné aplikace patří Apache HTTP Server, PHP, MySQL, PostgreSQL, Perl a BIND. Pro emaily nabídne podporu POP3, IMAP a SMTP. Instalaci cPanelu je doporučováno provést na nově instalovaný systém bez modifikací. Stejně tak při odinstalování je doporučeno raději přeinstalovat celý systém, než se snažit cPanel odstranit.

cPanel podporuje operační systémy CentOS, Red Hat Enterprise Linux, CloudLinux a v dřívějších verzích i FreeBSD. Pro OS Windows vznikla verze se jménem Enkompass, která ovšem nabízí pouze omezené možnosti pro správu a již není aktivně vyvíjena ani podporována.

Mezi nedostatky se může řadit minimální možnost konfigurace či správy dodatečných aplikací. Další omezení představuje nutnost instalace na čistě nainstalovaný systém a značně komplikovaný postup v případě deinstalace.

[11] Osoba, která je oprávněna dále prodávat přidělené zdroje

3.3.3 Openstack

Openstack [22] je program pro správu cloudu. Obvykle je používán jako IaaS (Infrastructure as a service), což je distribuční model cloud computingu, u kterého si zákazníci pronajímají škálovatelnou infrastrukturu. Díky možnosti vytvářet nové instance virtuálních strojů poskytuje snadnou škálovatelnost systému do šířky. Do vývoje je zapojeno mnoho velkých firem, mezi kterými jsou například Intel, Dell, Cisco a Oracle. Ovládání je umožněno přes webové rozhraní, příkazovou řádku či API. Velkou výhodou pro administrátory a vývojáře je detailní dokumentace celého programu, která umožní rychlou orientaci novým uživatelům.

Program je navržen modulárně, čímž umožňuje modifikaci nebo úplnou náhradu některého z podsystémů. Mezi dostupné komponenty patří například **Nova**, **Swift** a **Horizon**.

Nova je hlavní komponenta pro správu počítačových zdrojů. Dokáže pracovat přímo s hardwarem počítače a několika různými druhy hypervizorů, mezi které patří například KVM, VMWare a Xen.

Swift poskytuje škálovatelný redundantní úložný systém. Soubory jsou zapisovány na více disků mezi jednotlivými servery a OpenStack zajišťuje replikaci a integritu dat. Při poruše disku se sám postará o migraci dat z jiných aktivních disků na nový disk.

Horizon umožňuje uživatelům ovládání pomocí webového rozhraní. Mimo jiné poskytuje i monitoring celého systému.

3.3.4 ApacheGUI

ApacheGUI [23] je utilita[12] pro zjednodušení práce s webovým serverem Apache. Program je napsán v programovacím jazyce Java a pomocí webového rozhraní umožňuje správu Apache serveru na OS Windows, Linux či OS X. Nabízí přehled o běžícím Apache procesu, aktuální vytížení procesoru, detailní statistiky přístupu uživatelů a přenosu dat a možnost tento

[12] Pomocný program

proces restartovat či zastavit. Velkou výhodou je možnost editace zdrojových souborů webových stránek přímo z webového rozhraní, čímž lze ušetřit čas v případě nutnosti rychlé editace.

Samotné nastavení serveru lze jednoduše editovat pomocí vestavěného editoru a před nasazením nové konfigurace provést nejprve test, zda je syntax v pořádku. K monitorování běhu serveru je umožněn přístup k logovacím souborů či grafy, zobrazující například počet přístupů na konkrétní soubor. Funkcionalita pro ovládání více serverů s instalací ApacheGUI z jednoho uživatelského rozhraní chybí, což není příliš vhodné při velkém množství spravovaných serverů.

3.3.5 SVN Access manager a ViewVC

SVN Access manager [24] je program zaměřený na správu přístupu k SVN repozitářům. Pro správu přístupu nabízí uživatelské účty a skupiny, do kterých je možné uživatele sdružovat. Pro jednotlivé uživatele a skupiny lze nakonfigurovat přístup čtení a zápisu a také přístup do konkrétních složek v repozitáři. Umožňuje také hromadný import uživatelů pomocí LDAP protokolu a integraci s programem ViewVC.

ViewVC je rozhraní určené do webového prohlížeče pro CVS a SVN repozitáře. Generuje náhled na repozitáře, revize a jednotlivé changelogy[13]. Dále je možné zobrazit soubory v libovolné verzi a zobrazit rozdíl jednotlivých verzí souboru. Poskytuje podobné funkce jaké jsou dostupné z příkazového řádku, ale v přehlednější a uživatelsky přívětivější formě.

3.3.6 phpMyAdmin

phpMyAdmin [25] je nástroj pro správu MySQL databáze, napsán ve skriptovacím jazyce PHP a používající webové rozhraní. Ve svém webovém rozhraní nabídne možnost provádět SQL dotazy nad dostupnými databázemi, prohlížení struktur jednotlivých tabulek, import a export databáze a volbu jazyka z mnoha dostupných překladů. Pro urychlení práce nabídne přehled tabulek spolu s odhadovaným počtem záznamů, velikostí a typem tabulky.

[13] Seznam změn mezi verzemi repozitáře

Poskytuje také rychlé vyhledávání záznamů v tabulce pomocí předpřipravených SQL dotazů či vyprázdnění tabulky jedním kliknutím. Výhodou je také dostupnost ve více než 50 ruzných jazicích a multiplatfornost. Za nedostatek by se dala označit nepřítomnost logovacích záznamů databáze přímo z webového rozhraní.

3.3.7 pgAdmin

pgAdmin [26] je multiplatforní program umožňující správu PostgreSQL databáze. Ke správě databáze nevyužívá žádné abstraktní vrstvy, ale přistupuje k ní přímo pomocí PostgreSQL API. Nabízí zvýrazňování syntaxe SQL dotazů, rychlý přístup k velkému množství dat, prohlížení logovacích souborů, zobrazení stavu databáze a jejich procesů a plánování SQL dotazů, které budou ve specifikovaný čas automaticky šputěny. Díky multiplatformnosti umožňuje správu databáze ze všech dnes hojně používaných operačních systémů a pro uživatele nabídne přes 10 kompletních překladů a přibližně 30 částečně přeložených.

3.3.8 Oracle SQL Developer

Oracle SQL Developer [27] je vývojové prostředí pro usnadnění správy a práce s Oracle databází. Program umožňuje spuštění na operačních systémech s podporou jazyku Java. Pro vývojáře nabízí editory pro práci s SQL, PL/SQL, uloženými Java procedurami a XML. Pro administrátory poskytuje export databáze, audit databáze, správu uživatelů a rolí, správu úložného prostoru, diagnostiku databáze a práci s RMAN, což je manažer pro obnovu databáze.

3.3.9 EMS SQL Manager

EMS SQL Manager [28] je určen pro správu a práci s MS SQL Serverem. Aktuálně (květen 2015) podporuje verze od 2000 do 2014, spolu s nejnovějšími funkcemi. Nabízí možnost urychleného sestavování dotazů, průchodu databázemi a správy SQL Server objektů. Poskytuje nástroje pro práci s daty, import a export, návrh databází pouhými kliknutím myší a ladění procedur, funkcí a SQL skriptů.

3.3.10 AFS Manager

AFS Manager je program, umožňující správu OpenAFS buňek na operačních systémech, které podporují programovací jazyk Java. Nabízí možnost správy uživatelů, skupin, přístupových práv, svazků, oddílů a záloh.

3.3.11 Postfix Admin

Postfix Admin [29] je program s webovým rozhraním, zaměřený na správu MTA[14] Postfix. Pro ukládání uživatelů je použita MySQL nebo PostgreSQL databáze. Umožňuje vytvářet nové uživatele, aliasy a postfixové virtuální domény. Zároveň je možné nastavit automatické odpovědi na maily a integraci webového mailového klienta Squirrelmail či Roundcube.

3.3.12 XenCenter

XenCenter [30] je program pro správu virtualizační platformy Citrix XenServer dostupný pro OS Windows. Uživateli umožní pomocí grafického rozhraní vytvářet nové virtuální stroje, konfiguraci virtuálních strojů a přístup do konzole běžícího virtuálního stroje. Pro lepší přehled umožňuje řazení virtuálních strojů do skupin a snadné vyhledávání podle parametrů virtuálu.

Mezi hlavní funkce se řadí možnost dynamické alokace paměti ze zadaného rozmezí, integrace Active Directory, vytváření snapshotů[15] virtuálních strojů, vytváření snapshotu paměti virtuálu a možnost vrátit stav virtuálu do předchozího. Pro monitoring nabídne výkonové grafy jednotlivých strojů, které zahrnují například využití paměti, procesorů a síťových karet. Dále nabízí možnost nastavení obnovy virtuálu v případě poruchy hardwarové poruchy hypervizoru, na kterém virtuál běží.

[14] Mail Transfer Agent, program pro přenos mailů z jednoho počítače na druhý.
[15] Uložení aktuálního stavu virtuálního stroje

3.3.13 Přehled

Přestože některé aplikace nabízejí možnosti správy konkrétního zdroje, většinou nepodporují vzdálenou správu a tím pádem jsou schopny spravovat pouze lokální server, na kterém jsou spuštěny. Pouze lokální správa bohužel často nevyhovuje zadání.

Tabulka 3.1: Přehled podpory prvků zadání jednotlivými aplikacemi

	WWW	DNS	DB	Virt.	M. konf.	Mail	AFS	Sezn. SW	Repo.
Webmin	Ano ✓	Ne ✢	Ne ✢	Ne ✗	Ano ✓	Ne ✢	Ne ✗	Ne ✗	Ano ✓
cPanel	Ne ✢	Ne ✢	Ne ✢	Ne ✗	Ne ✗	Ne ✢	Ne ✗	Ne ✗	Ne ✗
Openstack	Ne ✗	Ne ✗	Ne ✗	Ne ✢	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗
ApacheGUI	Ne ✢	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗
SVN AM, VVC	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✢
phpMyAdmin	Ne ✗	Ne ✗	Ne ✢	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗
pgAdmin	Ne ✗	Ne ✗	Ne ✢	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗
Ora. SQL Dev.	Ne ✗	Ne ✗	Ne ✢	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗
EMS SQL Man.	Ne ✗	Ne ✗	Ne ✢	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗
AFS Man.	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ano ✓	Ne ✗	Ne ✗
Postfix Adm.	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✢	Ne ✗	Ne ✗	Ne ✗
XenCenter	Ne ✗	Ne ✗	Ne ✗	Ne ✢	Ne ✗	Ne ✗	Ne ✗	Ne ✗	Ne ✗

✓ splňuje všechny požadavky zadání pro zdroj
✢ splňuje zadání pouze částečně
✗ nesplňuje zadání vůbec

3.3.14 Souhrn

Byly otestovány aplikace, které se jevily jako nejvhodnější kandidáti na správu zdrojů katedry KIV. Z dosažených výsledků je vidět, že ani jedno řešení nepodporuje správu všech typů zdrojů podle požadavků.

Většina aplikací je zaměřena na jeden konkrétní účel a pokud nabídnou podporu jiných typů zdrojů, než k čemu jsou primárně určené, tak pouze s omezenou funkcionalitou. Dalším častým omezením byla možnost aplikace spravovat pouze lokální server, na kterém byly nainstalovány, což je v rozporu s cílem sjednotit správu do jednoho řešení.

Jelikož nebylo nalezeno vhodné existující řešení, bude navrženo vlastní, u jehož návrhu a vývoji se můžeme inspirovat vybranými prvky otestovaných aplikací, jako je například organizace ovládacích prvků v uživatelském rozhraní, přehled čekajících a aktuálně prováděných úkolů a případně i funkcionalitou některých řešení. Cílem je aplikace umožňující efektivní a pro uživatele nenáročnou správu požadovaných zdrojů, s přehledným, intuitivním uživatelským rozhraním a možností budoucího rozšíření o další funkce.

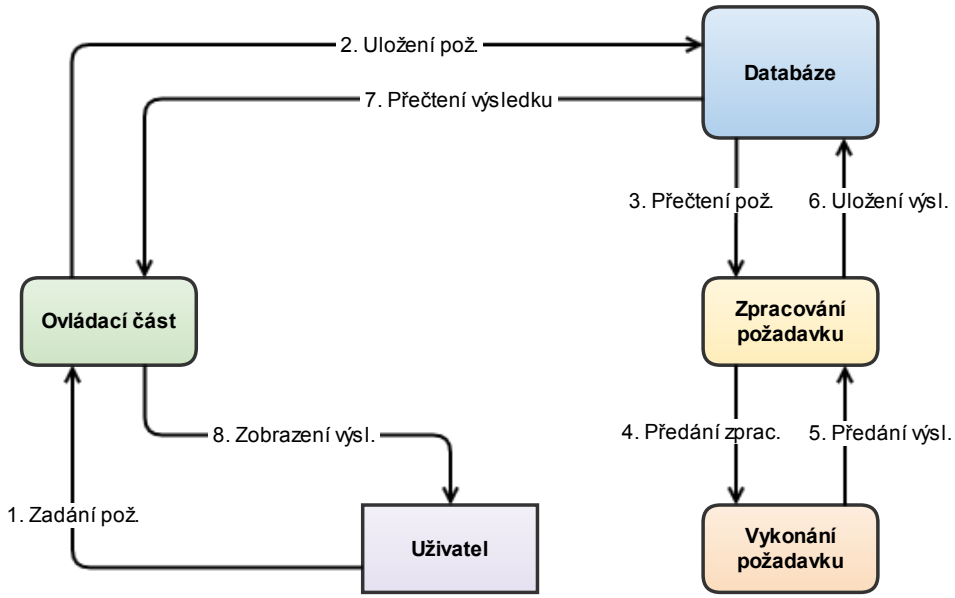
4 Návrh vlastního řešení

Při návrhu vlastního řešení budeme klást důraz na splnění všech požadovaných vlastností systému, eliminaci nedostatků stávajícího řešení administrace, zajištění bezpečnosti řešení a snadnosti ovládání. Dále bude třeba zvážit strukturu řešení a zvolit vhodné technologie.

4.1 Struktura a technologie

Výsledné řešení se bude skládat z několika částí a to z webové stránky, která bude sloužit pro ovládání programu, následně z části zpracovávající požadavky (démon) a části vyřizující zpracované požadavky. Části budou propojeny pomocí databáze, která bude uchovávat všechny zadané požadavky. Přibližná struktura programu je zobrazena na obr. 4.1.

Celý program bude muset být velmi dobře zabezpečen. Případné proniknutí útočníkem do některé z částí systému by mělo fatální následky. Pokud by se útočníkovi podařilo dostat do částí kde běží ovládací web nebo zpracovávající démon, dostal by přístup nejen k těmto serverům ale zároveň by mohl částečně ovládat i spravované klienty.



Obrázek 4.1: Schéma struktury navrhovaného řešení

4.1.1 Operační systém

Výběr technoligií pro jednotlivé části výsledného programu závisí na zvoleném operačním systému. Mezi hlavní kandidáty se řadí GNU/Linux a MS Windows, jelikož jsou použity na převážné většině serverů katedry KIV. Z těchto dvou kandidátů vychází navrch v tomto konkrétním případě GNU/-Linux, a to kvůli převaze v zastoupení cílových serverů přibližně 9:1 oproti systému MS Windows. Další výhodou je jednoduchost ovládání jednotlivých služeb pomocí konzole.

Konkrétní distribucí GNU/Linux bude Debian, který je jednou z nejpoužívanějších distribucí katedrou KIV. Mezi další výhody distribuce Debian patří stabilita systému, důkladné testování programů zařazených do oficiálních repozitářů[1] a jednoduchost instalace a konfigurace základních služeb systému.

[1] Hlavní zdroj programů, nebo-li balíčků

4.1.2 Databáze

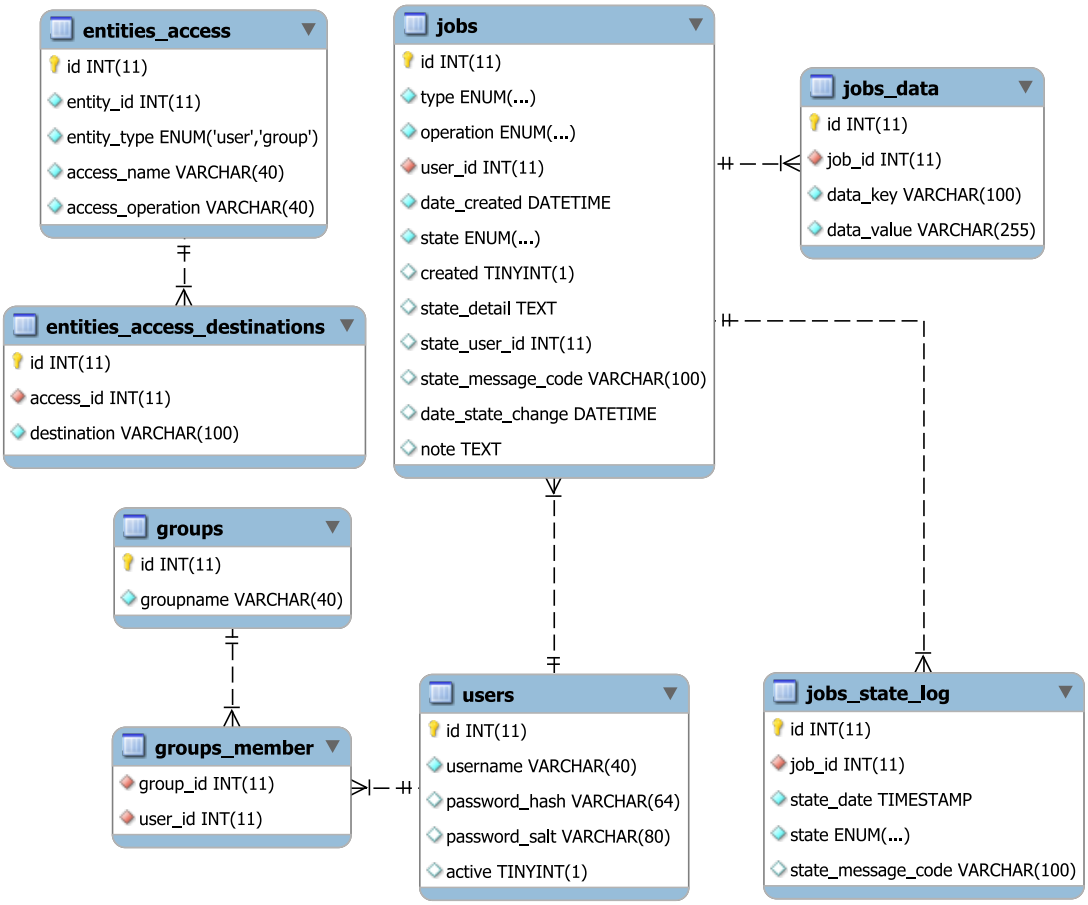
Databáze bude sloužit pro vkládání nových požadavků z webové stránky. Požadavky budou dále čteny démonem pro zpracování požadavků a následně bude výsledek vykonání zaznamenán do databáze. Jako databázi jsem zvolil databázi MySQL, která používá pro ovládání jazyku SQL[2]. Databáze MySQL se díky své popularitě dočká podpory v mnoha programovacích jazycích a operačních systémech, mezi které patří i vybraná distribuce Debian GNU/-Linux. Předpokládaná průměrná zátěž databáze bude v jednotkách dotazů za sekundu, pro kterou bude výkon databáze MySQL více než dostačující.

Struktura

Databáze bude složena z 8 tabulek, které budou hlavními zdroji dat pro fungování celého programu. Schéma databáze viz 4.2.

- `jobs` - bude uchovávat všechny požadavky a to hlavně jejich typ, uživatele, který požadavek založil, datum založení požadavku, aktuální stav a případného uživatele, který naposled měnil stav požadavku. Odkaz na uživatele bude řešen pomocí cizího klíče.
- `jobs_data` - specifická data pro každý typ požadavku spojena s daným požadavkem pomocí cizího klíče.
- `jobs_state_log` - historie stavů jednotlivých požadavků spolu s časem a případnou zkratkou stavové zprávy.
- `users` - tabulka bude seznam uživatelů a pokud se jedná o lokálního uživatele, bude zároveň uchovávat hash hesla a sůl použitou k zahashování. Dále zde bude sloupec označující zda je účet aktivní nebo ne (`TRUE`/`FALSE`). Místo smazání uživatele nastavíme tento sloupec na hodnotu `FALSE`, čímž uchováme v databázi uživatelovo údaje, abychom mohli rozlišit jím zadané požadavky.
- `groups` - seznam skupin. Skupina nebude mít žádné další sloupce kromě jejího názvu a identifikátoru.
- `groups_member` - přiřazení uživatele do existujících skupin pomocí cizího klíče skupiny a cizího klíče uživatele.

[2] Standardizovaný strukturovaný dotazovací jazyk



Obrázek 4.2: ER schéma databáze

- `entities_access` - bude sloužit k přiřazení oprávnění uživateli nebo skupině. Cílová entita bude jednoznačně určena pomocí typu a identifikátoru entity. Typ entity může následně být uživatel nebo skupina a samotné oprávnění bude označeno názvem oprávnění a povolenou akcí.
- `entities_access_destinations` - bude uchovávat cílové klienty, ke kterým se vztahuje odkazované oprávnění. Oprávnění může mít žádné nebo teoreticky neomezený počet cílových klientů. Na vztahované oprávnění se budeme odkazovat pomocí cizího klíče.

4.1.3 Webová stránka

Řešení v pohodě webové stránky nám umožní podporu širokého spektra zařízení a operačních systému, mezi které patří například MS Windows, GNU-/Linux, OS X, Android a iOS.

Pro vytvoření stránky bude použito PHP v kombinaci s HTML, CSS a Javascriptem. PHP je skriptovací programovací jazyk, který je aktuálně (květen 2015) [31] nejpoužívanějším skriptovacím jazykem pro vývoj webových stránek.

Díky svojí popularitě má širokou podporu webovými servery, čímž umožní snadné nasazení a zprovoznění výsledné stránky. Pro PHP zároveň existuje mnoho existujících frameworků[3], které můžou značně usnadnit a urychlit vývoj. Disponuje podporou vybrané distribuce Debian GNU/Linux a možností komunikace s databází MySQL.

Ovládací web bude třeba zabezpečit protokolem HTTPS, který zabrání odposlechnutí přihlašovacích hesel nebo kritických informací odesílaných oprávněným uživatelem. Samotné přihlašování uživatelů bude rozděleno na lokální a pomocí protokolu Kerberos. Lokální účty umožní přihlášení i v případě nedostupnosti Kerberos KDC (Key Distribution Center)[4]. U lokálních účtů bude třeba uložení hesla v databázi pro ověření správnosti při přihlašování. Hesla budou uložena v databázi v hashované podobě[5] a zároveň osolena[6], aby i v případě odcizení obsahu databáze útočník neměl přístup k heslům

[3] Knihovna sloužící pro ulehčení programování aplikace
[4] Důvěryhodná třetí strana skládající se z autorizačního a řídícího serveru
[5] Jednosměrný otisk vstupu
[6] Použití dodatečných dat pro vstup z důvodu ztížení získání původního hesla pomocí slovníkového útoku

uživatelů ve tvaru prostého textu.

4.1.4 Webový server

Jako webový server bude použit Apache. Podpora PHP je zajištěná jednoduchou instalací dodatečného modulu, stejně tak v případě modulu pro externí autentizaci uživatelů, díky kterému budeme moct ověřovat uživatele protokolem Kerberos. Díky modulu pro přepisování vyžádaných URL[7] jsme schopni směrovat všechny požadavky přes jeden soubor, což nám mimo jiné usnadní kontrolu oprávnění uživatele pro vyžádanou stránku. Apache je součástí repozitářů mnoha linuxových distribucí, včetně distribuce Debian.

4.1.5 Zpracovávání požadavků

Část zpracovávající požadavky poběží jako démon, tedy bez nutnosti uživatelské interakce. V nastavitelném časovém intervalu bude cyklicky kontrolovat nově zadané požadavky v databázi, zajistí zpracování potřebných dat a výsledek předá části vykonávající požadavky.

Jako programovací jazyk jsem si zvolil jazyk C. Disponuje podporou vláken, operačního systému Debian GNU/Linux a pomocí knihovny umožní komunikaci s databází MySQL. Zároveň nabídne dobrý výkon, schopnost komunikovat s operačním systémem a v případě nutnosti i multiplatformní podporu.

Démon už ze svého principu běží bez interakce uživatele, jediná komunikace s vnějškem je načtení konfiguračního souboru, přístup do databáze a zaznamenávání do logovacích souborů. Bude tedy třeba zajistit, aby spuštění démona mohla provést pouze oprávněná osoba. Pro konfigurační soubor a logovací soubory musíme zajistit správné nastavení přístupových práv, aby je nemohl běžný uživatel číst nebo měnit.

[7] Řetězec znaků pro přesnou identifikaci dokumentů

4.1.6 Vykonávání požadavků

Část vykonávající požadavky převezme zpracovaná data a požadavek vykoná. O svém výsledku bude informovat zpracovávajícího démona, kterému předá návratovou hodnotu a v případě neúspěchu hlášku s chybovým výpisem.

Vykonávání požadavků budou zprostředkovávat shellové skripty. Ve většině případů se bude jednat o volání konzolových příkazů a editaci konfiguračních souborů. Shellové skripty jsou podporovány nativně[8] distribucí Debian GNU/Linux a jsou ideální pro sekvenci příkazů.

Samotné skripty jsou schopny provést jen ty příkazy, na který má uživatel spouštějící skript systémová práva, jinak končí chybou. I přes to bude lepší, pokud k nim budou mít přístup pouze oprávněné osoby. Konfigurační soubory musejí být zabezpečeny přístupovými právy z důvodu možné přítomnosti kritických informací, kterými mohou být například přístupové údaje do databáze.

Hlavní ovládání klientů bude probíhat pomocí protokolu SSH. Tento protokol zajistí šifrovanou komunikaci, umožní omezení množiny dostupných příkazů na cílovém klientu a zároveň je součástí většiny instalací systému GNU/Linux, čímž zjednodušší konfiguraci klientů. Pro autentizaci na klientských serverech bude použito SSH klíčů[9], které umožní ověření pomocí veřejné části klíče. Pro privátní části SSH klíčů musíme zajistit bezpečné uložení.

U některých typů zdrojů bude potřeba na klienty přenést dodatečná data, kterými mohou být konfigurační soubory. Pro přenášení dat bude využito programu rsync, který je součástí repozitářů většiny linuxových distribucí. Jeho výhodou je možnost použití SSH protokolu pro přenos, což nám zajistí bezpečnost dat a možnost využití SSH klíču pro autorizaci. Zároveň jsme schopni u klientů omezit do kterých složek smí rsync s uvedeným SSH klíčem zapisovat.

4.2 Modularita

Jednotlivé podporované zdroje budou do programu přidány jako moduly, čímž se případnými změnami stávajících či přidáním podpory nového zdroje

[8] Podpora je součástí operačního systému
[9] Dvojice složená z privátního a veřejného klíče

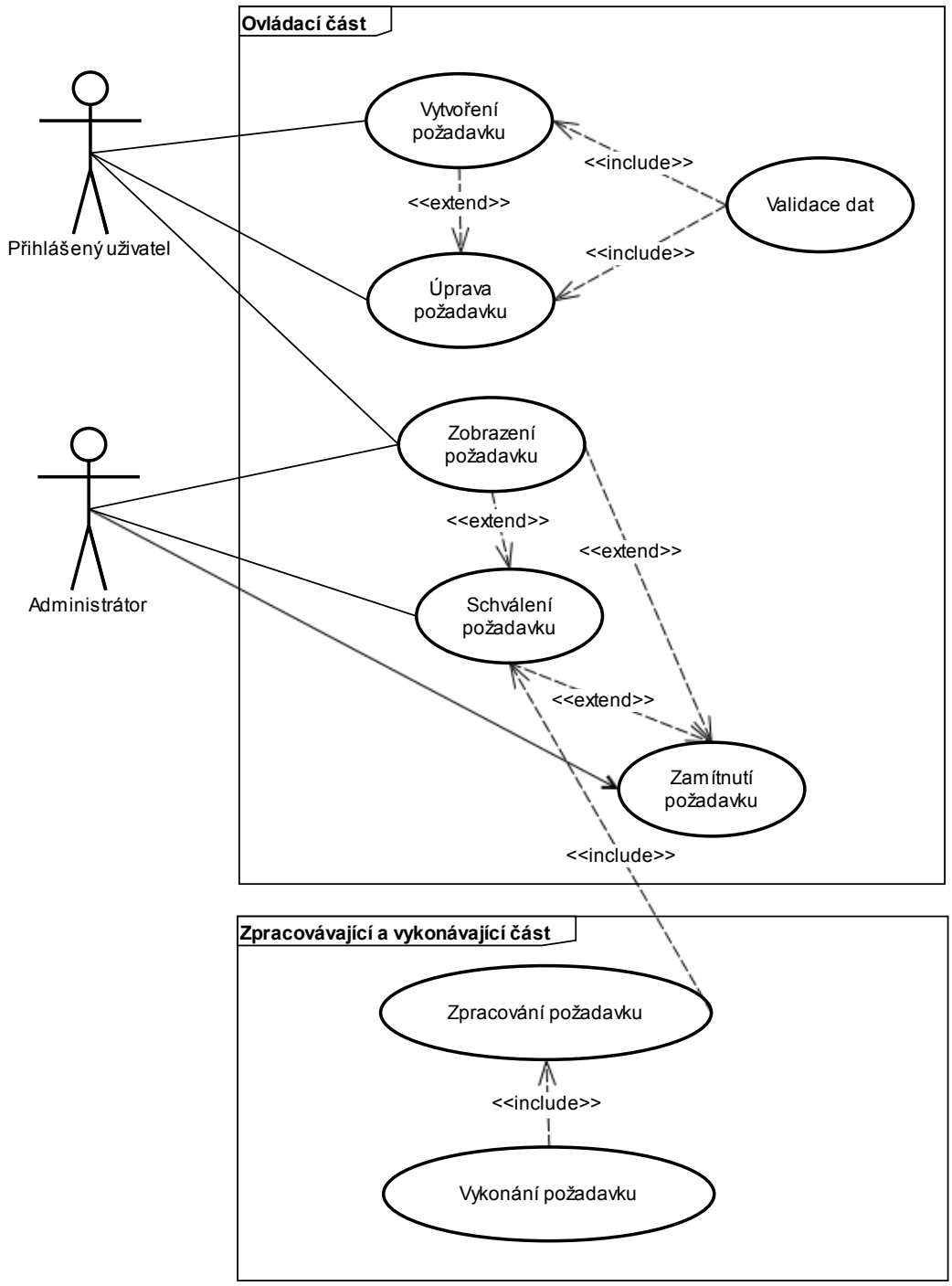
neovlivní funkcionalita ostatních. Zároveň zajistíme přehlednější návrh aplikace díky oddělení logiky pro všeobecnou funkčnost spojenou se spravováním zdrojů a logiky spojené s konkrétními zdroji.

4.3 Notifikace

Uživatel má možnost zapnout si notifikační emaily při změně stavu jím zadaných požadavků. Notifikace budou rozesílány při změně stavu z webové stránky, tedy například při schválení požadavku, a při vykonání požadavku vykonávací částí.

4.4 Použití

Základní diagram použití je vyobrazen na následujícím obrázku 4.3.



Obrázek 4.3: Diagram použití programu

5 Implementace

V následující kapitole je popsána implementace jednotlivých částí navrženého řešení.

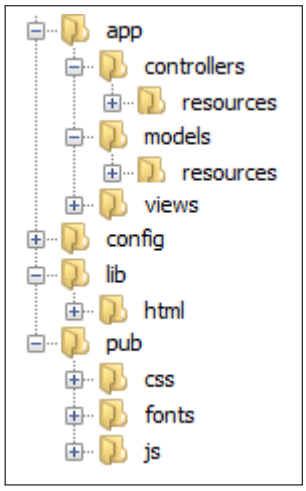
5.1 Webová stránka

Webová stránka je převážně zastoupena jazykem PHP a má architekturu modelu MVC(Model-view-controller), která rozděluje datový model na tři části a to model, view a controller. `Model` reprezentuje data, s kterými budeme pracovat. `View` transformuje data předaná modelem k zobrazení uživateli. `Controller` reaguje na události a může posílat příkazy `modelu` na aktualizaci dat nebo `view` na změnu typu zobrazení.

5.1.1 Adresářová struktura

Soubory jsou uspořádány do sdružujících adresářů pro lepší přehlednost. Struktura částečně vychází z architektury MVC, viz obr. 5.1.

- `app` - obsahuje všechny soubory typu model, controller a view. Controllery a modely jsou dále odděleny pro obecnou funkčnost a pro specifické jednotlivým typům zdrojů.
- `config` - v tomto adresáři jsou soubory pro konfiguraci webu, jako je například hlavní konfigurační soubor, který mimo jiné uchovává informace nutné pro připojení k databázi.
- `lib` - adresář s knihovnami. Obsahuje třídy jejichž instance reprezentují objekty jako jsou například uživatelé nebo jednotlivé požadavky. Dále jsou zde soubory s obecnými funkcemi, třída pro přístup do databáze a třídy použité pro vykreslování některých prvků uživatelského rozhraní.
- `pub` - obsahuje veřejně přístupné soubory, jako CSS, fonty či Javascripty. Dále obsahuje úvodní soubor `index.php`, na který je uživatel přesměrován při přístupu na jakoukoliv podstránku.



Obrázek 5.1: Adresářová struktura webu

5.1.2 Popis důležitých souborů

- `lib/bootstrap.php` - volán při každém načtení stránky. Načte konfigurační soubory, seznam přístupných stránek a podpůrné funkce. Následně získá z požadované adresy název volaného controlleru a ze seznamu přístupných stránek zjistí, zda je návštěvník oprávněn tuto stránku navštívit.

  Pokud oprávnění vyhovují, vytvoří instanci controlleru a zavolá příslušnou funkci. V opačném případě načte controller, který má na starosti chybové hlášky a zobrazí příslušnou chybu.

- `lib/resourcemodel.class.php` - abstraktní třída, od které dědí svojí funkčnost modely jednotlivých zdrojů. Zajišťuje vytváření nových požadavků, úpravu požadavků, načtení všech požadavků a obecné databázové operace.

- `app/controllers/ResourceController.php` - abstraktní třída, od které dále dědí controllery jednotlivých zdrojů. Sjednocuje metody, které následně komunikují s příslušným modelem. Obsahuje volání metod modelu jako je metoda pro vytvoření nového požadavku, zobrazení detailu požadavku a zobrazení všech požadavků daného typu.

- `lib/currentuser.class.php` - třída se statickými metodami pro práci s přihlášeným uživatelem. Pomocí této třídy probíhá ověřování práv přihlášeného uživatele, nastavit přepnutí uživatelského účtu na jiný účet nebo například ověřit, do kterých skupin uživatel patří.

- `lib/db.class.php` - třída pro získání instance databázového spojení. Implementována pomocí návrhového vzoru jedináček[1].
- `lib/job.class.php` - objekty této třídy představují jednotlivé požadavky a všechny jejich atributy jako jsou typ požadavku, zadavatel požadavku, aktuální stav požadavku a čas a datum vytvoření požadavku.

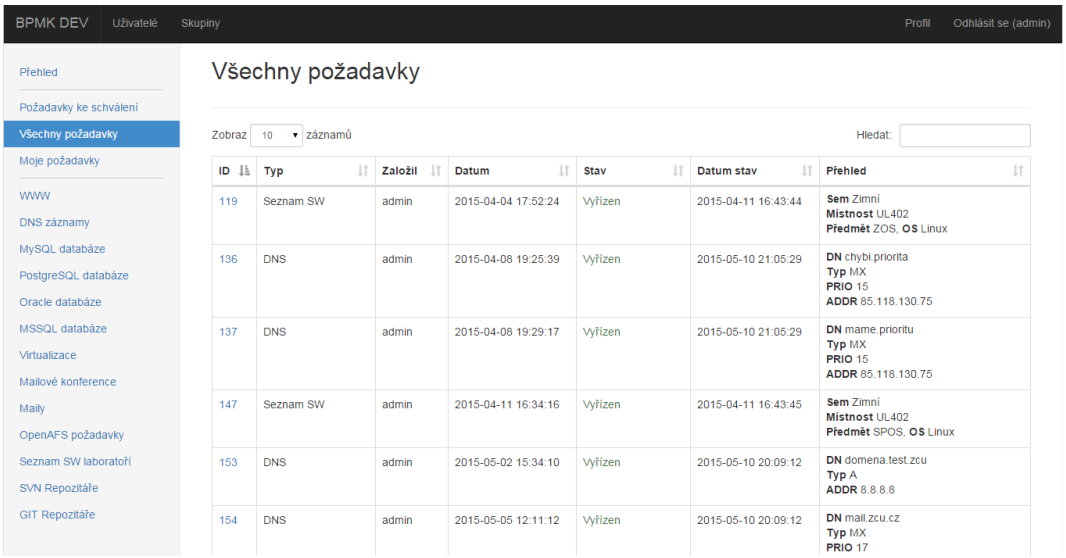
5.1.3 Vzhled

Pro vzhled webové stránky je použit `Twitter Bootstrap` [32], dále pouze bootstrap, což je front-end framework distribuován pod otevřenou licencí MIT. Díky Bootstrapu jsme schopni zajistit podporu uživatelského rozhraní na vícero rozlišeních. Součástí Bootstrapu jsou CSS, JavaScripty a často používané ikony.

Dále jsou použity knihony `jQuery` a `DataTables`. Knihovna jQuery je javascriptová knihovna, která usnadňuje práci s javascriptem svojí sadou funkcí. Lze jí využít k manipulaci s prvky na stránce, tvorbě animací či asynchronních volání ajax[2]. Knihovna DataTables je vybavena funkcemi obohacujícími jednoduché HTML tabulky. Nabízí například filtrování hodnot, seřazení sloupců a automatické stránkování. Vzhled webové stránky je zobrazen na obrázku 5.2.

[1] Zajistí přítomnost pouze jedné instance objektu napříč celou aplikací

[2] Asynchronní volání nám umožňují měnit obsah webových stránek bez nutnosti opětovného načtení stránky



Obrázek 5.2: Vzhled webové stránky

5.1.4 Stavy požadavku

Požadavek může být v některém z následujících stavů:

- `Čeká na schválení` - požadavek musí být nejdříve schválen.
- `Čeká na vyřízení` - požadavek je schválen a nyní čeká na vyřízení.
- `Vyřizuje se` - právě je vyřizován.
- `Vyřízen` - požadavek byl úspěšně vyřízen.
- `Zamítnut` - požadavek byl zamítnut oprávněnou osobou.
- `Chyba` - došlo k chybě při vyřizování požadavku.

Při zobrazení detailu požadavku můžeme jeho stav změnit ručně, například pokud po změně vstupních dat chceme data na cílovém serveru aktualizovat. Všechny změny požadavku, ať už ruční nebo automatické systémem, jsou zaznamenávány v databázi a jejich historii je možné zobrazit na detailu požadavku. V případě chyby při vykonávání požadavku si administrátor může zároveň zobrazit detailní výpis průběhu, při kterém nastala chyba.

5.1.5 Historie stavů požadavku

Všechny změny stavu požadavku je možné zobrazit na detailu požadavku. Změny jsou zaznamenávány pomocí databázového triggeru[3]. Trigger při každé změně stavu automaticky ukládá starý stav do vedlejší tabulky, z které jsme následně schopni zkonstruovat historii stavů daného požadavku.

Ukázka použitého triggeru:

```
CREATE TRIGGER jobs_state_trigger
   BEFORE UPDATE ON `jobs`
   FOR EACH ROW BEGIN
      IF NEW.state <> OLD.state
         THEN SET NEW.date_state_change = NOW();
         INSERT INTO `jobs_state_log`(`job_id`,`state_date`,
             `state`,`state_message_code`)
            VALUES (OLD.id, OLD.date_state_change, OLD.state,
               OLD.state_message_code);
      END IF;
   END
```

5.1.6 Ovládání

Popis důležitých operací na webu. U popisu prvků ovládání kromě `Přihlášení` se předpokládá přihlášený uživatel s dostatečnými právy.

Přihlášení

Uživatel, který neni přihlášen, je při návštěvě nejdřívě přesměrován na přihlašovací formulář. Pro přihlášení jsou použité tzv. PHP sessions, které řeší bezestavovost HTTP a umožňují udržení informací návštěvníka.

Uživatel je následně identifikován pomocí session identifikátoru, který je většinou uložen pomocí cookie[4]. Data jednotlivých session jsou uloženy na

[3] Databázový objekt spojen s tabulkou, spuštěný v případě splnění určitých událostí
[4] Malý textový soubor vygenerovaný web serverem a uložen u uživatele v prohlížeči

serveru ve speciálním adresáři a sdružena s uživateli pomocí session identifikátoru. Pokud se uživatel před přihlášením snažil přistoupit na jinou stránku než úvodní, je na ní po přihlášení a ověření práv následně přesměrován.

Navigace

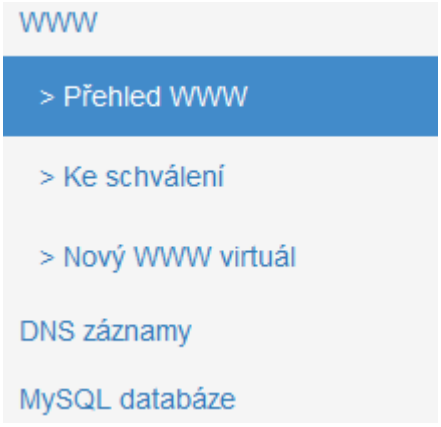
Hlavní navigace, viz obr. 5.3, se nachází v horní části stránky. Je tvořena odkazy sloužícími k administraci uživatelů. Zobrazení těchto odkazů je podmíněno vlastnictvým příslušných práv. Na odkazu `Uživatelé` můžeme prohlížet existující uživatele, vytvářet nové či modifikovat stávající. Jsme zároveň schopni přiřazovat uživatele do skupin, nastavovat přístupová práva uživatelům jak k jednotlivým částem webu tak i možnostem správy jednotlivých zdrojů. Odkaz `Skupiny` vede na seznam skupin a nabízí stejné množnosti jako `Uživatelé`.

Součástí menu jsou také odkazy `Profil`, který zobrazí profil přihlášeného uživatele a odkaz pro odhlášení.

Obrázek 5.3: Hlavní navigace

Navigace správy zdrojů

Po levé části se uživateli zobrazují odkazy na správu jemu dostupných typů zdrojů, možný vzhled menu lze vidět na obrázku 5.4. Po rozkliknutí odkazu se uživateli zobrazí přehled požadavků daného typu a podle dostupných práv i požadavky ke schválení nebo vytvoření nového požadavku.

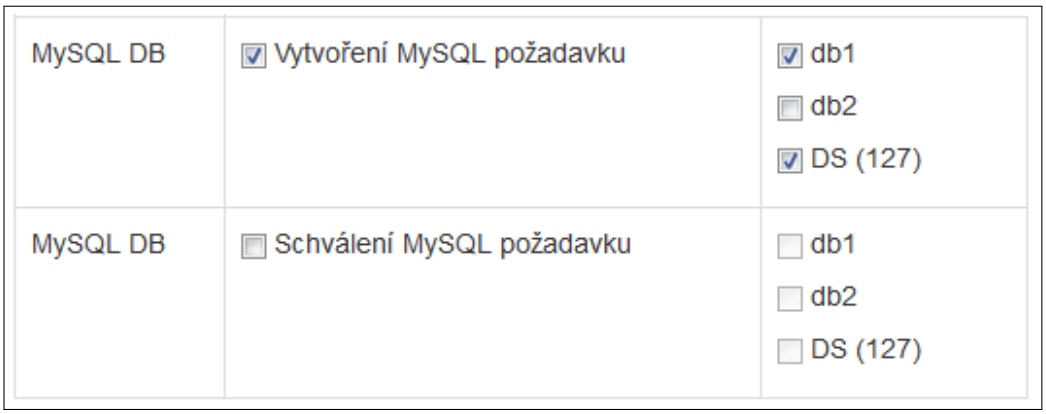


Obrázek 5.4: Část navigace zdrojů

Nastavení oprávnění

Nastavením oprávnění řídíme možnosti správy jednoho či více uživatelů. Práva lze nastavit jak skupinám tak jednotlivým uživatelům. Uživatelé mohou být členy žádné nebo jedné a více skupin a jejich výsledná práva jsou sjednocením práv všech skupin a samotných práv uživatele.

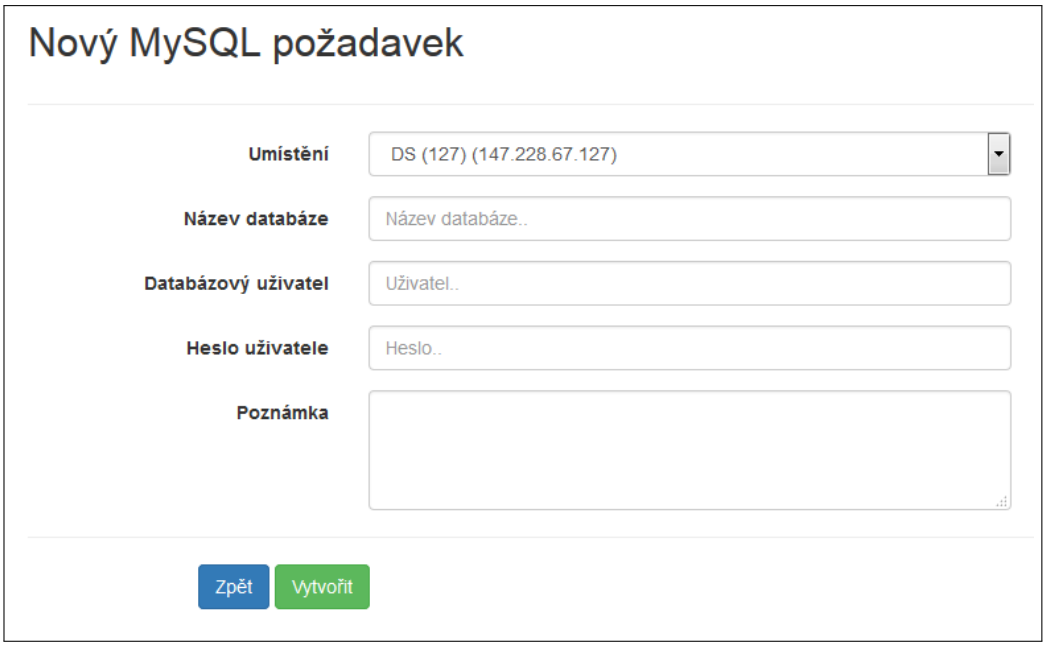
Práva jsou rozdělena na administraci uživatelů a skupin a práva pro správu jednotlivých typů zdrojů. U zdrojů je dále možnost specifikovat umístění, na které přiřazujeme práva, viz obr. 5.5. To nám v případě existence více serverů, na kterých je provozován stejný typ zdroje, umožní specifikovat ke kterým má uživatel přístup.



Obrázek 5.5: Ukázka možností při nastavování práv pro databázi MySQL, zleva: název oprávnění, akce oprávnění a cílové servery

Vytvoření požadavku

K vytvoření požadavku se dostaneme pomocí navigace umístěné na levé části stránky. Po vyplnění všech potřebných údajů tlačítkem `Vytvořit` založíme nový požadavek. Požadavek bude mít výchozí stav `Čeká na schválení` pokud nejsme oprávněni schválit tento typ požadavku s daným umístěním. Pokud oprávnění vlastníme, požadavku se automaticky nastaví stav `Čeká na vyřízení`. Příklad založení nového požadavku je zobrazen na obr. 5.6.



Obrázek 5.6: Příklad založení nové MySQL databáze

Schvalování požadavků

Oprávněné osoby mají možnost schvalovat požadavky ostatních uživatelů. Seznam požadavků všech typů čekajících na schválení lze zobrazit v menu na levé straně pod odkazem `Požadavky ke schválení`. Seznam čekajících požadavků konkrétního typu lze zobrazit při rozkliknutí menu položky požadovaného typu. Požadavky lze schválit hromadně ze seznamu pomocí zatrhávacího políčka v řádce tabulky nebo na detailu požadavku pomocí tlačítka `Schválit`. Po schválení je požadavek zařazen do fronty na vyřízení.

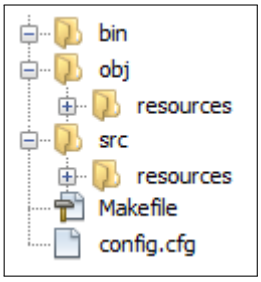
5.2 Zpracovávající část (démon)

Démon je napsán v jazyce C a je kompatibilní se standardem C11. Je určen pro operační systém Linux, konkrétně Debian GNU/Linux ale zároveň by měl být přeložitelný na většině systémech s knihovnami `mysqlclient`, `rt` a `pthread`. Program se skládá z několika vláken a to hlavního vlákna, dotazovacího vlákna a thread poolu[5]. Přístup do sdílené paměti mezi vlákny je zabezpečen pomocí mutexů[6]. Konfiguraci démona lze provést pomocí konfiguračního souboru, jehož jméno předáme jako argument při spuštění.

5.2.1 Adresářová struktura

Zdrojové soubory jsou rozděleny na dvě části a to na soubory zajišťující všeobecnou funkčnost programu a soubory (moduly) s podporou jednotlivých zdrojů. Struktura je vyobrazena na obrázku 5.7.

- `bin` - obsahuje po překladu výsledný spustitelný binární soubor s názvem `bpmk-daemon`.
- `obj` - slouží při předkladu pro uložení objektových souborů, které jsou následně použity při sestavení spustitelného souboru.
- `src` - zdrojové a hlavičkové soubory démona.



Obrázek 5.7: Adresářová struktura zdrojových souborů démonu

[5] Několik spuštěných vláken, kterým mohou být, většinou pomocí fronty, přiřazovány úkoly

[6] Mutex slouží k synchronizaci vláken, zamezuje více vláknům přístup do kritické sekce kódu

5.2.2 Přístup do databáze

Pro spojení s MySQL databází je využita knihovna `mysqlclient`. Pro připojení nejdříve použijeme funkci `mysql_init` a následně `mysql_real_connect`. Funkce `mysql_init` vrátí adresu na strukturu jménem `MYSQL`, která je dále použita při komunikaci s databází.

Každé vlákno, které vyžaduje přístup do databáze, vlastní svojí strukturu `MYSQL`, což nám umožní paralelní komunikaci s databází a odpadne nutnost používání mutexů pro přístup k této struktuře.

5.2.3 Hlavní vlákno

Hlavní vlákno se stará o inicializaci programu a všech jeho důležitých částí. Zároveň kontroluje, zda byl při spuštění předán argument, který by představoval cestu ke konfiguračnímu souboru.

Pokud byl program spuštěn s parametrem obsahujícím název konfiguračního souboru, je tento soubor načten a příslušné výchozí hodnoty proměnných změněny. Následně spustí logování, zapíše do logu hodnoty konfiguračních proměnných, inicializuje thread pool a nakonec dotazovací vlákno.

Po dokončení inicializace vlákno slouží k zachycení signálu SIGINT, který zahájí ukončování běhu programu.

5.2.4 Dotazovací vlákno

Slouží k cyklické kontrole požadavků čekajících na vyřízení v databázi. Při kontrole využívá limitu v SQL dotazu o velikosti volných míst fronty thread poolu. Interval je ve výchozí hodnotě nastaven na 10 sekund a lze ho měnit pomocí konfiguračního souboru.

Jsou-li volná místa ve frontě a existují-li požadavky čekající na vyřízení, čekající požadavky jsou zařazeny do fronty thread poolu a jejich stav je nastaven na hodnotu `executing`, která označuje právě vyřizované požadavky. Pokud byly všechny čekající požadavky předány k vyřízení nebo nebyly žádné nalezeny, je vlákno uspáno na nastavený interval. Nejsou-li volná místa ve frontě, je vlákno uspáno a čeká na signalizaci o uvolněném místu.

5.2.5 Thread pool

Pro pracovní vlákna je využito návrhového vzoru thread pool. Při startu je vytvořen určitý počet vláken, které mohou být následně použity pro zpracování požadavku. Požadavky jsou řazeny do fronty, která má většinou velikost, ne však nutně, stejnou jako počet vláken thread poolu.

Struktura použitá pro reprezentaci thread poolu:

```
typedef struct {
    /* velikost */
    int size;
    /* pocet aktivnich vlaken */
    int active;
    /* pole vlaken */
    pthread_t* threads;
    /* podminena promenna pro notifikace */
    pthread_cond_t notify;
    /* mutex pro podminenou promenou */
    pthread_mutex_t mutex;

    /* signalizace o uvolneni mista ve fronte */
    pthread_cond_t notify_waiting;

    /* velikost fronty */
    int queue_size;
    /* pocet prvku ve fronte */
    int queue_count;
    /* pocet prvku cekajicich na prirazeni vlakna */
    int queue_pending_count;
    /* pole ukolu */
    thread_pool_task_t *queue;
    /* ukazatel na zacatek pole ukolu */
    int queue_head;
    /* ukazatel na konec pole ukolu */
    int queue_tail;

    /* signalizuje ze pool bezi */
    int running;
    /* signalizuje ze jsme ve stavu ukoncovani poolu */
    int shutdown;
} thread_pool_t;
```

5.2.6 Zpracovávání požadavků

Požadavky jsou zpracovávány pracovními vlákny. Každé pracovní vlákno má svojí strukturu `MYSQL` pro přístup do databáze, z které jsou následně načteny specifické informace spojené s typem požadavku. Poté je provedena kontrola, zda jsou zadány všechny povinné hodnoty. Po úspěšné kontrole je zavolán kontrolní skript pro vyřízení požadavků s načtenými parametry.

Nastane-li během kontroly povinných hodnot chyba nebo vrátí-li kontrolní skript jinou hodnotu než 0, je u požadavku nastaven status na hodnotu signalizující chybu a uložena chybová hláška.

5.2.7 Logování

Důležité informace jsou při běhu programu ukládány do nastaveného logovacího souboru. Logování je zajištěno vlastním logovacím řešením, které je součástí programu. Použití vlastního řešení pro logování nám umožní specifikovat cestu k logovacímu souboru při startu programu a tak umožníme nastavení všech důležitých hodnot pomocí jednotného konfiguračního souboru. Ukládání do vlastního logovacího souborů nám umožní lepší orientaci při čtení logovacích zpráv. Jsou dostupné následující úrovně logování:

- `LOG_NONE(0)` - při této úrovni nebudou zaznamenány žádné zprávy.
- `LOG_ALL(1)` - nejnižší úroveň sloužící pro zaznamenání kritických informací.
- `LOG_ERR(2)` - úroveň pro důležité chybové zprávy.
- `LOG_WARN(3)` - varovná hlášení, většinou nezabrání pokračování programu.
- `LOG_INFO(4)` - informační zprávy. Slouží hlavně pro informaci o aktuálně prováděné akci.
- `LOG_DEBUG(5)` - ladící zprávy. Slouží pro odhalení chyb při vývoji.

5.2.8 Konfigurační soubor

Démonu můžeme při spuštění předat pomocí argumentu název konfiguračního souboru, kterým následně máme možnost změnit defaultní hodnoty některých proměnných. Konfigurovat můžeme následné proměnné:

- `db_charset` - kódování databáze.
- `db_host` - adresa databázového serveru.
- `db_name` - název databáze.
- `db_user` - uživatel pro přístup do databáze.
- `db_password` - heslo databázového uživatele.
- `db_reconnect_attempts` - počet pokusů o navázání spojení s databází.
- `db_reconnect_interval` - interval mezi jednotlivými pokusy o připojení.
- `job_poller_interval` - interval pro kontrolu požadavků dotazovacím vláknem.
- `log_buffer_sizel` - velikost pole pro logování zpráv.
- `log_file` - cesta k souboru, do kterého budou logovány zprávy.
- `tp_pool_size` - počet vláken thread poolu.
- `tp_queue_size` - velikost fronty thread poolu.
- `dispatch_file` - cesta k serverové části vykonávacího skriptu, volán pro vykonání požadavků.
- `log_level` - číselná hodnota reprezentující úroveň detailu logovacích zpráv zapsaných do souboru.
- `verbose_level` - číselná hodnota reprezentující úroveň detailu logovacích zpráv vypsaných do konzole.

5.2.9 Přeložení

Pro přeložení programu využijeme programu make a přiloženého souboru `Makefile`, který obsahuje sadu pravidel pro překlad. Zdrojové soubory rozbalíme do složky a následně se do ní přepneme příkazem `cd`. Poté stačí zavolat příkaz `make` a spustí se překlad, viz obr. 5.8.

```
martin@phobos:/$ cd /usr/src/bpmk-daemon
martin@phobos:/usr/src/bpmk-daemon$ make
gcc -Wall -pedantic -pthread -std=gnu11 -c src/config.c -o obj/c
gcc -Wall -pedantic -pthread -std=gnu11 -c src/thread_pool.c -o
gcc -Wall -pedantic -pthread -std=gnu11 -c src/error.c -o obj/er
gcc -Wall -pedantic -pthread -std=gnu11 -c src/db.c -o obj/db.o
```

Obrázek 5.8: Část výpisu při překladu programu

5.2.10 Spuštění

Po úspěšném přeložení se vytvoří soubor `bin/bpmk-daemon`. Pro spuštění stačí zadat do příkazové řádky `bin/bpmk-daemon` nebo se přepnout do složky `bin` a zadat příkaz `./bpmk-daemon`. Tímto postupem ovšem spustíme program jako regulérní proces, postup pro spuštění jako démona bude následně popsán v kapitole věnované instalaci.

5.3 Skripty

Vykonávání požadavků zajišťují shellové skripty, které jsou rozděleny na `serverové skripty` a `klientské skripty`. Pro svojí funkci využívají shellu `Bash`[7]. Při volání skriptů používáme jak u klienta tak u serveru jeden hlavní ovládací skript, kterému předáme parametrem typ zdroje, požadovanou operaci a případně další argumenty spojené s typem zdroje. Komunikace s klienty probíhá pomocí protokolu `SSH` a přenos souborů programem `rsync`.

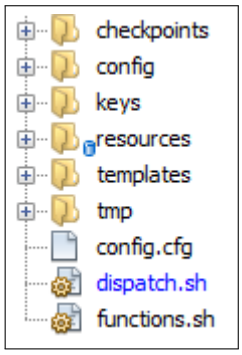
Díky použití ovládacího skriptu jsme schopni zjednodušit logiku démona, kdy stačí znát cestu k jednomu skriptu pro všechny typy zdrojů. Jelikož půjdou všechny požadavky přes tento ovládací skript, jsme zároveň schopni určit

[7] Unixový shel umožňující interpretaci příkazů

které typy zdrojů klient ovládá pouhým nastavením proměnných spjatých s typem zdroje na `1` pro povoleno nebo `0` pro zakázáno.

5.3.1 Adresářová struktura

Struktura souborů je podobná jak u serverové tak klientské části. Struktura serverové části je zobrazena na obrázku 5.9.



Obrázek 5.9: Adresářová struktura skriptů

- `checkpoints` - složka obsahující soubory s hodnotami proměnných uložených při kontrolních bodech.
- `config` - obsahuje konfigurační soubory. Všechny soubory z této složky s příponou `.cfg` jsou automaticky načteny.
- `keys` - složka serverové části, uchovává klíče použité pro SSH spojení s klienty.
- `resources` - skripty pro ovládání podle typu zdroje.
- `templates` - šablony konfiguračních souborů.
- `tmp` - adresář pro ukládání dočasných souborů.
- `config.cfg` - hlavní konfigurační soubor.
- `dispatch.sh` - kontrolní serverový skript, zajišťující volání ovládajících skriptů podle typu zdroje předaného parametrem.
- `functions.sh` - všeobecné funkce.

5.3.2 Ošetření chybových stavů

Pokud při vykonávání požadavku dojde k chybě, kterou může být například při vytváření databáze existence databáze s požadovaným názvem, skript je schopen dostat systém klienta do původního stavu, ať už je to smazáním nahraných souborů nebo vrácení změn provedených v konfiguračních souborech.

5.3.3 Ukládání stavu

Jelikož může při běhu skriptu dojít k jeho přerušení, je vhodné si ukládat aktuální stav při důležitých operacích, aby jsme byli schopni následně na tento stav navázat.

Při spuštění je vypočítáno jméno souboru pro ukládání stavu z názvu aktuálního skriptu a předaných parametrů. Pro výpočet jména jsou všechny argumenty spojeny do jednoho dlouhého řetězce a poté zahashovány pomocí SHA1 algoritmu. Následně je provedena kontrola, zda soubor s tímto názvem již existuje a pokud ano, jsou z něho načteny hodnoty proměnných.

Během vykonávání skriptu jsou do tohoto soubory v kontrolních bodech ukládány hodnoty proměnných s tím, že předchozí hodnoty jsou nejdříve smazány. Po úspěšném doběhnutí skriptu je soubor smazán.

5.3.4 Skripty

Skripty určené pro plnění úkolů pro jednotlivé zdroje jsou umístěny ve složce `resources` a pojmenovány prefixem `res` a typem zdroje, tedy například `reswww`.

WWW

Vytvoření nového virtuálního hostu probíhá pomocí konfiguračních souborů web serveru Apache, které jsou uloženy ve speciální složce. Jako vzor konfiguračního souboru slouží předpřipravená šablona, která je uložena v serverové části. Šablona je následně přenesena pomocí programu rsync na klienta a za-

volán kontrolní skript klienta, kterému mimo jiné parametry předáme i název šablony.

Dostupní virtuálni hosti jsou většinou uloženi v složce s konfiguračními soubory Apache, konkrétně v relativní cestě `sites-available` a aktuálně povolení ve složce `sites-enabled`. Zvykem je vytvořit konfiguraci ve složce `sites-available` a následně vytvořit symbolický odkaz do složky `sites-enabled`. Konfigurační soubory budeme pojmenovávat podle domény virtuálního hostu, za kterou připojíme koncovku `.conf` - bude-li doména `test.zcu.cz` potom výsledný soubor bude pojmenován `test.zcu.cz.conf`.

Šablona konfiguračního souboru je uložena v serverové části, aby bylo možné soubor měnit bez nutnosti jeho aktualizace na všech klientech. Šablona je nejdříve přenesena na klienta a následně jsou do ní vloženy potřebné hodnoty.

Po zavolání kontrolního skriptu klienta si nejdříve zkontroluje existenci souboru se šablonou a poté kontroluje, zda byly předány všechny potřebné argumenty. Následně ověřuje, zda již neexistuje virtuální host se stejnou doménou a to nejdříve kontrolou existence konfiguračního souboru se stejným jménem a následným průchodem již existujících konfiguračních souborů a porovnáváním domén.

Povolení nově vytvořeného konfiguračního souboru probíhá až po úspěšném splnění všech kontrol. Posledním krokem je kontrola syntaxe konfiguračního souboru pomocí příkazu `apachectl configtest` a pokud je v pořádku, zavoláme znovunačtení konfigurace serverem Apache, aby se změny projevily.

Ukázka vygenerovaného konfigurančího souboru virtualního hostu:

```
<VirtualHost *:80>
        ServerAdmin webmaster@kiv.zcu.cz
        ServerName test.kiv.zcu.cz

        DocumentRoot /var/www/test.kiv.zcu.cz/

        <Directory /var/www/test.kiv.zcu.cz/>
                AllowOverride all
                Order allow,deny
              allow from all
        </Directory>

        LogLevel warn
```

```
        ErrorLog ${APACHE_LOG_DIR}/test.kiv.zcu.cz.error.log
        CustomLog ${APACHE_LOG_DIR}/test.kiv.zcu.cz.access.log
            combined
</VirtualHost>
```

DNS záznamy

DNS záznamy mohou být typu `A`, `CNAME` nebo `MX`. Záznam typu `A` nasměruje doménu na konkrétní IP adresu, například `zcu.cz -> 147.228.58.47`. `CNAME` funguje jako alias, tedy k nasměrování jednoho jména na jiné, již existující, jméno. Záznam `MX` slouží k nasměrování emailů zasílaných na danou doménu na IP adresu. U MX záznamů můžeme zároveň specifikovat prioritu jednotlivých záznamů, která umožňuje nastavení záložních mailových serverů.

Jelikož má samotnou správu DNS na starosti organizace CIV, je třeba odeslat email s detailem požadavku. Pro úspěšné doručení požadavku musí být email odeslán z domény `zcu.cz`. Jako odesílací email použijeme email zadavatele požadavku, na který se bude očekávat i případná odpoveď.

Vytvoření a odesílání probíhá v serverové části. Pro vytvoření emailu použijeme šablonu, do které budou následně doplněny všechny zadané hodnoty. Zjištění emailu zadavatele probíhá pomocí protokolu LDAP z adresářového serveru `ldap.zcu.cz`.

Výsledný email může mít následující podobu:

```
Zadana zadost o vytvoreni DNS zaznamu uzivatelem feelus s
    emailem feelus@students.zcu.cz. Zadost o DNS zaznam typu A.
    Jmeno test.kiv.zcu.cz, hodnota 147.228.63.36
```

MySQL databáze

Pro přístup do databáze je použito administrátorského účtu, jehož údaje jsou uloženy v konfiguračním souboru `config/resdbmysql.cfg`. Při spuštění skriptu pro vytvoření databáze jsou nejdříve ověřeny povinné vstupní parametry, mezi které patří název databáze, databázový uživatel a heslo. Komunikace s databází probíhá pomocí konzolového klienta `libmysqlcli-`

`ent`, který umožňuje vykonávání dotazů přímo z příkazové řádky.

V prvním kroku je ověřeno spojení s databází. Následně ověříme, jestli již neexistuje databáze se stejným názvem a pokud ne, vytváříme novou. U MySQL účtu je spolu s jménem a heslem nutno uvést z jaké adresy se může uživatel připojit, například localhost, který umožní pouze lokální spojení nebo znak `%`, který zastupuje libovolnou adresu. Při vytváření účtu uživatele zkontrolujeme existenci uživatele se stejným jménem a adresou, neexistuje-li, vytvoříme jej. Po zajištění existence uživatele už mu jen stačí nastavit přístupová práva do nově vytvořené databáze.

PostgreSQL databáze

Vytvoření PostgreSQL databáze probíhá na podobném principu jako vytváření MySQL databáze. Rozdílem je určení adresy, z které se může uživatel připoji. Adresa není uváděna při vytváření uživatele ale následně v konfiguračním souboru `pg_hba.conf`.

MS SQL databáze

Jelikož jsou skripty spouštěny z linuxového systému a MS SQL databáze běží na systému Microsoft Windows, vytváření databáze probíhá ze serverové části a komunikace s databází je zajištěna pomocí `unixODBC` [33] a `FreeTDS` [34]. Údaje pro připojení jsou nastaveny v konfiguračních souborech `/etc/odbc.ini`, `/etc/odbcinst.ini` a `/etc/freetds.conf`. Po připojení je zbylý postup je stejný jako v případě MySQL databáze.

Oracle databáze

Při vytváření Oracle databáze použijeme pro komunikaci s databází programu `SQL*Plus`, který nabídne možnost spouštění dotazů do databáze pomocí příkazové řádky. U Oracle databáze se na rozdíl od MySQL a PostgreSQL nevytváří databáze do které se následně nastavují práva pro uživatele, ale každý vytvořený uživatel má svoje vlastní schéma, ve kterém jsou jeho objekty jako tabulky, pohledy a podobné.

Před vytvořením uživatele zkontrolujeme, zda existuje uživatel se stejným

jménem a pokud ne, vytváříme účet. Nově vytvořenému účtu je ještě třeba přidat práva pro připojení do databáze a to pomocí příkazu `GRANT CREATE SESSION`.

Virtualizace

Vytváření nového virtuálního stroje probíhá ze serverové části a je rozděleno na několik důležitých kroků. Před samotným vytvořením virtuálního stroje musíme zajistit existenci LVM, DRBD a iSCSI targetu na obou úložných serverech `storage1` a `storage2`. Ovládání úložných serverů a následně XEN hypervizorů probíhá klasicky pomocí ovládacího klientského skriptu a komunikaci zajišťuje protokol SSH.

LVM vytvoříme pomocí příkazu `lvcreate -L <velikost> -n <název> <LVM skupina>`. Po vytvoření LVM na obou serverech pokračujeme vytvořením synchronizovaného blokového zařízení DRBD. Pro vytvoření použijeme šablonu konfiguračního souboru, do které vložíme potřebné parametry, kterými jsou například název DRBD zařízení, DRBD port a název LVM. Následně je provedena kontrola konfiguračního souboru příkazem `drbdadm dump all`. Pokud nenastaly chyby, vytváříme nové zařízení příkazem `drbdadm create-md <název>` a vynutíme znovunačtení konfiguračních souborů DRBD. Jelikož v tomto stavu DRBD neví který disk má aktuální data, musíme na jednom ze serverů vynutit přepsání dat protějšku. Po vytvoření a načtení nového DRBD začne synchronizace dat, kterou necháme doběhnout. Stav synchronizace lze pozorovat v `/proc/drbd` nebo pomocí programu `drbdadm`.

Po dokončení synchronizace DRBD je třeba vytvořit iSCSI, které vytvoříme přidáním nové logické jednotky do hlavního konfiguračního souboru programu `istgt` na obou úložných servech. Následně službu `istgt` restartujeme, aby se projevily změny. Pro přístup k diskům budou virtuální stroje používat `dm-multipath`, proto na všech XEN hypervizorech musíme spustit načtení nově dostupných iSCSI targetů. Následně se vytvoří nový multipath, pro který definujeme alias pomocí identifikátoru UUID. UUID nově vytvořeného multipathu zjistíme po výpisu příkazu `multipath -ll`, kde najdeme záznam bez existujícího aliasu. Nalezené UUID a vybraný alias následně vložíme do souboru `/etc/multipath.conf`.

Po zajištění přístupu k disku můžeme na všechny XEN hypervizory nahrát konfigurační soubor nového virtuálního stroje. Posledním krokem je vy-

tvoření virtuálního stroje na vybraném serveru příkazem `xl create <cesta ke konfiguračnímu souboru>`.

Mailové konference

Mailové konference jsou spravovány programem `mailman`. Před vytvořením je třeba zkontrolovat, zda neexistuje konference se stejným názvem. Následně konferenci vytvoříme příkazem `newlist <nazev> <administrator> <heslo>`. Po vytvoření je třeba nastavit emailové aliasy pro nově vytvořenou konferenci, které jsou vypsány skriptem `newlist` po vytvoření konference a následně je vložíme do `/etc/aliases` a zavoláme příkaz `newaliases` pro obnovení databáze aliasů.

Mailové účty

Pro vytvoření nového mailového účtu stačí na cílovém serveru zajistit existenci systémového uživatele se stejným jménem. Pokud vytváříme alias nebo virtuální alias, je třeba nejdříve zkontrolovat existenci aliasu v `/etc/aliases` resp. `/etc/postfix/virtual`. Pokud již existuje, budeme k tomuto aliasu přidávat nového mailového uživatele. V opačném případě vytvoříme alias a to přidáním řádky `<alias>: <uzivatel>` do `/etc/aliases` nebo v případě virtuálního aliasu `<alias> <uzivatel>` do `/etc/postfix/virtual`. Při přidávání nového uživatele k existujícímu aliasu zároveň kontrolujeme, zda již tento alias není nasměrován na zadaného uživatele, aby nedocházelo k duplicitním záznamům.

OpenAFS

Vytvoření AFS svazku je provedeno příkazem `vos create -server <server> -partition <oddil> -name <nazev>`. Po vytvoření svazku je třeba vytvořit přípojný bod příkazem `fs mkmount <cesta> <nazev svazku>`. Nastavení přístupových práv pro čtení na nově vytvořený svazek zajistíme příkazem `fs setactl <cesta> -acl <uzivatel> read`. Posledním krokem je nastavení kvóty svazku pomocí příkazu `fs setquota -path <cesta> -max <kvóta>`.

SVN repozitáře

Příkazem `svnadmin create -fs-type fsfs <cesta>` vytvoříme v uvedené cestě nový SVN repozitář. Následně je třeba zajistit přístup do repozitáře pomocí Apache virtuálního hostu. Ze serverové části přeneseme na klienta šablonu virtuálního hostu do které vložíme potřebné údaje.

Ověřování uživatelů je zajištěno Apache Kerberos modulem. Do konfiguračního souboru virtuálního hostu stačí vložit za direktivu `Require user` adresy všech uživatelů, kteří mají mít do repozitáře přístup. Po dokončení změn v konfiguračním souboru spustíme kontrolu syntaxe a vynutíme Apache znovu načíst konfigurační soubory.

Ukázka vygenerovaného konfigurančího souboru virtualního hostu pro SVN:

```
<Location /svn-testrepo>

    DAV svn
    SVNPath /home/svn/testrepo

        AuthType Kerberos
        KrbAuthRealms ZCU.CZ
        KrbServiceName http/ares.fav.zcu.cz
        Krb5Keytab /etc/krb5.keytab.http
        KrbVerifyKDC off
        KrbSaveCredentials on
        KrbDelegateBasic on
        AuthName "Subversion Repository"
        Require user feelus@ZCU.CZ

</Location>
```

GIT repozitáře

Repozitáře jsou spravovány programem `Gitolite`. Pro úspěšné fungování je třeba nastavit v konfiguračním souboru cestu k spustitelnému souboru gitolite, složku s klíči pro přístup uživatelů do repozitářů (klasicky `.gitolite/keydir`) a cestu ke konfiguračnímu souboru (klasicky `.gitolite/conf/gitolite.conf`). Samotné vytvoření repozitáře probíhá vložením řetězce `repo <název>` do konfiguračního souboru a jeho následnou kompilací příkazem

`gitolite compile; gitolite trigger POST_COMPILE` se automaticky vytvoří nově přidané repozitáře.

Pro nastavení přístupu uživatelů do repozitářů stačí v konfiguračním souboru zadat typ přístupu, kterým může být například `R` omezující přístup pouze pro čtení. Za uvedeným typem přístupu a znakem `=` je výsledný seznam uživatelů. Samotné ověření uživatele probíhá pomocí SSH klíčů, které jsou pojmenovány ve formátu `<uživatel>.pub` a uloženy ve složce `.gitolite/keydir`.

6 Instalace

V následující kapitole je popsána kompletní instalace a konfigurace implementovaného řešení na čistě nainstalovaný operační systém Debian GNU/Linux.

6.1 Instalace serveru

Nejaktuálnější verzí (květen 2015) distribuce Debian je Debian 8 s označením Jessie, ovšem instalace na jiné verzi Debianu se nebude příliš lišit. Dále budeme předpokládat nainstalovaný a plně aktualizovaný systém.

6.1.1 Apache a PHP

Apache bude sloužit jako webový server pro webové rozhraní programu. Budeme potřebovat modul `ssl` pro podporu šifrované komunikace protokolem HTTPS, `rewrite` pro povolení manipulaci s URL a modul `php5` pro podporu jazyka PHP. Pro PHP budeme následně potřebovat modul `mcrypt` pro vytvoření soli při hashování hesel a modul `mysql` pro komunikaci s databází.

SSL certifikát pro zabezpečenou komunikaci můžeme získat od důvěryhodné certifikační autority nebo vygenerovat svůj vlastní. V našem případě budeme používat vlastní certifikát vygenerovaný příkazem `openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout bpmk-ssl.key -out bpmk-ssl.crt`.

Bude se jednat o virtuálního hosta, který bude poslouchat na IP adrese `147.228.63.36` a portech `80` pro nezabezpečenou verzi a `443` pro zabezpečenou. Zdrojové kódy webu budou následně v adresáři `/var/www/bpmk`, certifikát v `/etc/ssl/bpmk-ssl.cert` a soukromý klíč spojený s certifikátem v `/etc/ssl/bpmk-ssl.key`.

Instalaci PHP, Apache a potřebných modulů provedeme následující sekvencí příkazů:

```
$ apt-get install php5 php5-mcrypt php5-mysql
$ apt-get install apache2
$ apt-get install libapache2-mod-php5
```

```
$ apt-get install php-pear php5-dev libpam0g-dev
```

Následně povolíme modul pro šifrovanou komunikaci a modul pro přepis URL:

```
$ a2enmod ssl rewrite
```

Následně vytvoříme konfigurační soubor pro nezabezpečenou verzi virtuálního hostu, která poběží na klasickém protokolu HTTP, portu 80 a návštěvníky bude přesměrovávat na protokol HTTPS a port 443.

Konfigurační soubor bude v `/etc/apache2/sites-available/bpmk.conf`:

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        RewriteEngine on
        ReWriteCond %{SERVER_PORT} !^443$
        RewriteRule ^/(.*) https://%{HTTP_HOST}/£1 [NC,R,L]
</VirtualHost>
```

Nainstalujeme modul pro PAM autentizaci, umožňující ověřování pomocí systémových uživatelů:

```
$ pecl install pam
$ echo "extension=pam.so" > /etc/php5/apache2/conf.d/20-pam.ini
```

Konfigurační soubor pro zabezpečený virtuální host `/etc/apache2/sites-available/bpmk-ssl.conf`:

```
<VirtualHost *:443>
        SSLEngine On
        SSLCertificateFile /etc/ssl/bpmk-ssl.crt
        SSLCertificateKeyFile /etc/ssl/bpmk-ssl.key

        ServerAdmin webmaster@localhost
        ServerName bpmk

        DocumentRoot /var/www/bpmk/pub

        <Directory /var/www/bpmk/pub>
                Options Indexes FollowSymLinks
```

```
                AllowOverride All
                Order allow,deny
                allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/bpmk.error.log

        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/bpmk.access.log combined
</VirtualHost>
```

Následně nakonfigurujeme Apache aby naslouchal na adrese `147.228.63.36` a portech `80` a `443`. Konfigurace proběhne v souboru `/etc/apache2/ports.conf` ve kterém přepíšeme původní direktivu `Listen` na následující:

```
Listen 147.228.63.36:80

<IfModule ssl_module>
        Listen 147.228.63.36:443
</IfModule>
```

Povolení obou virtuálních hostů a následný restart Apache serveru proběhne pomocí příkazů:

```
$ a2ensite bpmk bpmk-ssl
$ service apache2 restart
```

6.1.2 MySQL

Databáze MySQL bude sloužit pro komunikaci jednotlivých částí programu. Pro databázi máme strukturu v souboru `mysql_struktura.sql` a defaultní data v souboru `mysql_default.sql` ve složce s webem, u nás tedy v `/var/www/bpmk`.

Instalace MySQL databáze:

```
$ apt-get install mysql-server
```

Vytvoření databáze a naplnění databáze defaultními daty, bude potřeba za-

dat heslo zvolené při instalaci databáze:

```
$ mysql -e "CREATE DATABASE bpmk;" -p
$ mysql -p bpmk < /var/www/bpmk/mysql_struktura.sql
$ mysql -p bpmk < /var/www/bpmk/mysql_default.sql
```

Vytvoření uživatele pro web a démona:

```
$ mysql -e "CREATE USER 'bpmk'@'localhost' identified by
    'tajneheslo';" -p
$ mysql -e "GRANT ALL PRIVILEGES ON bpmk.* TO 'bpmk'@'localhost'
    IDENTIFIED BY 'tajneheslo';" -p
```

6.1.3 Web

Obsah adresáře `bpmk-web` překopírujeme do složky `/var/www/bpmk`. Následně je potřeba v konfiguračním souboru nastavit přihlašovací údaje do databáze.

V souboru `/var/www/bpmk/config/config.php` upravíme následující hodnoty:

```
# Database info
define("DB_HOST", "localhost");
define("DB_NAME", "bpmk");
define("DB_USER", "bpmk");
define("DB_PASSWORD", "tajneheslo");
```

Po nastavení údaju pro přístup databáze můžeme ověřit funkčnost webu zadáním adresy `https://147.228.63.36` do webového prohlížeče. Po úspěšném načtení by se měl zobrazit přihlašovací formulář, do kterého zadáme výchozí přihlašovací údaje: jméno `admin` a heslo `admin`. Následně jsme přihlášeni a systém nám doporučuje změnu hesla administrátorského účtu.

6.1.4 Démon

Překopírujeme obsah adresáře `bpmk-daemon` do složky `/usr/src/bpmk-daemon`.

Pro přeložení a posílání emailů musíme nainstalovat následující balíčky:

```
$ apt-get install mailutils postfix
$ apt-get install libmysqlclient-dev build-essential
```

Nastavíme přístup do databáze v souboru `/usr/src/bpmk-daemon/config.cfg`:

```
db_host=localhost
db_user=bpmk
db_password=tajneheslo
db_name=bpmk
```

Přeložení démona provedeme přepnutím do složky `/usr/src/bpmk-daemon` a spustíme:

```
$ make
```

Následně vytvoříme skript, který bude sloužit pro démonizaci procesu a jeho následné spuštění či zastavení. Skript uložíme do umístění `/etc/init.d/bpmk-daemon` s obsahem:

```
#!/bin/sh

set -e

NAME=bpmk-daemon
PIDFILE=/var/run/$NAME.pid

DAEMON=/usr/src/bpmk-daemon/bin/bpmk-daemon
DAEMON_OPTS="/usr/src/bpmk-daemon/config.cfg"

export PATH="${PATH:+$PATH:}/usr/sbin:/sbin"

case "$1" in
  start)
        echo -n "Starting daemon: "$NAME
        start-stop-daemon --start -b --quiet --pidfile $PIDFILE
            --make-pidfile --exec $DAEMON -- $DAEMON_OPTS
        echo "."
        ;;
  stop)
        echo -n "Stopping daemon: "$NAME
        start-stop-daemon --stop --quiet --oknodo --pidfile
            $PIDFILE
```

```
        echo "."
        ;;
  restart)
        echo -n "Restarting daemon: "$NAME
        start-stop-daemon --stop --quiet --oknodo --retry 30
            --pidfile $PIDFILE
        start-stop-daemon --start --quiet --pidfile $PIDFILE
            --exec $DAEMON -- $DAEMON_OPTS
        echo "."
        ;;

  *)
        echo "Usage: "$1" {start|stop|restart}"
        exit 1
esac

exit 0
```

Proces nejprve ručně pustíme, následně nastavíme potřebná práva a zajistíme start při spuštění systému:

```
$ chmod +x /etc/init.d/bpmk-daemon
$ chmod 755 /etc/init.d/bpmk-daemon
$ update-rc.d bpmk-daemon defaults
```

6.1.5 Serverové skripty

Obsah adresáře `bpmk-daemon-control-srv` překopírujeme do `/usr/src/bpmk-daemon-control-srv`.

Nainstalujeme balík pro komunikaci s LDAP serverem:

```
$ apt-get install ldaputils
```

Vygenerujeme SSH klíče pro SSH a RSYNC:

```
$ cd /usr/src/bpmk-daemon-control-srv/
$ ssh-keygen -f keys/id_rsa_control
$ ssh-keygen -f keys/id_rsa_rsync
```

6.1.6 Klientské scripty

Obsah adresáře `bpmk-daemon-control-client` překopírujeme do `/usr/src/bpmk-daemon-control-client` na klientovi. Pokud se některé hodnoty liší od zde použitých, lze je konfigurovat pro příslušné typy zdrojů ve složce `/usr/src/bpmk-daemon-control-client/config`.

Soubor `/usr/src/bpmk-daemon-control-client/config/resources.cfg` slouží pro zakázání či povolení jednotlivých typů zdrojů, na následující ukázce je povolen zdroj WWW a zakázán zdroj MySQL:

```
RES_WWW_ENABLED=1
RES_DBMYSQL_ENABLED=0
```

Pro omezení přístupu uživatele přes SSH musíme přidat do autorizovaných klíčů uživatele `root` veřejnou část našeho vygenerovaného kliče `id_rsa_control.pub`. Veřejnou část si můžeme uložit do schránky a do souboru `/root/.ssh/authorized_keys` vložíme následující řádku:

```
command="/usr/src/bpmk-daemon-control-client/control.sh
    ${SSH_ORIGINAL_COMMAND#*
    }",no-port-forwarding,no-x11-forwarding,no-agent-forwarding
    <obsah id_rsa_control.pub>
```

Vytvoříme uživatele, který bude sloužit k nahrávání dat programem `rsync` a potřebné složky:

```
$ adduser bpmk-rsync
$ mkdir --parents /home/bpmk-rsync/bin/
$ mkdir /home/bpmk-rsync/upload/
$ mkdir /home/bpmk-rsync/.ssh/
```

Nainstalujeme `rsync` a použijeme skript `rrsync`, který nám umožní specifikovat složku, do které má uživatel přístup:

```
$ apt-get install rsync
$ gunzip -c /usr/share/doc/rsync/scripts/rrsync.gz >
    /home/bpmk-rsync/bin/rrsync
```

Nastavíme vlastnictví složky `/home/bpmk-rsync` pro uživatele `bpmk-rsync`:

```
$ chown -R bpmk-rsync:bpmk-rsync /home/bpmk-rsync/
```

Omezíme přístup přes SSH uživateli bpmk-rsync pouze na zavolání příkazu `bin/rrsync` a nahrávání souborů do složky `upload/`. Do schránky si zkopírujeme obsah souboru `id_rsa_rsync.pub` ze serverové části.

Do souboru `/home/bpmk-rsync/.ssh/authorized_keys` vložíme řádku:

```
command="$HOME/bin/rrsync /home/bpmk-rsync/upload/"
,no-agent-forwarding,no-port-forwarding,no-pty,no-user-rc,
no-X11-forwarding <obsah id_rsa_rsync.pub>
```

7 Testy

Aby jsme ověřili správné fungování administrace zdrojů, je třeba ověřit provedení úkolů a zároveň chování v případě vynucené chyby. Budeme ověřovat funkčnost klientských skriptů pomocí testovacích skriptů ve složce `bpmk-control-tests`, které umožní otestování všech typů zdrojů.

Při ověření chování v případě vynucené chyby, tedy negativních testů, bychom ideálně testy prováděli splněním. Připravené skripty ovšem testují pouze korektní vykonání požadavku a tak i při správné reakci na chybu bude výsledek testu chybný, ovšem výsledek snadno ověříme pomocí chybové hlášky.

7.1 Připravení testů

Nejprve rozbalíme obsah `bpmk-control-tests` do složky se serverovými skripty, v našem případě tedy do `/usr/src/bpmk-daemon-control-srv/` a výsledná složka s testy bude `/usr/src/bpmk-daemon-control-srv/tests/`. Jednotlivé testy lze spouštět samostatně z konzole pomocí názvu skriptu testu a předání potřebných parametrů, nebo pomocí hromadného spouštěcího skriptu `run_tests.sh`. Pro konfiguraci parametrů jsou v hlavičce spouštěcího skriptu definovány proměnné s argumenty pro příslušné testy zdrojů, pojmenovány ve formátu `<ZDROJ>_ARGS`, například tedy `WWW_ARGS`. Pro lepší orientaci ve výsledku testu budou provedené testy spouštěny po jednom hromadným spouštěčem.

7.2 WWW

Test ověřující vytvoření WWW virtuálního hostu a schopnost předcházet neočekávané chybě v podobě existence konfiguračního souboru virtuálního hostu se stejným názvem.

7.2.1 Ověření funkčnosti

Vstupní data

```
Umisteni: 147.228.67.127
Adresa virtualu: test.zcu.cz
Aliasy virtualu: alias.test.zcu.cz
Uzivatele: feelus
```

Výsledek testu

```
[1/1] - Testuji WWW s argumenty: -d 147.228.67.127 -m
   test.zcu.cz -s alias.test.zcu.cz -a feelus: [OK]
```

7.2.2 Zareagování na chybu

Před spuštěním testu nejdříve ručně vytvoříme soubor `/etc/apache2/sites-available/testchyba.zcu.cz.conf`.

Vstupní data

```
Umisteni: 147.228.67.127
Adresa virtualu: testchyba.zcu.cz
Aliasy virtualu: alias.testchyba.zcu.cz
Uzivatele: feelus
```

Výsledek testu

```
[1/1] - Testuji WWW s argumenty: -d 147.228.67.127 -m
   testchyba.zcu.cz -s alias.testchyba.zcu.cz -a feelus: [CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
Splnenych: 0
Chybnych: 1
======================================================
Test WWW CHYBA
Chyba - cilovy soubor
   /etc/apache2/sites-available/testchyba.zcu.cz.conf jiz
```

```
existuje
```

7.3 DNS

Test ověřující úspěšné odeslání emailu s požadavkem o založení nového DNS záznamu.

7.3.1 Ověření funkčnosti

Vstupní data

```
Typ: A
Domenove jmeno: test.zcu.cz
Host: 147.228.67.127
```

Požadavky na DNS jsou řešeny emaily organizaci CIV a tak pro úspěšné ověření funkčnosti dočasně změníme odchozí adresu na testovací adresu a ověříme přijetí emailu.

Ukázka přijatého mailu:

```
From: feelus@students.zcu.cz
To: email@example.com
Subject: DNS Zadost typu A
Message-ID: <5571e4f4.OYpHYzOco9PvX5sR%feelus@students.zcu.cz>
User-Agent: Heirloom mailx 12.5 6/20/10
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Zadana zadost vytvoreni DNS zaznamu uzivatelem feelus s emailem
   feelus@students.zcu.cz. Zadost o DNS zaznam typu A. Jmeno
   test.zcu.cz, hodnota 147.228.67.127
```

7.4 MySQL databáze

Test ověřující úspěšné vytvoření MySQL databáze a reakci na neočekávanou chybu v podobě existence databáze se stejným jménem.

7.4.1 Ověření funkčnosti

Vstupní data

```
Umisteni: 147.228.67.127
Nazev databaze: databaze_mysql
Databazovy uzivatel: uzivatel_mysql
Heslo uzivatele: sUp3rtajn3h3slo
```

Výsledek testu

```
[1/1] - Testuji MYSQL s argumenty: -d 147.228.67.127 -m
   databaze_mysql -a uzivatel_mysql -p sUp3rtajn3h3slo: [OK]
```

Zároveň můžeme ověřit přímo spojení s databází příkazem:

```
$ mysql -u uzivatel_mysql -psUp3rtajn3h3slo -h 147.228.67.127
```

7.4.2 Zareagování na chybu

Před spuštěním testu ručně vynutíme chybu již existující databáze vytvořením databáze s názvem `databaze_mysql_chyba`.

Vstupní data

```
Umisteni: 147.228.67.127
Nazev databaze: databaze_mysql_chyba
Databazovy uzivatel: uzivatel_mysql
Heslo uzivatele: sUp3rtajn3h3slo
```

Výsledek testu

```
[1/1] - Testuji MYSQL s argumenty: -d 147.228.67.127 -m
    databaze_mysql_chyba -a uzivatel_mysql -p sUp3rtajn3h3slo:
    [CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
Splnenych: 0
Chybnych: 1
======================================================
Test MYSQL CHYBA
Chyba - databaze databaze_mysql_chyba jiz existuje
```

7.5 PostgreSQL databáze

Test ověřující úspěšné vytvoření PostgreSQL databáze a reakci na neočekávanou chybu v podobě existence databáze se stejným názvem.

7.5.1 Ověření funkčnosti

Vstupní data

```
Umisteni: 147.228.67.127
Nazev databaze: databaze_postgres
Databazovy uzivatel: uzivatel_postgres
Heslo uzivatele: sUp3rtajn3h3slo
```

Výsledek testu

```
[1/1] - Testuji POSTGRESQL s argumenty: -d 147.228.67.127 -m
    databaze_postgres -a uzivatel_postgres -p sUp3rtajn3h3slo:
    [OK]
```

Zároveň můžeme ověřit přímo spojení s databází příkazem:

```
$ psql -d databaze_postgres -U feelus -h 147.228.67.127
```

7.5.2 Zareagování na chybu

Před spuštěním testu ručně vynutíme chybu již existující databáze vytvořením databáze s názvem `databaze_postgres_chyba`.

Vstupní data

```
Umisteni: 147.228.67.127
Nazev databaze: databaze_postgres_chyba
Databazovy uzivatel: uzivatel_postgres
Heslo uzivatele: sUp3rtajn3h3slo
```

Výsledek testu

```
[1/1] - Testuji POSTGRESQL s argumenty: -d 147.227.67.127 -m
   databaze_postgres_chyba -a uzivatel_postgres -p
   sUp3rtajn3h3slo[CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
Splnenych: 0
Chybnych: 1
======================================================
Test POSTGRES CHYBA
Kontroluji, zda databaze databaze_postgres_chyba existuje
Chyba - databaze databaze_postgres_chyba jiz existuje
```

7.6 Oracle databáze

Test ověřující úspěšné vytvoření Oracle účtu a reakci na neočekávanou chybu v podobě existence uživatele se stejným jménem.

7.6.1 Ověření funkčnosti

Vstupní data

```
Umisteni: 147.228.67.127
```

```
Databazovy uzivatel: uzivatel_oracle
Heslo uzivatele: sUp3rtajn3h3slo

[1/1] - Testuji ORACLE s argumenty: -d 147.228.67.127 -m
    uzivatel_oracle -p sUp3rtajn3h3slo: [OK]
```

Zároveň můžeme ověřit přímo spojení s databází příkazem:

```
sqlplus uzivatel_oracle/sUp3rtajn3h3slo
```

7.6.2 Zareagování na chybu

Před spuštěním testu ručně vynutíme chybu již existujícího uživatele vytvořením uživatele s názvem `uzivatel_oracle_chyba`.

Vstupní data

```
Umisteni: 147.228.67.127
Databazovy uzivatel: uzivatel_oracle_chyba
Heslo uzivatele: sUp3rtajn3h3slo
```

Výsledek testu

```
[1/1] - Testuji ORACLE s argumenty: -d 147.228.67.127 -m
    uzivatel_oracle_chyba -p sUp3rtajn3h3slo: [CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
Splnenych: 0
Chybnych: 1
======================================================
Test ORACLE CHYBA
Kontroluji, zda existuje databazovy uzivatel
    UZIVATEL_ORACLE_CHYBA
Chyba - uzivatel se jmenem uzivatel_oracle_chyba jiz existuje
```

7.7 MSSQL databáze

Test ověřující úspěšné vytvoření MS SQL databáze a reakci na neočekávanou chybu v podobě existence databáze se stejným názvem.

7.7.1 Ověření funkčnosti

Vstupní data

```
Umisteni: 147.228.67.127
Nazev databaze: databaze_mssql
Databazovy uzivatel: uzivatel_mssql
Heslo uzivatele: sUp3rtajn3h3slo

[1/1] - Testuji MSSQL s argumenty: -d 147.228.67.127 -m
   databaze_mssql -a uzivatel_mssql -p sUp3rtajn3h3slo: [OK]
```

7.7.2 Zareagování na chybu

Před spuštěním testu ručně vynutíme chybu již existující databáze vytvořením databáze s názvem `databaze_mssql_chyba`.

Vstupní data

```
Umisteni: 147.228.67.127
Nazev databaze: databaze_mssql_chyba
Databazovy uzivatel: uzivatel_mssql
Heslo uzivatele: sUp3rtajn3h3slo
```

Výsledek testu

```
[1/1] - Testuji MSSQL s argumenty: -d 147.228.67.127 -m
   databaze_mssql_chyba -a uzivatel_mssql -p sUp3rtajn3h3slo:
   [CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
Splnenych: 0
```

```
Chybnych: 1
======================================================
Test MSSQL CHYBA
Kontroluji, zda databaze databaze_mssql_chyba existuje
Chyba - databaze s nazvem databaze_mssql_chyba jiz existuje
```

7.8 Virtualizace

Test ověřující úspěšné vytvoření nového virtuálního stroje a reakci na neočekávanou chybu v podobě existence LVM se stejným názvem.

7.8.1 Ověření funkčnosti

Vstupní data

```
Typ: Vyuka
Hostname: virtualizace_hostname
IP adresa: 147.228.67.128
MAC adresa: AA:BB:AA:BB:AA:BB
```

Výsledek testu

```
[1/1] - Testuji VIRTUALIZACI s argumenty: -s debian -p vyuka -h
   virtualizace_hostname -a 147.228.67.128 -w
   AA:BB:AA:BB:AA:BB: [OK]
```

7.8.2 Zareagování na chybu

Před spuštěním testu ručně vytvoříme LVM s názvem `virtualizace_hostname_chyba`.

Vstupní data

```
Typ: Vyuka
Hostname: virtualizace_hostname_chyba
IP adresa: 147.228.67.128
```

```
MAC adresa: AA:BB:AA:BB:AA:BB
```

Výsledek testu

```
[1/1] - Testuji VIRTUALIZACI s argumenty: -s debian -p vyuka -h
    virtualizace_hostname_chyba -a 147.228.67.128 -w
    AA:BB:AA:BB:AA:BB: [CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
Splnenych: 0
Chybnych: 1
======================================================
Test VIRTUALIZACE CHYBA
Chyba - doslo k chybe pri vytvareni LVM prikazem: lvcreate -L 2G
    -n kiv.virtualizace_hostname_chyba vgstorage, vylsedek:
    Logical volume "kiv.virtualizace_hostname_chyba" already
    exists in volume group "vgstorage"
```

7.9 Mailové konference

Test ověřující úspěšné vytvoření nové mailové konference a reakci na neočekávanou chybu v podobě existující konference se stejným názvem.

7.9.1 Ověření funkčnosti

Vstupní data

```
Umisteni: 147.228.67.127
Jmeno konference: nazev_konference
Mail admina: feelus@students.zcu.cz
Heslo konference: sUp3rtajn3h3slo
```

Výsledek testu

```
[1/1] - Testuji MAILOVOU KONFERENCI s argumenty: -d
    147.228.67.127 -m nazev_konference -a feelus@students.zcu.cz
    -p sUp3rtajn3h3slo: [OK]
```

Zároveň můžeme ručně ověřit existenci konference a jejího administrátora příkazem:

```
$ /usr/lib/mailman/bin/list_owners nazev_konference
feelus@students.zcu.cz
```

7.9.2 Zareagování na chybu

Před spuštěním testu ručně vytvoříme konferenci s názvem `nazev_konference_chyba`.

Vstupní data

```
Umisteni: 147.228.67.127
Jmeno konference: nazev_konference_chyba
Mail admina: feelus@students.zcu.cz
Heslo konference: sUp3rtajn3h3slo
```

Výsledek testu

```
[1/1] - Testuji MAILOVOU KONFERENCI s argumenty: -d
   147.228.67.127 -m nazev_konference_chyba -a
   feelus@students.zcu.cz -p sUp3rtajn3h3slo: [CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
Splnenych: 0
Chybnych: 1
============================================================
Test MAILOVE KONFERENCE CHYBA
Chyba - konference s timto nazvem 'nazev_konference_chyba' jiz
   existuje!
```

7.10 Maily

Test ověřující úspěšné vytvoření nového mailového účtu a reakci na neočekávanou chybu v podobě existence uživatele se stejným jménem.

7.10.1 Ověření funkčnosti

Vstupní data

```
Typ: Uzivatel
Uzivatel: uzivatel_email
```

Výsledek testu

```
[1/1] - Testuji MAIL s argumenty: -a uzivatel_email: [OK]
```

Zároveň můžeme ověřit přítomnost systémového uživatele příkazem:

```
$ id -u uzivatel_email
59984
```

7.10.2 Zareagování na chybu

Před spuštěním testu ručně vytvoříme uživatele `uzivatel_email_chyba`.

Vstupní data

```
Typ: Uzivatel
Uzivatel: uzivatel_email_chyba
```

Výsledek požadavku

```
[1/1] - Testuji MAIL s argumenty: -a uzivatel_email_chyba:
   [CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
Splnenych: 0
Chybnych: 1
======================================================
Test MAIL CHYBA
Chyba - uzivatel uzivatel_email_chyba jiz existuje!
```

7.11 OpenAFS svazky

Test ověřující úspěšné vytvoření nového OpenAFS svazku a reakci na neočekávanou chybu v podobě existence svazku se stejným názvem.

7.11.1 Ověření funkčnosti

Vstupní data

```
Typ: vyuka
Nazev svazku: openafs_svazek
Velikost svazku: 10000K
Uzivatele: feelus
```

Výsledek testu

```
[1/1] - Testuji AFS s argumenty: -p vyuka -m openafs_svazek -q
   10000 -a feelus: [OK]
```

Zareagování na chybu

Před spuštěním testu ručně vytvoříme svazek `openafs_svazek_chyba`.

Vstupní data

```
Typ: vyuka
Nazev svazku: openafs_svazek_chyba
Velikost svazku: 10000K
Uzivatele: feelus
```

Výsledek testu

```
[1/1] - Testuji AFS s argumenty: -p vyuka -m
   openafs_svazek_chyba -q 10000 -a feelus: [CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
```

```
Splnenych: 0
Chybnych: 1
=====================================================
Test OPENAFS CHYBA
Chyba - svazek s nazvem openafs_svazek_chyba jiz existuje
```

7.12 SVN repozitáře

Test ověřující úspěšné vytvoření nového SVN repozitáře a reakci na neočekávanou chybu v podobě existence repozitáře v cílové složce.

7.12.1 Ověření funkčnosti

Vstupní data

```
Umisteni: 147.228.67.127
Nazev repozitare: nazev_svn_repo
Uzivatele: feelus
```

Výsledek testu

```
[1/1] - Testuji SVN s argumenty: -d 147.228.67.127 -m
   nazev_svn_repo -a feelus: [OK]
```

Zároveň můžeme ručně ověřit vytvoření adresáře `/home/svn/nazev_svn_repo` příkazem:

```
$ file /home/svn/nazev_svn_repo
/home/svn/nazev_svn_repo/: directory
```

7.12.2 Zareagování na chybu

Před spuštěním testu ručně vytvoříme repozitář `nazev_svn_repo_chyba`.

Vstupní data

```
Umisteni: 147.228.67.127
Nazev repozitare: nazev_svn_repo_chyba
Uzivatele: feelus
```

Výsledek požadavku

```
[1/1] - Testuji SVN s argumenty: -d 147.228.67.127 -m
   nazev_svn_repo_chyba -a feelus: [CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
Splnenych: 0
Chybnych: 1
======================================================
Test SVN CHYBA
Chyba - repozitar v ceste /home/svn/nazev_svn_repo_chyba jiz
   existuje
```

7.13 GIT repozitáře

Test ověřující úspěšné vytvoření nového GIT repozitáře a reakci na neočekávanou chybu v podobě existence repozitáře v cílové složce.

7.13.1 Ověření funkčnosti

Vstupní data

```
Umisteni: 147.228.67.127
Nazev repozitare: nazev_git_repo
Uzivatele: feelus
SSH klic: ssh-rsa AAAAB3NzaC1yc2EAAAAD...
```

Výsledek testu

```
[1/1] - Testuji GIT s argumenty: -d 147.228.67.127 -k ssh-rsa
   AAAAB3NzaC1yc2EAAAAD -m nazev_git_repo -a feelus: [OK]
```

Zároveň můžeme ověřit vytvoření adresáře `/home/svn/nazev_git_repo` příkazem:

```
$ file /home/git/nazev_git_repo
/home/git/nazev_git_repo/: directory
```

Zareagování na chybu

Před spuštěním testu ručně vytvoříme repozitář `nazev_git_repo_chyba`.

Vstupní data

```
Umisteni: 147.228.67.127
Nazev repozitare: nazev_git_repo_chyba
Uzivatele: feelus
SSH klic: ssh-rsa AAAAB3NzaC1yc2EAAAAD...
```

Výsledek testu

```
[1/1] - Testuji GIT s argumenty: -d 147.228.67.127 -k ssh-rsa
    AAAAB3NzaC1yc2EAAAAD -m nazev_git_repo_chyba -a feelus:
    [CHYBA]
---------------------- VYSLEDEK ----------------------
Spusteno: 1
Splnenych: 0
Chybnych: 1
======================================================
Test GIT CHYBA
Chyba - repozitar s nazvem nazev_git_repo_chyba jiz existuje
```

7.14 Výsledek testů

Pomocí testů jsme ověřili funkčnost požadavků všech typů. Zároveň jsme otestovali odhalení chyb, které včas zamezí problémům jako jsou výskyty duplicit nebo nesrovnalosti při vykonávání úkolu.

8 Závěr

Cílem této práce bylo nastudování potřeb organizace KIV pro jednotnou správu zdrojů, nalezení vhodného řešení či navržení a implementování vlastního. Detailně jsme nastudovali jednotlivé typy zdrojů a jejich konkrétní využití v rámci organizace.

Následně jsme prostudovali a otestovali existující řešení, která se podle předpokladů jevila jako nejvhodnější kandidáti pro jednotnou správu zdrojů KIV. Během testování jsme došli k závěru, že žádný z kandidátů nevyhovuje kompletně celému zadání. Hlavním důvodem je heterogenita prostředí organizace, ne příliš častá řešení použití některých zdrojů a propojení funkčnosti s existující infrastrukturou organizace. Z otestovaných programů jsme se inspirovali některými vhodnými vlastnostmi, které jsme následně zvážili při návrhu a implementaci vlastního řešení.

Ve fázi návrhu vlastního řešení jsme brali v potaz použité softwarové vybavení organizace, aby řešení bylo možné v celém systému jednoduše provozovat a udržovat. Následně jsem navržené řešení programově implementoval a otestoval v testovacím prostředí KIV. Vysoký důraz byl kladen na bezpečnost, jelikož navrhovaný program bude vlastnit přístupová oprávnění do klientských systémů.

Zadání práce bylo úspěšně splněno a implementovaný systém vyhovuje všem požadavkům zadání. Systém zároveň umožňuje snadné rozšíření o další typy zdrojů díky modulárnímu návrhu. Správa všech typů zdrojů byla důkladně otestována, spolu s chováním systému v případě výskytu vynucené chyby.

Vhodným rozšířením práce by bylo vytvoření automatických instalátorů jak pro serverovou tak klientskou část, která by umožňila výběr podporovaných a následnou automatickou konfiguraci systému, s žádnou či minimální nutností zásahu uživatele.

Použité zkratky

- **KIV**, `Katedra informatiky a výpočetní techniky Západočeské univerzity v Plzni`
- **CIV**, `Centrum informatizace a výpočetní techniky Západočeské univerzity v Plzni`
- **SQL**, `Structured Query Language`, standardizovaný strukturovaný jazyk navržený pro práci s daty v relačních databázích
- **SSH**, `Secure Shell`, zabezpečený síťový protokol pro vzdálený přístup
- **LVM**, `Logical Volume Manager`, sloužící pro flexibilní alokování prostoru na úložném zařízení
- **DRBD**, `Distributed Replicated Block Device`, distribuovaný replikovaný úložný systém
- **HTTP**, `Hypertext Transfer Protocol`, internetový protokol převážně využíváný pro výměnu hypertextových dokumentů
- **HTTPS**, `Hypertext Transfer Protocol Secure`, zabezpečená verze protokolu HTTP
- **API**, `Application Programming interface`, programátorské rozhraní pro přístup k funkčnosti aplikace

Přílohy

K práci je přiloženo CD s následujícím obsahem:

- **src** - zdrojové kódy
  - **bpmk-web** - zdrojové soubory webu
  - **bpmk-daemon** - zdrojové kódy démona
  - **bpmk-daemon-control-client** - klientské skripty
  - **bpmk-daemon-control-srv** - serverové skripty
  - **bpmk-tests** - testovací skripty
- **doc** - PDF verze dokumentace a zdrojové soubory TEX

Literatura

[1] Thomas Uphill: **Mastering Puppet**
Packt Publishing
2014

[2] Lorin Hochstein: **Ansible: Up and Running**
O'Reilly Media
2015

[3] Colton Myers: **Learning Saltstack**
Packt Publishing
2015

[4] Santos Martinez, Peter Daalmans, Brett Bennett: **Mastering System Center 2012 R2 Configuration Manager**
Sybex
2014

[5] Rich Bowen, Ken Coar: **Apache Cookbook: Solutions and Examples for Apache Administrators**
O'Reilly Media
2008

[6] Jason Helmick: **Learn Windows IIS in a Month of Lunches**
Manning Publications
2014

[7] Jason Brittain: **Tomcat: The Definitive Guide**
O'Reilly Media
Listopad 2007

[8] https://eclipse.org/jetty/documentation/current: **Jetty : The Definitive Reference**

https://eclipse.org/jetty/documentation/current
online květen 2015

[9] Paul DuBois: **MySQL Cookbook: Solutions for Database Developers and Administrators**
O'Reilly Media
2014

[10] Richard Stones, Neil Matthew: **Beginning Databases with PostgreSQL: From Novice to Professional**
Apress
2007

[11] Rick Greenwald, Robert Stackowiak, Jonathan Stern: **Oracle Essentials: Oracle Database 12c**
O'Reilly Media
2013

[12] Ross Mistry, Stacia Misner: **Introducing Microsoft SQL Server 2014**
Microsoft Press
2014

[13] Jeanna N. Matthews, Eli M. Dow, Todd Deshane, Wenjin Hu, Jeremy Bongio, Patrick F. Wilbur, Brendan Johnson: **Running Xen: A Hands-On Guide to the Art of Virtualization**
Prentice Hall
2008

[14] http://tldp.org/HOWTO/LVM-HOWTO: **LVM HOWTO**
http://tldp.org/HOWTO/LVM-HOWTO
online květen 2015

[15] http://drbd.linbit.com/docs/about: **DRBD:About**
http://drbd.linbit.com/docs/about
online květen 2015

[16] John L. Hufferd: **iSCSI: The Universal Storage Connection: The Universal Storage Connection**
Addison-Wesley Professional
2002

[17] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/DM_Multipath: **DM**

Multipath Configuration and Administration
`https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/DM_Multipath`
online květen 2015

[18] https://www.gnu.org/software/mailman/docs.html: **Mailman Documentation**
`https://www.gnu.org/software/mailman/docs.html`
online květen 2015

[19] Franco Milicchio, Wolfgang Alexander Gehrke: **Distributed Services with OpenAFS: for Enterprise and Education**
Springer
2007

[20] Jamie Cameron: **Managing Linux Systems with Webmin: System Administration and Module Development (Bruce Perens' Open Source)**
Prentice Hall
2003

[21] Aric Pedersen: **cPanel User Guide and Tutorial: Get the most from cPanel with this easy to follow guide**
Packt Publishing
2006

[22] John Rhoton, Jan De Clercq, Franz Novak: **OpenStack Cloud Computing: Architecture Guide**
Recursive Press
2014

[23] http://apachegui.net: **ApacheGUI**
`http://apachegui.net`
online únor 2015

[24] http://svn-access-mana.sourceforge.net: **SVN Access Manager Documentation**
`http://svn-access-mana.sourceforge.net`
online únor 2015

[25] Marc Delisle: **Mastering phpMyAdmin 3.4 for Effective MySQL Management**
Packt Publishing
2012

[26] http://www.pgadmin.org/docs: **pgAdmin: Documentation**
`http://www.pgadmin.org/docs`
online květen 2015

[27] http://docs.oracle.com/cd/E12151_01: **SQL Developer Documentation**
`http://docs.oracle.com/cd/E12151_01`
online květen 2015

[28] http://www.sqlmanager.net/en/products/mssql/manager/documentation: **EMS SQL Manager for SQL Server**
`http://www.sqlmanager.net/en/products/mssql/manager/documentation`
online květen 2015

[29] http://sourceforge.net/p/postfixadmin/wiki/Home: **Postfix Admin / Wiki / Home**
`http://sourceforge.net/p/postfixadmin/wiki/Home`
online květen 2015

[30] http://xenserver.org/partners/developing-products-for-xenserver/21-xencenter-development/88-xc-dev-home.html: **XenCenter Development**
`http://xenserver.org/partners/developing-products-for-xenserver/21-xencenter-development/88-xc-dev-home.html`
online květen 2015

[31] http://w3techs.com/technologies/history_overview/programming_language: **Historical trends in the usage of server-side programming languages**
`http://w3techs.com/technologies/history_overview/programming_language`
online květen 2015

[32] Riwanto Megosinarso: **Step By Step Bootstrap 3: A Quick Guide to Responsive Web Development Using Bootstrap 3**
CreateSpace Independent Publishing Platform
2014

[33] http://www.unixodbc.org/doc: **unixODBC**
`http://www.unixodbc.org/doc`
online květen 2015

[34] http://www.freetds.org/docs.html: **Documentation**
`http://www.freetds.org/docs.html`
online květen 2015