

Webový nástroj pro automatické stahování, kontrolu a nastavení konfigurace CISCO zařízení

Web-based Tool for Checking and Configuration of CISCO
devices

Bc. Jiří Leskovec

Diplomová práce
2013



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2012/2013

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Jiří LESKOVEC**
Osobní číslo: **A10384**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **kombinovaná**

Téma práce: **Webový nástroj pro automatické stahování, kontrolu
a nastavení konfigurace CISCO zařízení**

Zásady pro vypracování:

1. Zpracujte literární rešerši na dané téma.
2. Popište nedostatky současných řešení.
3. Navrhněte architekturu webové aplikace sloužící pro konfiguraci a kontrolu zařízení.
4. Vytvořte webovou aplikaci.
5. Věnujte pozornost zabezpečení webové aplikace.
6. Vyhodnoťte přínos řešení a uveďte možnosti dalšího rozvoje a uplatnění vytvořené aplikace.

Rozsah diplomové práce:
Rozsah příloh:
Forma zpracování diplomové práce: **tištěná/elektronická**

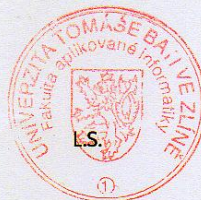
Seznam odborné literatury:

1. KOMUNITA JQUERY. JQuery: kuchařka programátora. Vyd. 1. Brno: Computer Press, 2010, 436 s. ISBN 978-80-251-3152-7.
2. KOFLER, Michael a Bernd ÖGGL. PHP 5 a MySQL 5: průvodce webového programátora. Vyd. 1. Brno: Computer Press, 2007, 607 s. ISBN 978-80-251-1813-9.
3. WELLMAN, Dan a Bernd ÖGGL. JQuery UI 1.8: the user interface library for jQuery : build highly interactive web applications with ready-to-use widgets from the jQuery user interface library. Vyd. 1. Birmingham [U.K.]: Packt Pub., 2011, 401 p. ISBN 978-1849516525.
4. MEDUNA, Alexander a Zbyněk KRÍVKA. Přednášky předmětu IFJ [online]. 2012 [cit. 2013-02-04]. Dostupné z: <http://www.fit.vutbr.cz/study/courses/IFJ/public/materials/>
5. The PHP Group. PHP Manual [online]. 2013 [cit. 2013-02-04]. Dostupné z: <http://www.php.net/manual/en/index.php>
6. JQUERY FOUNDATION. JQuery UI API Documentation [online]. 2013 [cit. 2013-02-04]. Dostupné z: <http://api.jqueryui.com/>
7. JQUERY FOUNDATION. Plugins - jQuery Wiki: livequery [online]. 2010 [cit. 2013-02-04]. Dostupné z: <http://docs.jquery.com/Plugins>
8. ZAKAS, Nicholas C. JavaScript pro webové vývojáře: programujeme profesionálně. Vyd. 1. Brno: Computer Press, 2009, 832 s. ISBN 978-80-251-2509-0.

Vedoucí diplomové práce: **Ing. Miroslav Matýsek, Ph.D.**
Ústav počítačových a komunikačních systémů
Datum zadání diplomové práce: **22. února 2013**
Termín odevzdání diplomové práce: **22. května 2013**

Ve Zlíně dne 22. února 2013

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této práce bylo zhodnotit současné možnosti automatické kontroly a konfigurace CISCO zařízení a vytvořit open-source aplikaci, která umožňuje operace nad CISCO zařízeními. Nejprve je zmíněna aktuální situace na trhu, popis současných komerčních řešení a dále pak SW návrh vlastní aplikace. Dále se text zaměřuje na popis řešení a implementace každého bloku aplikace, chování aplikace, technické možnosti aplikace a hranice těchto možností. Závěrem je pak vyhodnocen přínos, možnost uplatnění a zejména pak další rozšíření celého systému.

Klíčová slova:

CISCO, zařízení, kontrola, konfigurace, SSH, telnet, webová aplikace

ABSTRACT

The objective of this thesis was to evaluate current configuration and check options for CISCO devices and create open-source application that provides subset of operations on CISCO devices. At first is mentioned the actual situation on market, then description of actual commercial solutions and finally design of the application. Next chapters of this thesis are focused on the description and implementation of each block the application, application behavior, technical aspects and limitations of these aspects. In the end of the thesis is evaluation of benefits, possibilities of usage and mainly future extensions of entire system.

Keywords:

CISCO, device, check, configuration, SSH, telnet, web application

Děkuji tímto panu Ing. Miroslavu Matýskovi, Ph.D. za odborné vedení, rady a konzultace při psaní této práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

ÚVOD	12
I. TEORETICKÁ ČÁST	13
1 SOUČASNÉ MOŽNOSTI AUTOMATICKÉHO MANAGEMENTU A KONTROLY	14
1.1 UNIXOVÉ SKRIPTY	14
1.2 HP AUTOMATED NETWORK MANAGEMENT SOFTWARE	14
1.3 NESSUS	15
1.4 CISCO WORKS	15
1.5 SOLARWINDS	16
1.6 TITANIA	17
2 NÁVRH APLIKACE	18
2.1 POŽADAVKY A VÝBĚR ŘEŠENÍ	18
2.1.1 Použité řešení	20
2.2 SOUČÁSTI APLIKACE	21
2.3 ČÁSTI APLIKACE A UML NÁVRHY	22
2.3.1 GUI.....	22
2.3.2 SPUŠTĚNÍ KONTROLNÍ A KONFIGURAČNÍ ŠABLONY	22
2.3.3 SPRÁVA ZEMÍ, LOKACÍ A ZÁKAZNÍKŮ	23
2.3.4 SPRÁVA ŠABLON, PŘÍKAZŮ A VÝSTUPNÍCH ROZLOŽENÍ (LAYOUTŮ).....	24
2.3.5 SPRÁVA UŽIVATELŮ	26
2.3.6 NASTAVENÍ APLIKACE.....	26
2.3.7 PŘIPOJENÍ K ZAŘÍZENÍ	26
2.3.8 KONTROLNÍ ČÁST	27
2.3.9 DETERMINISTICKÝ KONEČNÝ AUTOMAT (DKA)	27
2.4 MYSQL A NÁVRH DATABÁZE	28
2.4.1 MYISAM	30
2.4.2 INNODB	30
2.5 JQUERY	31
II. PRAKTICKÁ ČÁST	33
3 NASTAVENÍ A SPUŠTĚNÍ PROSTŘEDÍ	34
3.1 MYSQL	35
3.2 APACHE A PHP	36
4 IMPLEMENTACE APLIKACE	38
4.1 ADRESÁŘOVÁ STRUKTURA A POPIS SOUBORŮ	38
4.2 APLIKAČNÍ TŘÍDY A POPIS JEJICH FUNKCÍ	40
4.2.1 TŘÍDA PRO OPERACE S DATABÁZÍ.....	40
4.2.2 TŘÍDA PRO ŘÍZENÍ CHYB	41
4.2.3 TŘÍDA PRO VYTVÁŘENÍ HTML ELEMENTŮ.....	42
4.2.4 TŘÍDY PRO PŘIPOJENÍ K ZAŘÍZENÍ.....	42
4.3 SYSTÉM PŘEKLADŮ	45
4.4 SPRÁVA A UŽIVATELŮ A SYSTÉMOVÉ ZABEZPEČENÍ	46

4.4.1	SYSTÉMOVÁ OPRÁVNĚNÍ A JEJICH POPIS	46
4.4.2	ZABEZPEČENÍ APLIKACE.....	48
4.5	SPRÁVA ZAŘÍZENÍ	49
4.6	SPRÁVA ZEMÍ, LOKACÍ A ZÁKAZNÍKŮ	50
4.7	SPRÁVA ŠABLON A PŘÍKAZŮ	50
4.7.1	VYUŽITÍ JQUERY UI.....	52
4.7.2	SPRÁVA PŘÍKAZŮ	53
4.8	SPRÁVA LAYOUTŮ A VÝSTUP SKRIPTU	54
4.8.1	KONTROLNÍ BLOKY	56
4.8.2	TYPY BLOKŮ A JEJICH POPIS	57
4.8.3	SEKČNÍ PŘÍKAZ A IMPLEMENTACE DKA.....	58
4.8.4	SEKČNÍ DKA.....	59
4.8.5	PŘÍKLAD DKA SKRIPTU.	63
4.9	SPUŠTĚNÍ KONTROLY A KONFIGURACE ZAŘÍZENÍ.....	63
4.10	NASTAVENÍ APLIKACE	64
4.11	APLIKAČNÍ PROBLÉMY A BUDOUCÍ ROZŠÍŘENÍ APLIKACE.....	64
	ZÁVĚR	66
	CONCLUSION	67
	SEZNAM POUŽITÉ LITERATURY.....	68
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	70
	SEZNAM OBRÁZKŮ	71
	SEZNAM TABULEK.....	72
	SEZNAM PŘÍLOH.....	73

ÚVOD

V každé společnosti zabývající se správou síťových technologií je vždy třeba v určitém časovém intervalu zkontrolovat integritu a nastavení všech zařízení připojených do počítačové sítě, zda splňují bezpečnostní politiku dané společnosti. Ať už je to z důvodu přestavby sítě, přidání nových prvků nebo kvůli bezpečnostním kontrolám (auditům). V současnosti existuje několik nástrojů, které to umožňují lépe či hůře. Nejčastěji se však jedná o nástroje pro pouhý monitoring sítě, nebo zálohu konfigurací vybraných zařízení. Vrcholem funkčnosti je pak jen porovnání konfigurace kontrolovaného zařízení s nějakým jiným (správně nastaveným) zařízením. Ve všech těchto případech je nutný ruční – lidský zásah pro danou činnost. To je však krajně nepraktické, protože je vždy třeba člověk obeznámený s interní firemní bezpečnostní politikou a také – a to je hlavní – lidský faktor může zapříčinit chybu. V tomto případě se pak jedná o zcela zbytečnou práci. Pokud se opravdu jedná o nástroj, který je schopný nějaké pokročilejší analýzy konfiguračního souboru, je velmi drahý a pro běžné použití příliš komplexní.

Pozorováním situace a požadavků u několika zaměstnavatelů bylo zjištěno, že kontroly konfigurace jsou prováděny poměrně často, avšak většinou nekvalitně, či neúplně. Chybějící technické či lidské prostředky mohou vést ke spoustě situací, které mohou libovolnou společnost stát nemalé problémy, ale i finanční prostředky. Aplikace, která by toto umožňovala dostatečně kvalitně a zároveň levně se na trhu v podstatě nevyskytuje. Cílem této práce bylo tedy takovouto aplikaci vytvořit.

Z povahy práce v podstatě neexistují o tomto tématu žádné knihy. Veškeré použité materiály jsou získány v elektronické formě. Zpravidla se jedná o webové stránky produktů, reference k programovacím jazykům a několik portálů s tematikou programování a sítí.

I. TEORETICKÁ ČÁST

1 SOUČASNÉ MOŽNOSTI AUTOMATICKÉHO MANAGEMENTU A KONTROLY

1.1 Unixové skripty

V praxi nejvíce využívaným způsobem hromadné správy zařízení pod systémem Unix jsou nástroje z rodiny RANCID (Really Awesome New Cisco confIg Differ) [1], zejména potom skript clogin. Tyto nástroje lze snadno nainstalovat pomocí balíčkových manažerů přímo v systému Unix a jejich spuštění je tedy poměrně jednoduché. Pomocí těchto nástrojů je možné se přihlásit k libovolnému zařízení (popisovaný je však určen zejména pro CISCO zařízení) a stáhnout, eventuálně nahrát na něj libovolnou konfiguraci. Vždy je však nutné vytvořit soubor s příkazy, které chceme použít a rovněž i seznam zařízení. Je možné jej pomocí dalších nadstavbových skriptů rozšířit tak, aby se tyto soubory příkládaly samy při každém použití. Problémem tohoto řešení zůstává to, že veškerý výstup musí být ručně zpracován, nebo jej musí zpracovat odlišná sada skriptů. Tento přístup v zásadě znemožňuje nějaké jednoduché modifikace hledaných informací a vždy vyžaduje zásah uživatelů do skriptů. Výhodou ovšem je, že je zcela zdarma, není náročný na zdroje a pokud technická komplikovanost není překážkou, pak se jedná o dobré řešení.

1.2 HP Automated Network Management Software

Tento software od společnosti Hewlett-Packard obsahuje celou škálu nástrojů pro komplexní správu sítě. Jedná se o aplikaci, kterou lze nainstalovat jak na Windows, tak na Unixové systémy. Nevyžaduje tedy ke svému běhu aplikace třetích stran (Javu). Zde jsou vyjmenovány jeho klíčové vlastnosti, které jsou zmíněny v oficiální dokumentaci [2]

- Jediný nástroj pro kompletní řízení síťové infrastruktury.
- Vylepšená dostupnost sítě s multiuživatelským řešením správy sítě.
- Společný pohled a kontext pro veškeré bezpečnostní a síťové problémy.
- Zvýšená produktivita a efektivita operátorů, redukováné MTTR (Mean Time To Restore).
- Umožňuje správu více zákazníků, oddělení nebo míst při nejnižší ceně.

Na stránkách výrobce bohužel chybí detailnější informace o produktu, včetně systémových nároků a ceny. Pokud potřebujeme jen kontrolu zařízení, nebo konfiguraci na zařízeních, není tento nástroj z důvodu své komplexnosti příliš vhodný. Nicméně pro nasazení ve

větším prostředí, kde je třeba i monitorování sítě a další aplikace, je zcela jistě možné o něm uvažovat.

1.3 NESSUS

Rodina nástrojů Nessus od firmy Tenable je primárně zaměřena pouze na kontrolu zařízení od většiny firem na trhu a podporuje všechny typy zařízení. Speciálně se potom zaměřuje na hledání zranitelných částí konfigurace na zařízeních, tzv. Vulnerabilit¹. Klíčové vlastnosti popsané na stránkách produktu [3]:

- nejpoužívanější nástroj pro zjišťování slabín používaný více než 15000 zákazníky po celém světě.
- Provádí auditování konfigurace, záplat a webových aplikací.
- Testuje více než 50 tisíc vulnerabilit a umožňuje kontrolu konfigurace za pomoci přídatných částí kódu, tzv. pluginů².
- Skenuje sítě, systémy, data a aplikace.

Jak bylo zmíněno výše, primární účel je hledání slabín na širokém spektru systémů. Pokud by bylo třeba i něco nakonfigurovat, je nutné použít jiný nástroj, nebo provést změny ručně. Nessus však obsahuje i plugin, který se specializuje na kontrolu konfigurace CISCO zařízení se systémem IOS (Internetwork operating systém). Jedná se o Cisco IOS Compliance Checks s identifikátorem 46689. Za pomoci něj je možné rozšířit funkcionalitu nástroje o přímé kontrolování konfigurace oproti požadovaným politikám. Výhodou celého systému je použitelnost. Stačí 1 nástroj pro hledání a identifikaci problémů v síti. Nevýhodou pro menší firmy pak může být cena, která se pohybuje kolem 1500\$ za rok.

1.4 CISCO WORKS

Společnost CISCO nezůstává samozřejmě stranou a nabízí svoje řešení pro správu svých zařízení. Toto řešení se nazývá CiscoWorks LAN Management Solution v současné verzi 4.0. Jde o celou řadu klient-server aplikací, které jsou dostupné pouze pro Windows a

¹ Česky „slabina“ - programátorská chyba, která způsobuje v softwaru zneužitelný bezpečnostní problém

² Česky „zásuvný modul“, který rozšiřuje funkcionalitu programu

Solaris a to jak 32bitové, tak 64 bitové. Dle dokumentace je i výborně podporována virtualizace. Hlavní oblasti systému jsou následující [4]:

- Monitorování a hledání problémů - rychle a proaktivně detekuje a řeší problémy v síti.
- Konfigurační management - zálohy konfigurace, zálohy systému.
- Inventář - informace o detailech jednotlivých zařízení (chassis, moduly,...).
- Reporty - Centralizace reportů, EOL (End of Life).
- Administrace - jednoduché nastavení a administrace.

Jeho používání se přímo nabízí, pokud je síť sestavena převážně z CISCO komponent. Zejména doplňující aplikace, jako inventář, je ideální k uchování informací, které jsou nutné z pohledu uživatele i výrobce. Jedná se hlavně o sériová čísla modulů, přídatných karet, celých zařízení, napájecích zdrojů a mnoha dalších. Pokud se však dále podíváme do specifikací a požadavků na systémové prostředky, dělají tyto přídatné aplikace a nástroje z celého systému poměrně těžkopádné řešení, která není příliš vhodná pro jednoduchou správu a kontrolu. I cena bude pro malou společnost pravděpodobně faktorem, který ji od tohoto řešení odradí. Jedná se totiž o téměř 10 000\$.

1.5 SOLARWINDS

Další v řadě nástrojů je Network Configuration Manager od společnosti Solarwinds. Tato společnost je velmi známá svým portfoliem síťových aplikací a výše jmenovaná aplikace je jen jedna z několika, tvořící ucelenou skupinu aplikací pro správu, dohled a kontrolu sítě. Aplikace existuje bohužel jen pro Windows. Podporuje širokou škálu protokolů pro připojení, správu, detekci zařízení a pro zálohu či upgrade celých zařízení. Vlastností je celá řada, a proto jsou zde zmíněny jen relevantní pro tento přehled [5]:

- Automatické zálohování konfigurace.
- Real-time detekce změn na zařízeních.
- Automatická detekce zařízení v síti.
- Porovnání konfigurací a návrat zpět k původní.

Tato aplikace plní dobře účel, ke kterému byla vytvořena. Bohužel jí chybí právě možnost porovnání konfigurace s interní nastavenou firemní politikou a jednoduchý přehled chyb.

Umožňuje pouze porovnání rozdílů konfigurací návrat k předchozí konfiguraci. Cena je rovněž vyšší a dle výrobce začíná u 2500€.

1.6 TITANIA

Společnost Titania vyvinula nástroj jménem Nipper. Jedná se o čistě kontrolní nástroj, který se skládá ze 2 podprogramů: Nipper One a Nipper. Nipper One poskytuje jednoduchý způsob přístupu k funkcionalitám Nipperu – GUI (Graphical user interface). Nipper je příkazový řádek (CLI – Command Line Interface) pro uživatele, kteří znají lépe vlastní prostředí a navíc umožňuje využívat Nipper používán automatickými skripty. Je dostupný pro Windows, Unix i Macintosh. Klíčové vlastnosti jsou následující [6]:

- Obecné nastavení zařízení (jméno, verze systému,...).
- Síťové služby - přehled aktivních síťových služeb.
- Administrační nastavení - aktivní Telnet, SSH,...
- Autentizační nastavení - způsob přístupu k zařízení (TACACS - Terminal Access Controller Access-Control System, lokální účty).
- Porty, routovací informace, nastavení SNMP, času a dalších.

Nipper je vhodný spíše pro administrátory, než pro koncové uživatele. Ti sice mohou vybrat v GUI, které položky mají být kontrolovány, avšak s použitím dalších skriptů je jeho využitelnost daleko větší. Obsahuje též pouze pevnou množinu hodnot, které je schopný zkontrolovat. Ta je sice poměrně široká a obsahuje dost možností pro kontrolu, avšak časem se může stát, že bude potřeba ověřit nastavení, které nástroj neumožňuje zkontrolovat. Cenu se nepodařilo zjistit.

2 NÁVRH APLIKACE

Ani jedno z výše uvedených řešení není vhodné pro jednoduchou správu a kontrolu zařízení. Vždy se jedná buď o velmi drahou aplikaci, nebo nesplňuje účel, pro který je třeba. Jednoduchá a přitom účelná kontrola zařízení ve firemním prostředí byla velmi obtížná. Situace byla řešena nejdříve jednoduchými Unixovými skripty. V 1.1 je popsáno jedno z možných použití skriptu Clogin. Pomocí něj bylo možné stahovat konfigurace ze zařízení a poté z nich byly jinými, velmi jednoduchými skripty získávány požadované informace. Velmi brzy však bylo dosaženo omezení a to takových, že pro každý typ zařízení je třeba mít zvlášť stahovací skript a poté kontrolní skript. Vystal rovněž problém s přenositelností. Na většině serverů má uživatel jen práva pro čtení a při pokusu nastavit souboru práva pro spuštění narazí uživatel často na problém.

Další etapou byl nástroj psaný v jazyce PHP (Hypertext Preprocessor). Jednalo se o poměrně jednoduchou aplikaci, která měla i GUI. To však nebylo příliš přívětivé. Navíc veškerá kontrolní činnost byla zabudována přímo v kódu a tak jakákoliv změna požadavků měla za následek jejich úpravu. Aplikace se rovněž nebyla schopná připojit k zařízení, a stáhnout sama konfiguraci. To bylo třeba provést buď ručně, nebo za pomoci výše zmíněných skriptů a poté do aplikace importovat. Celý systém byl tedy poněkud těžkopádný a náročný na další údržbu.

Poslední verze se tedy musela vyvarovat všech těchto neduhů. Proto jsem začal úplně znovu a od začátku, aby se do programu nezanášely staré chyby a byla zajištěna veškerá požadovaná funkcionalita.

2.1 Požadavky a výběr řešení

Cílem tedy bylo vytvořit aplikaci s GUI, pomocí které by bylo možné provádět kontroly zařízení na základě volitelných parametrů, jako např. verze systému, zákazníka, typu zařízení apod. Hlavním požadavkem byla jednoduchá přenositelnost, tzn. umožnit nasadit aplikaci v podstatě kdekoliv, bez komplikované instalace prostředí, rekompilací apod. Z celého spektra jazyků jsem nakonec zúžil výběr na jazyky C a PHP. Oba jazyky poskytují potřebnou funkcionalitu. Jazyk C sám o sobě neobsahuje potřebné knihovny a funkce, které by zabezpečily hladkou funkčnost na různých prostředích. Jedná se hlavně o GUI, SSH a Telnet spojení a MySQL knihovny. Všechny je možné získat za použití knihoven třetích stran.

wxWidgets knihovna je určena pro programy s grafickým prostředím a umožňuje přenositelnost mezi Windows a Unixem. Je bohužel nutné mít na obou systémech kompilátory s příloženými knihovnami a při přenosu celý program překompilovat. To není příliš praktické, a i když by bylo možné napsat skripty nebo batch³soubory pro okamžitou kompilaci, není to ovšem příliš komfortní řešení.

Připojení pomocí SSH by se dalo realizovat pomocí libssh [7] knihovny. Ta je současně ve verzi 0.5.4. Knihovna obsahuje implementaci klienta i serveru. Pro naše účely by však stačila jen klientská část. Knihovna je poměrně bohatá na funkce. Obsahuje samozřejmě funkce pro připojení k zařízení přes SSH, správu chyb a sezení, ale také i funkce pro práci s řetězci, buffery⁴, vlákna, správu klíčů a přenos přes SCP (Secure Copy). Knihovna funkcí pro telnet existuje ve formě Open source na je dostupná ne webu [8]. Je potřeba přidat i nějaké další hlavičkové soubory, to je však již nad rámec této práce.

MySQL databáze byla zvolena proto, že obsahuje širokou podporu pro většinu používaných jazyků a disponuje tedy i vlastním API pro jazyk C. Jedná se o velmi rozsáhlou knihovnu, která umožňuje kompletní správu databází z klienta psaného v jazyce C. Obsahuje předdefinované struktury a funkce pro jednoduchou manipulaci s daty. Stačí pouze přiložit hlavičkové soubory a změnit některá nastavení (jako např. velikost komunikačního bufferu).

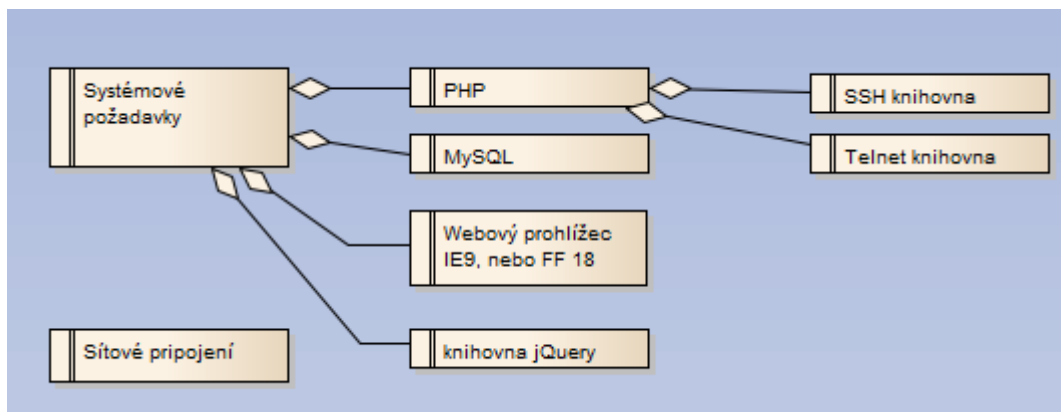
Jak je vidět, pro vyvinutí celého nástroje by byla třeba celá řada podpůrných prostředků a knihoven. Tím by se poměrně zkomplikovala udržitelnost celého systému, protože by se musely hlídat aktualizace a bug fixy⁵, nová vydání a vylepšení pro každou jednotlivou knihovnu. Z těchto důvodů byla zvolena jako cílová platformu webová aplikace psaná v PHP verzi 5 s využitím MySQL databáze verze 5. Protože se jedná o webovou aplikaci, bude za pomoci PHP vytvářen HTML (Hypertext Markup Language) grafické prostředí generováno webovým prohlížečem. Na klientské straně se pro větší funkcionalitu využívat i javascriptová knihovna jQuery a její nadstavba jQuery UI. Ta obsahuje funkce jak pro operaci s objekty tak funkce pro zpříjemnění práce s webovou

³ Soubor systémových příkazů obsahující i některé řídicí bloky jako IF, GOTO, apod.

⁴ Vyrovnávací paměť, slouží k meziukládání výsledků, využívá se typicky při síťové komunikaci

⁵ Bugfix – oprava známého problému

stránkou. Zejména pak drag&drop⁶ funkce, funkce pro třídění bloků, dialogy, AJAX (Asynchronous JavaScript and XML) funkce a spoustu dalších prvků, včetně různých grafických efektů.



Obr. 1. Nefunkcionální požadavky

Protože celá tato skladba programů by při separátní instalaci trpěla stejnými neduhy jako řešení v C, bylo zvoleno kompletní řešení v jediném balíku. Tím je xampplite [9].

2.1.1 Použité řešení

Tato aplikace existuje jak pro Windows, tak pro Unix. Stačí tedy nainstalovat jediný balík, a pokud zachováme stejnou verzi balíku na Windows i Unix, bude veškerá funkčnost zajištěna na obou systémech. Pro vývoj této práce bylo použito prostředí XAMPP pro Windows ve verzi 1.7.3, které obsahuje následující aplikace:

- Apache 2.2.14.
- PHP 5.3.1.
- MySQL 5.1.41.

Pro webové rozhraní byla použita knihovna jQuery ve verzi 1.7.2 a jQuery UI 1.9.1. Součástí balíku je i webová aplikace pro správu MySQL databází – phpmyadmin. Pomocí ní je možné přímo přistupovat k databázím a tabulkám v MySQL.

PHP bohužel neobsahuje v originální kompilaci podporu pro SSH, ani telnet. V případě SSH existuje knihovna funkcí, kterou je možné jako PECL (PHP Extension Community Library) modul přidat v konfiguraci PHP. V případě, že se jedná o Windows

⁶ Drag&drop – umístění požadovaných dat na místo přetáhnutím myši

platformu, je nutné přidat DLL (Dynamic Link Library) knihovnu. U Linuxové verze se přidává modul SO (Shared Object). Rovněž MySQL DLL soubor musí být přidán ručně v konfiguraci PHP, jinak MySQL funkce nebudou pracovat. Funkce pro připojení přes telnet budou muset být kompletně naprogramovány, kvůli specifickým požadavkům CISCO zařízení.

2.2 Součásti aplikace

Již při návrhu bylo nutné oddělit zobrazovací část od pracovní, tj. té, která se stará o zápis a modifikaci dat v databázi. Dále bylo třeba oddělit zápisové a modifikační skripty od těch, která data jen čtou. Ve všech však musí být přítomna kontrola uživatelských vstupů, aby se nemohlo stát, že nepřihlášený uživatel bez patřičných práv bude moci spustit svůj kód.

Z důvodu ochrany systému je tedy pro fungování aplikace nutné, aby v ní byl přihlášený schválený uživatel. Ten může být vytvořen pouze uživatelem, který má opět oprávnění pro vytvoření nového uživatele. Při první instalaci a spuštění je vytvořen výchozí uživatel „admin“, který má oprávnění pro veškeré činnosti. Pomocí něj bude možné vytvářet další uživatele a měnit práva těm již vytvořeným.

Celá aplikace je rozdělena na 3 základní bloky – GUI, konektivita a kontrolní část. Každá část byla vyvíjena odděleně a následně provázána přes interní operace do jednoho celku. Původní návrh aplikace počítal ještě s rozdělením na administrativní a uživatelskou část programu. To by však znamenalo vyšší množství obsahu, který by musel být explicitně rozdělen a byl by třeba i mnohem větší počet oprávnění. Proto byla nakonec zvolena pouze jednoúrovňové verifikace a to tak, že pro každou akci je definováno oprávnění a každé toto oprávnění obsahuje 3 další oprávnění pro zápis, změnu a čtení. S důrazem na jednoduchost budou volena na jediné stránce pro specifického uživatele. Každá programová sekce bude rovněž před svým načtením kontrolovat, zda uživatel má aktivní přihlášení a zda má oprávnění pro požadovanou akci.

Pro zabránění přímého spuštění skriptů musí být tyto kontroly prováděny rovněž i v exekuční části, spolu s ověřením všech vstupů na neplatné nebo příliš dlouhé řetězce. Veškerá aktivita, kde je třeba ověřit oprávnění, bude monitorována a záznamy budou ukládány do databáze pro eventuální audit a přehled změn.

Aplikace je rovněž vícejazyčná. Toto je řešeno načítáním PHP souboru, kde budou v poli uložené konstanty s textovými řetězci. Toto řešení umožní jednoduché překlady

pouhým přepsáním hodnot. Pokud se dodrží jmenná konvence u souboru, aplikace sama zobrazí dostupné jazyky v menu nastavení. V prvotní verzi se počítá pouze s anglickým textem.

2.3 Části aplikace a UML návrhy

V této části jsou popsány návrhy jednotlivých bloků aplikace. U velmi jednoduchých částí jsou UML bloky vynechány.

2.3.1 GUI

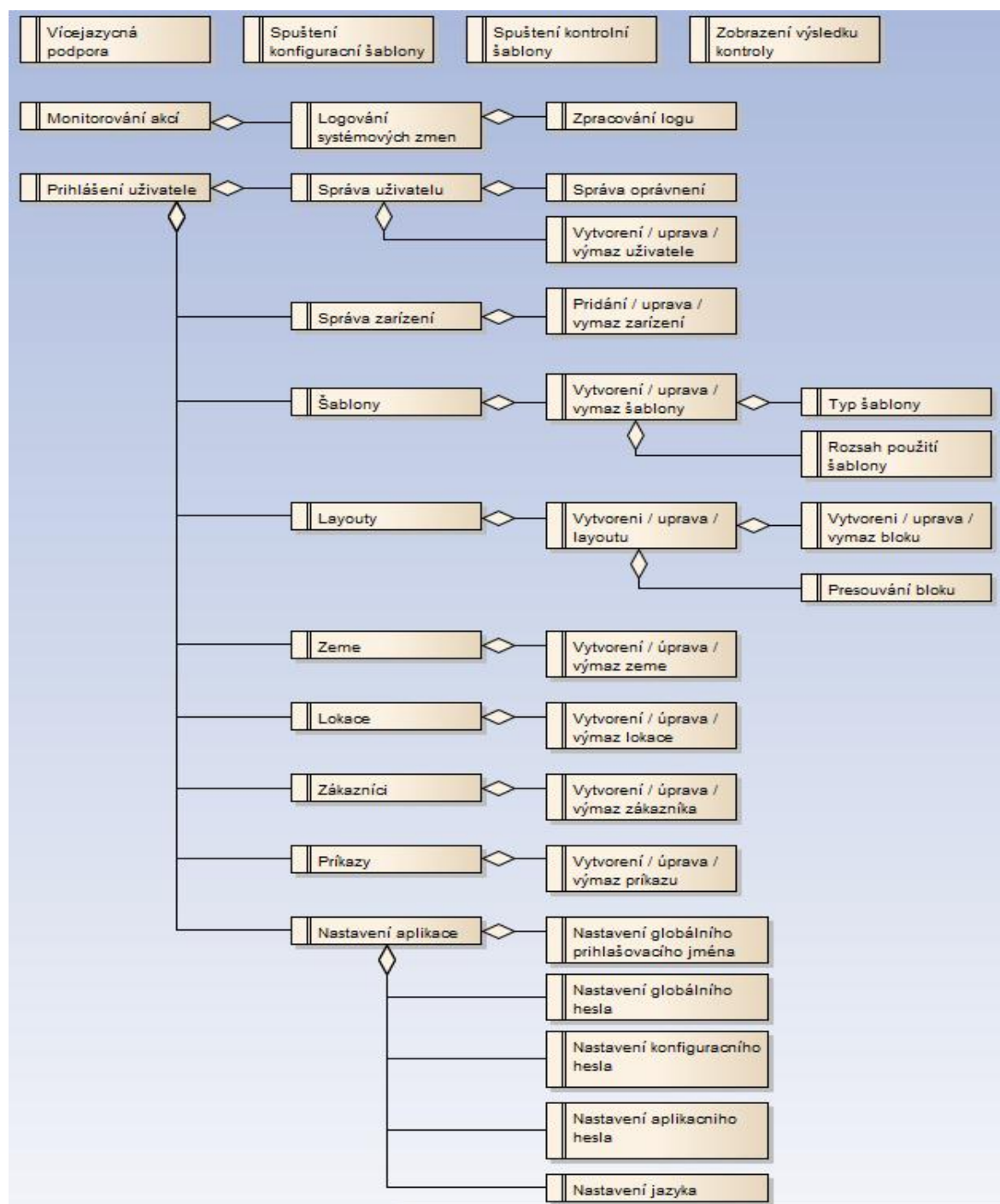
Aplikace obsahuje základní obrazovku s názvem úvodní stránky, přihlášeným uživatelem a menu na levé straně obrazovky. Celé řešení se snaží o zachování jednotného konceptu a vzhledu. V menu pak jsou položky pro správu systému a vlastní aplikace. V jednotlivých podsekcích jsou dále možnosti pro vytvoření nové položky a editace či výmaz stávající položky. Ke každé akci je samozřejmě třeba mít patřičné oprávnění.

2.3.2 Spuštění kontrolní a konfigurační šablony

Před vytvořením a spouštěním šablony je nutné, abych byla v systému zavedená zařízení, nad kterými chceme operace provádět. Cílem aplikace však není správa aktiv (asset management⁷), nicméně vždy je nutné vést alespoň minimální množinu informací o zařízeních, která mají být spravována a kontrolována. Jako povinné parametry jsou IP, jméno zařízení a model. Dále je nutné vybrat z předvybraných možností typ zařízení, status, zemi, lokaci a zákazníka. V případě země, lokace a zákazníka se bude jednat o možnosti již uložené v systému. Pokud bude potřebná jiná než dostupná hodnota, musí se vytvořit v požadované sekci. Rozšiřující a tedy nepovinnou hodnotou je sériové číslo a verze OS.

Jakmile je v systému zavedeno alespoň jedno zařízení, je možné spouštět kontrolní a konfigurační šablony. V případě kontrolních šablon je třeba vybrat seznam zařízení, na která se má šablona aplikovat. U konfiguračních šablon stačí pouze vybrat šablonu a veškeré její nastavení se poté načte z databáze např. aplikace na určitý typ zařízení, zákazníka atd.

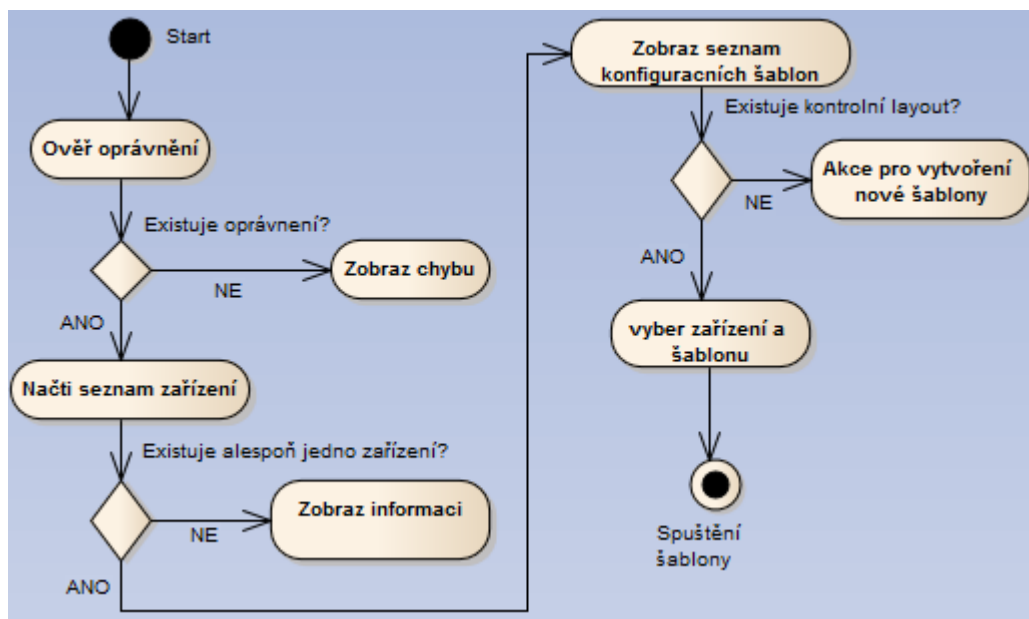
⁷ přehled o hardwarové a softwarové situaci v celé organizaci



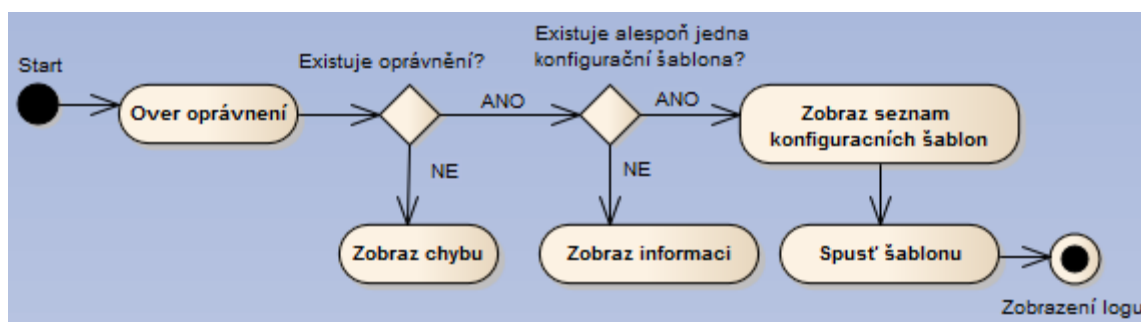
Obr. 1. Funkční požadavky

2.3.3 Správa zemí, lokací a zákazníků

Všechny části mají podobné požadavky na typ dat. V případě země se musí zadat název země a zkratka dané země. U lokace je třeba název, zkratka a následně ji přiřadit k požadované zemi. V případě zákazníka je nutný opět jen název a zkratka.



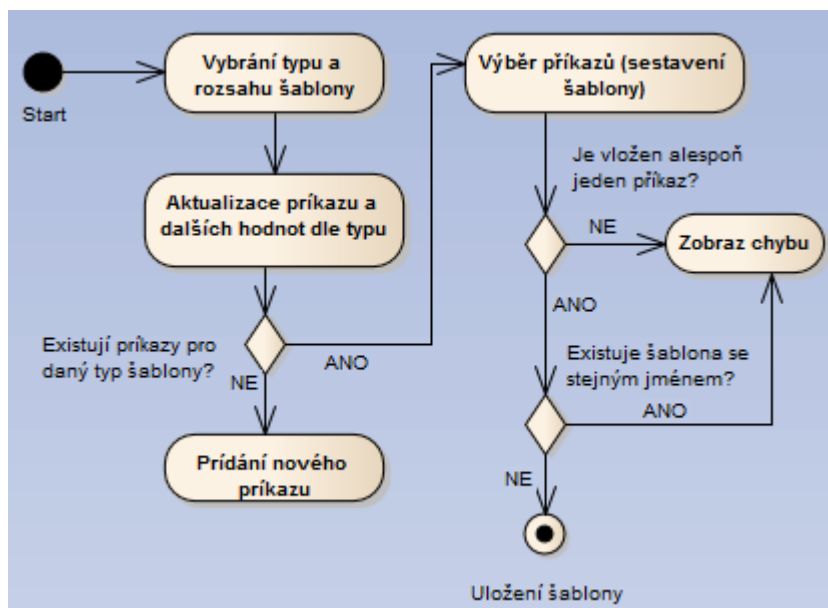
Obr. 2. Diagram aktivit pro spuštění kontrolní šablony



Obr. 3. Diagram aktivit pro spuštění konfigurační šablony

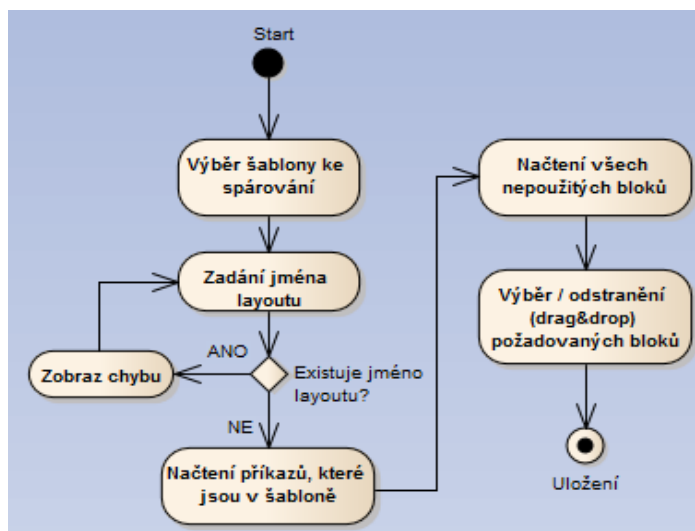
2.3.4 Správa šablon, příkazů a výstupních rozložení (layoutů)

U šablony musí být zadán název, typ a aplikovatelný rozsah – tj. zda se má aplikovat na zákazníka, verzi OS, typ zařízení atd. Další informace budou zobrazeny podle toho, jaký aplikovatelný rozsah je vybrán - tzn. že pokud je vybrán zákazník, zobrazí se seznam zákazníků, pokud je vybrána verze OS, zobrazí se seznam všech použitých OS verzí atd. Dále je zobrazen seznam všech příkazů, které je možné pomocí drag&drop přetahovat z kompletního seznamu do šablony. V případě, že je zakládána nová, nebo je editována již existující šablona, musí být vždy vybrán alespoň 1 příkaz. Typy jsou 2 a to čtecí (bez konfiguračního módu) a konfigurační. Seznam příkazů se aktualizuje podle toho, který mód máme právě vybraný.



Obr. 4. Diagram aktivit pro vytvoření nové šablony

Layouty jsou provázány se šablonami. Každý layout musí být s nějakou spárován. Šablona může mít více layoutů, obráceně to však neplatí. Části layoutu budou tvořit sekce příkazů, do kterých bude možné vkládat kontrolní bloky. Ty je možné přesouvat opět pomocí drag&drop. S bloky je také možné po umístění libovolně manipulovat, přesouvat, mazat nebo upravovat.



Obr. 5. Diagram aktivit pro vytvoření nového layoutu

Bloky – pojem blok se rozumí entita, která definuje hledané a kontrolované hodnoty. Blok musí obsahovat jméno, typ, řetězec, který se bude hledat (kontrolovat), za jakých okolností bude hledání platné a co se bude vypisovat. Typ může být jeden

z následujících: Statický blok, řádek, obsah, multi řádek a sekce. Implementační detaily a podmínky použití jsou uvedeny v 4.8.1.

Příkazy – v této sekci lze najít seznam všech uložených příkazů a samozřejmě je možné s nimi pracovat (editace, nový příkaz, výmaz). Pro vytvoření nového příkazu je nutné zadat alias pro příkaz a pak samotný příkaz. V současné době je možné zadat jen jeden příkaz do jednotlivého aliasu, tzn. že není možné spustit sérii příkazů jedním blokem. Samozřejmostí je ověření uživatelských práv pro danou akci.

2.3.5 Správa uživatelů

Zde je možné vytvářet uživatelské účty, přiřazovat k nim oprávnění a mazat je. Seznam základních kategorií oprávnění je následující: země, lokace, zákazník, zařízení, šablona, layout, uživatelé a příkazy. U všech těchto kategorií je možné vybrat ze 3 voleb – vytvoření (přidání), úprava a výmaz. Speciální oprávnění jsou u spuštění kontrolní a konfigurační šablony. Zde je, možné vybrat pouze povolení nebo zakázání.

2.3.6 Nastavení aplikace

V této sekci je možné nastavení jazyka (aktuálně pouze angličtina), dále pak změna vlastního uživatelského hesla a nastavení přihlašovacích údajů pro připojení k zařízení. Bude třeba, aby uživatel mohl bezpečným způsobem zobrazit aktuální hodnoty (pokud jsou nastaveny). V budoucnu budou možnosti nastavení aplikace podstatně rozšířeny dle potřeb uživatelů. Pokud je uživatel smazán ze systému, jsou v návaznosti na tuto akci automaticky smazány i všechny hodnoty nastavení pro daného uživatele. Tím je zamezeno ukládání dat v DB, která by již nikdy nebyla použita.

2.3.7 Připojení k zařízení

Tato část obsluhuje SSH a telnet připojení k zařízením. Protože se jedná o objektovou aplikaci, je toto řešeno knihovnou funkcí v podobě PHP třídy. Nezbytná data pro vytvoření objektu jsou:

- IP adresa – ta bude získána automaticky z databáze.
- Uživatelské jméno.
- Heslo k zařízení.

Volitelně může být zadáno ještě tzv. enable heslo. To slouží pro vstup do konfiguračního módu CISCO zařízení. Toto heslo není použité, pokud není spuštěna konfigurační šablona.

Všechna tato hesla bude možné globálně uložit pro každého uživatele. Musí obsahovat funkce pro získání CLI (Command Line Interface) a zejména rozhodovací funkce, které určí, zda byl přijat kompletní výstup. Řešení je popsáno v 4.3.4. Dále jsou přítomny funkce pro získání dat, jejich zpracování a konečné odpojení od zařízení. V případě SSH se bude jednat o zapouzdření SSH funkcí do uživatelské třídy. Pro telnet připojení je vytvořena zcela nová třída se stejnými funkcemi jako SSH knihovna.

2.3.8 Kontrolní část

Kontrolní část je finální část zobrazována při kontrole zařízení. Je dostupná po spuštění kontrolního layoutu a obsahuje kompletní výstup, který byl nakonfigurován v požadovaném layoutu pro všechna zařízení, na která byla aplikována. Vše bude zvýrazněno barvou, podle toho, jestli je nalezený řádek správný, nebo nikoliv. Je nutné, aby byla konfigurace ze zařízení stažena automaticky, jinak kontrola nebude pracovat. Aktuálně není možné nahrát konfiguraci jiným způsobem.

2.3.9 Deterministický konečný automat (DKA)

Speciálním případem kontrolní části je pak „sekční příkaz“. Konfigurace CISCO zařízení obsahuje mnoho částí, které se s drobnými obměnami opakují. Typickým příkladem je seznam rozhraní (interfaces), nebo routovací informace (OSPF či BGP sekce). Tyto sekce obsahují v podstatě identické informace s jinými hodnotami, které je třeba zkontrolovat. Zároveň je třeba, aby se z těchto sekcí získaly i informace, které se poté použijí pro výpis dat o sekci. Toto by bylo při klasickém řádkovém průchodu obtížné. Proto bylo třeba vymyslet řešení, které by toto umožnilo pokud možno co nejuniverzálněji a s možností dalšího rozšíření funkcionality v budoucnosti. Výsledkem je konečný automat, který bude možné uživatelsky konfigurovat.

Aby bylo možné získat jen relevantní informace, musí DKA umět zpracovávat klíčová slova a mít možnost ukládat jen části textu v závislosti na konfiguraci uživatelem. Proto se pro začátek počítá s klíčovými slovy, která získají data od aktuální pozice do dalšího zvoleného oddělovače, nebo do konce řádku. Tato data bude možné současně ukládat do proměnných, kumulovat je a následně s nimi dále pracovat.

Nejdříve bude nutné lokalizovat začátek sekce, kde má prohledávání začít. Následně skript bude procházet všechny řádky nakonfigurované v sekčním příkazu a bude hledat shodu v prohledávané sekci. Protože každý řádek obvykle reprezentuje informace,

placenou licenci. MySQL je multiplatformní databáze. Komunikace s ní probíhá – jak už název napovídá – pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními. Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích [10].

Aby bylo možné uložení používat, je třeba nejprve vytvořit databázi a tabulky společně s relacemi. Relace je, jak již název vypovídá, vzájemný vztah mezi různými tabulkami. Aby této funkcionality mohlo být využíváno, je nutné navrhnout DB tak, aby byly relace správné.

Aplikační DB nese název „hctool“ (z anglického healthcheck tool – nástroj pro kontrolu systému). Databáze pak obsahuje 15 tabulek, které obsahují data relevantní pro aplikaci. Každá tabulka musí mít nastaven engine, který definuje jaké vlastnosti a funkce budou poskytnuty při operacích nad tabulkami. MySQL jich obsahuje celou řadu. Výchozím enginem je MyISAM. V aplikaci je však použitý engine InnoDB, protože oproti MyISAM umožňuje používat cizí klíče. Rozdíly mezi těmito 2 hlavními enginy je popsán níže a text je převzat ze stránek Ondřeje Čecháka [11]. Popis účelu jednotlivých tabulek je v tabulce 1. Vztahy mezi jednotlivými tabulkami a datové položky tabulek (sloupce) je možné vidět v ER diagramu (Obrázek 7) nebo po spuštění aplikace v nástroji phpmyadmin.

Tab. 1. Seznam MySQL tabulek.

Název tabulky	Uložená data
blocks	Data bloků pro použití v layoutech, typ bloků, název a hledaný / vypisovaný text.
commands	Příkazy, jejich popis a typ
countries	Seznam zemí uložených v systému
customers	Seznam zákazníků uložených v systému
devices	Seznam zařízení uložených v systému
device_model	Modely zařízení
layouts	Informace o lyoutech a vazby na šablony
layout_blocks	Bloky použité v jednotlivých layoutech
log	Tabulka obsahující záznamy událostí z aplikace
sites	Seznam lokací
templates	Informace o šablonách a jejich typech
template_commands	Příkazy použité v jednotlivých šablonách a jejich pořadí provádění.
users	Seznam uživatelů a jejich oprávnění
user_settings	Nastavení pro jednotlivé uživatele

Název tabulky	Uložená data
versions	Seznam použitých verzí OS

2.4.1 MyISAM

„Je založen kódu staršího enginu ISAM (od MySQL 5.1 již není ISAM podporován). Tento engine je velmi rychlý, avšak v případě havárie nenabízí téměř žádné možnosti pro obnovu dat. Celá databáze je uložena ve složce, která obsahuje jednotlivé datové soubory. Tyto soubory jsou rozděleny do 3 kategorií, každá databázová tabulka se fyzicky skládá z 3 souborů:

- .frm - Definiční soubor tabulky,
- .MYD - Datový soubor, obsahuje samotná data,
- .MYI - Soubor s indexy.

Díky tomuto rozložení je sensitivita na velikost písmen ovlivněna sensitivitou file systému, na kterém jsou data uložena. Toto rozložení lze změnit a pro vyšší výkon rozložit soubory mezi více fyzických zařízení při vytváření tabulky. Maximální počet řádků v jedné tabulce je 2^{32} ve standardním nastavení. Maximální počet indexů v jedné tabulce je 64 ve standardním nastavení. Délka klíče je maximálně 1000 bajtů.“

2.4.2 InnoDB

„Je transakčně bezpečný ACID (Atomicity, Consistency, Isolation, Durability) splňující databázový engine. Podporuje commit, rollback a opravy při haváriích. Využívá ho například známý server Slashdot.org. Pro MySQL je vyvíjen firmou Innobase Oy, která je vlastně Oraclem. Narozdíl od MyISAM (Indexed Sequential Access Method) a Maria jsou data InnoDB tabulek uložena v tablespace. Do adresáře s databází se uloží pouze .frm soubor s definicí tabulky.

Velikostní omezení

- Tabulka nemůže mít více než 1000 sloupců.
- Délka klíče je maximálně 767 bajtů.
- Maximální velikost tablespace je 64TB.
- Maximální délka řádku včetně BLOB a TEXT je 4GB (prvních 768 bajtů je uloženo v řádku, zbytek ve zvláštních stránkách).“

2.5 jQuery

Javascript je multiplatformní, objektově orientovaný interpretovaný skriptovací jazyk, jehož implementace je součástí každého moderního webového prohlížeče. JavaScript byl původně obchodní název implementace společnosti Netscape, kde byl vyvíjen nejprve pod názvem Mocha, později LiveScript, ohlášen byl společně se společností Sun Microsystems v prosinci 1995 jako doplněk k jazykům HTML a Java.

Používání javascriptu pro komplikovanější použití, jako například práce s obrázky, soubory, animacemi a zejména práce s událostmi je poměrně nekomfortní. Velký problém pak nastává zejména v tom, že každý prohlížeč obsahuje implementační odlišnosti od standardizované verze, a tak se může stát, že stejný skript se bude chovat v různých prohlížečích různě, nebo nebude fungovat vůbec.

jQuery je oproti tomu od začátku stavěna jako knihovna s co nejširším použitím napříč prohlížeči., která klade důraz na interakci mezi JavaScriptem a HTML. Celá jQuery knihovna je vyvíjena jako opensource a větev 1.x je kompatibilní zpětně až s prohlížečem IE 6, a u všech ostatních prohlížečů jako Safari, Chrome, Firefox a dalších je plně funkční v aktuální verzi a ve verzi o 1 zpět. Vývojová větev 2.x již nepodporuje starší prohlížeče IE6, IE7 a IE8. Základní vlastnosti

- Výběr DOM (Document Object Model) elementů pomocí otevřeného cross-browser selektorového enginu Sizzle, odnože projektu jQuery.
- Funkce pro procházení a změnu DOM (včetně podpory pro CSS 1–3 a základní XPath).
- Události.
- Manipulace s CSS (Cascade StyleSheet).
- Efekty a animace.
- AJAX.
- Rozšiřitelnost.
- Utility – např. informace o prohlížeči nebo funkce `each`.
- Javascriptové pluginy.

Samotné jQuery však nemá příliš možností pro práci s HTML elementy, jejich animaci, a vylepšení samotného grafického rozhraní. Z tohoto důvodu je nutné použít rozšíření v podobě jQuery UI (User Interface). To obsahuje celou řadu widgetů (v češtině nemá

ekvivalent). Ty umožňují velmi jednoduché použití grafických vylepšení. Například práci s dialogy, menu, kalendáři, záložkami a další.

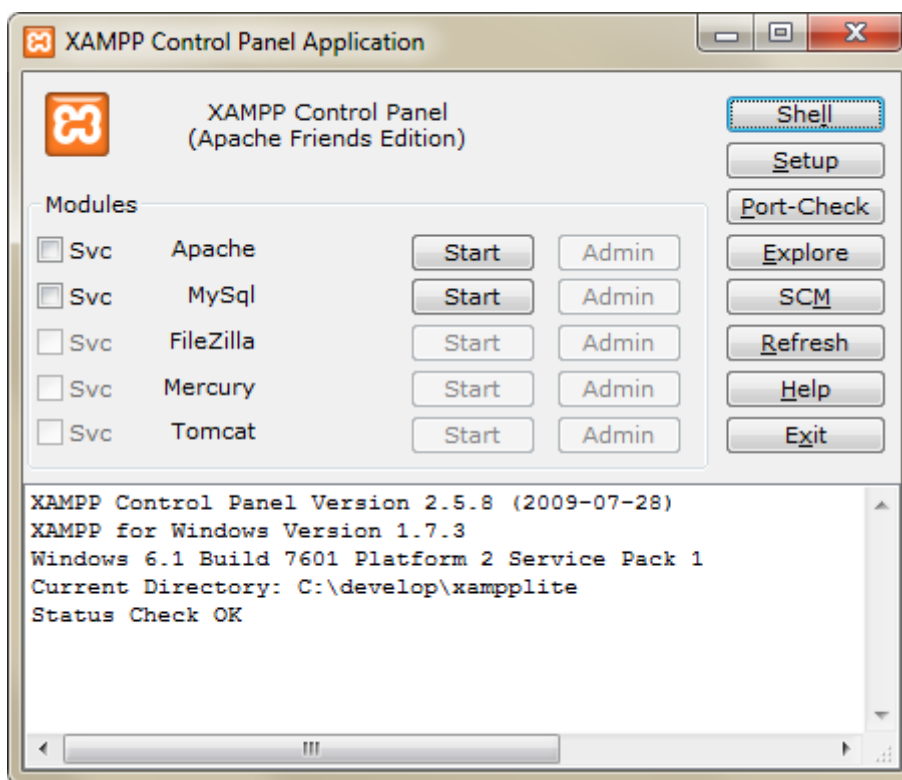
jQuery má však zásadní nevýhodu. Umožňuje práci s DOM, který obsahuje data platná v okamžiku načtení stránky. To znamená, že jakmile je ve skriptu přítomný element, který má přiřazenou akci na událost, tato akce bude fungovat pouze u těch elementů, které byly na stránce již v době vytvoření. To přináší problém u nově dynamicky vytvořených elementů, které jsou uloženy do DOM až při práci s dokumentem. Tento problém je však možné úspěšně vyřešit pomocí pluginu Livequery [12]. Pomocí tohoto malého pluginu je možné přiřadit požadovanou událost na všechny elementy, které jsou vytvořeny kdykoliv, tedy jak při načtení stránky, tak při práci s ní.

II. PRAKTICKÁ ČÁST

3 NASTAVENÍ A SPUŠTĚNÍ PROSTŘEDÍ

Přestože se jedná o přenositelnou aplikaci a veškeré potřebné podaplikace a servery jsou součástí tohoto balíku, je třeba pro řádné spuštění aplikace změnit některá nastavení v serveru Apache, MySQL a PHP knihovně. V této části jsou popsána nastavení Xampplite pro systém Windows. Vzhledem k tomu, že se v systému UNIX jedná o identický soubor programů, nastavení bude z velké části stejné.

Xampplite je možné stáhnout z [13] Ideální je komprimovaný soubor 7z. Po stažení je třeba jej rozbalit do adresáře xampp na disku C:\. Po rozbalení všech složek je nutné spustit soubor *setup_xampp.bat* aby došlo k aktualizaci kořenového adresáře pro podaplikace. Následně je možné zobrazit kontrolní panel souborem *xampp-control.exe*

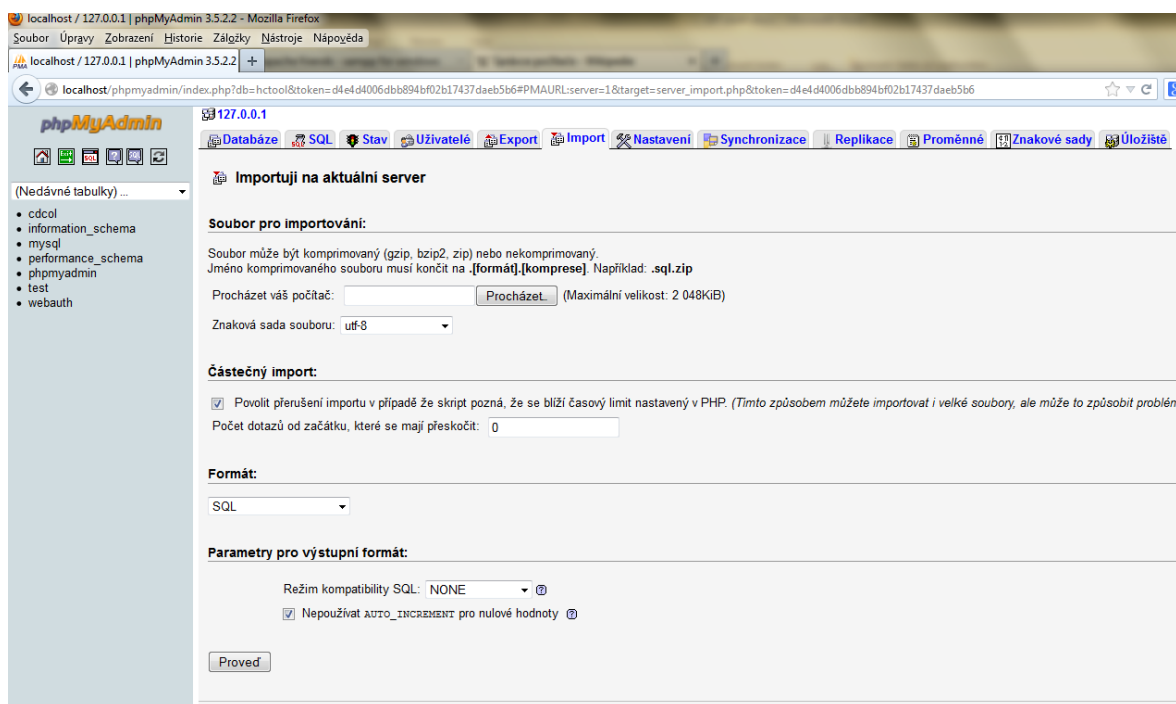


Obr. 7. Kontrolní panel aplikace Xampplite

Zde lze následně spustit několik programů. Pro aplikaci jsou relevantní pouze MySQL a Apache. Jakmile programy běží, musí být provedeny změny v nastavení a systému, aby bylo možné aplikaci používat.

3.1 MySQL

Nejprve je nutné vytvořit databázi, všechny pracovní tabulky a nastavit jim oprávnění pro aplikační účet. Pro tyto účely byl vytvořen MySQL skript, který vše udělá pouhým importováním do MySQL. Skript s názvem „hctool.sql“ je vložen v adresáři „_init“. Import lze nejpohodlněji udělat prostřednictvím webového správce databází – phpmyadmin⁸, který je standardní součástí xampplite. Správce je dostupný po spuštění Apache a MySQL na adrese <http://localhost/phpmyadmin>. Přihlášení probíhá poprvé automaticky s použitím root⁹ účtu a nezadává se žádné heslo. Heslo je však nutné vytvořit, aby se nemohlo stát, že neoprávněný uživatel bude moci zasahovat přímo do databáze a tím obejít aplikaci a narušit její běh. Detailní popis zabezpečení je nad rámec této práce a je k dispozici v AJ na stránkách phpmyadmin.



Obr. 8. Okno aplikace phpmyadmin, kde je třeba vložit MySQL skript.

⁸ Informace dostupné v [14]. Dokumentace dostupná v AJ.

⁹ Tzv. super účet na systémech UNIX. S tímto účtem je možné provést v systému libovolnou operaci. Typicky má k němu přístup jen pověřená osoba a pečlivě se chrání. Označení se vžilo pro účty stejného typu i na ostatních platformách a aplikacích.

3.2 Apache a PHP

V klasické instalaci serveru je nutné nainstalovat PHP a přidat jeho podporu do konfigurace serveru Apache. Dále pak v PHP doinstalovat chybějící moduly a přidat podporu pro MySQL. V xampplite je však vše potřebné již uděláno a stačí udělat pouze 2 drobné změny v konfiguračním souboru PHP.

Tou první je úroveň reportování chyb. Ve výchozím nastavení je úroveň hlášení chyb nastavena na nejvyšší hodnotu. To znamená, že se hlásí veškeré i minoritní chyby v systému. To je dobré při vytváření a ladění programu, nikoliv však v provozu. Proto je tedy třeba změnit konfiguraci v souboru *php.ini* a nastavit proměnnou *error_reporting* na hodnotu *E_ALL & ~E_NOTICE & ~E_STRICT*. Nastavením na tuto hodnotu říkáme enginu¹⁰ PHP, aby nevypisoval informativní hlášky a další systémová hlášení (např. chybějící index pole, které zkoušíme číst).

Druhým nastavením je import knihovny podporující SSH funkce. Ta je nezbytně nutná pro fungování části aplikace, která se připojuje k zařízením a stahuje konfigurační soubory. V souboru *php.ini* v sekci Dynamic Extensions (dynamická rozšíření) je třeba přidat řádek *extension=php_ssh2.dll* a současně je nutné mít tento soubor – *php_ssh2.dll* – v adresáři *ext*. Po provedení těchto úkonů je třeba restartovat Apache server, aby došlo ke znovunačtení požadované knihovny.

V průběhu počáteční fáze vývoje aplikace byl značný problém s nalezením DLL knihovny s SSH funkcemi pro novější verzi PHP než je 5.3.1. Většina zdrojů na internetu doporučovala vlastní kompilaci knihovny ze zdrojových souborů, což bylo z praktických důvodů zamítnuto. Není možné, aby se po každé změně verze PHP kompilovala nová knihovna pro SSH. Z toho důvodu byla použita starší, nicméně plně funkční a naprosto dostačující verze xampplite s verzí PHP, pro kterou již zkompileovaná knihovna existuje. Knihovna je dostupná na stránkách rozšíření pro xampplite [15].

Bohužel se vyskytl při instalaci knihovny ještě jeden problém. Pokud se knihovna (potažmo celé PHP) používá pod serverem Apache, musí být zkompileována kompilátorem Visual Studio 6 (VC 6). Pokud by byla aplikace instalována pod IIS (Internet Information

¹⁰ Engine – v překladu pohon, v tomto kontextu spíše výkonný stroj. Kód, který se stará o veškeré zpracování PHP skriptů.

Services), musí být knihovny a PHP kompilovány programem Visual Studio 2008 (VC 9). Proto bylo nutné najít verzi knihovny, která nejen odpovídá verzi PHP, ale současně i způsobu použití. Další je třeba rozhodnout, zda je třeba použít Thread Safe a Non Thread Safe. Verze Thread Safe může být nainstalována jako modul Apache. Kdežto Non Thread Safe se používá jako instalace CGI (Common Gateway Interface) [16].

Tyto problémy, které vyvstaly až v průběhu implementace, se nakonec podařilo úspěšně vyřešit a knihovna funguje tak jak bylo očekáváno při návrhu. Avšak v budoucnosti by tento postup nemusel fungovat, protože jak PHP, tak i vlastní knihovna se bude zcela jistě dále vyvíjet. Proto dojde v následující verzi aplikace s největší pravděpodobností k přechodu na jiné open source řešení, a to seclib [17].

Seclib je v PHP psaná sada skriptů a funkcí, které plně nahradí přidané rozšíření a aplikace se tak stane plně nezávislá na jakýchkoliv přidaných funkcích. V této aplikaci je však již zakomponována funkční knihovna a proto není nutná žádná další akce.

4 IMPLEMENTACE APLIKACE

PHP 5 je objektový jazyk. Proto bylo těchto vlastností využito a klíčové funkce byly zapouzdřeny do tříd. Celé řešení sestává z několika klíčových komponent, které spolu vzájemně spolupracují a využívají se. Každá stránka obsahuje jednotnou strukturu, která je složena ze stejné hlavičky, kde se vkládají nezbytné soubory, těla stránky a zápatí. Soubory se rozumí základní nastavení aplikace, javascripty, CSS styly a další. Dále se pak ověřuje, zda má přihlášený uživatel oprávnění ke zvolené akci, a zda mu nevypršela přihlašovací session.¹¹

Celá aplikace byla napsána v angličtině včetně komentářů. Důvodem je to, že aplikace bude v blízké budoucnosti rozšířena o nové funkce a uvolněna jako open-source na internetu. Cílem je vybudovat základnu uživatelů, kteří by aplikaci nasadili prozatím v malých prostředích a poskytli tak zpětnou vazbu spolu s nápady na další rozšíření, vylepšení a zejména pak testovací výsledky, pro odstranění případných chyb.

4.1 Adresářová struktura a popis souborů

Soubory jsou seskupeny v adresářích dle vlastností a funkce, kterou plní. Pro přístup k jakékoliv stránce je nutné být přihlášen. Jediná stránka, která je dostupná bez přihlášení je *index.php*. Tak se označuje výchozí soubor v adresáři a webový server jej automaticky použije, pokud není v URL uvedeno jinak. Všechny ostatní stránky musí ověřit, jestli má uživatel k dané činnosti oprávnění a současně i jestli je stále platná session. Doba platnosti je nastavena na 15 minut. Po této době je uživatel automaticky přeměrován zpět na *index.php*, kde je třeba se znovu přihlásit. Veškeré akce jsou však tímto zrušeny. V horní části stránky je ale vždy zobrazen čas do vypršení přihlášení. Kdykoliv v průběhu tak stačí stránku obnovit, nebo přejít na jinou. Tím se session obnoví zpět na 15 minut.

Jednotlivé stránky jsou pojmenovány anglicky podle svého účelu, který plní. Název souboru obsahuje prefix *admin*. To označuje, že je třeba být přihlášen, aby byla daná stránka dostupná. Výjimkou jsou pouze *index.php* a *logout.php*, které zajišťují odhlášení a přihlášení. Soubory budou popsány v následujících kapitolách, které se zaměřují na jednotlivé části aplikace.

¹¹ Česky sezení. Jsou to informace o jednotlivých přihlášených uživateli

Tab 2. Seznam složek a popis jejich obsahu

Název složky	Popis
class	Obsahuje soubory s definicemi tříd. Každá třída je v separátním PHP souboru
docs	Průvodní dokumentace k práci
exec	Soubory, pro práci s databází. Jsou obvykle volány pomocí AJAXU a jsou rozděleny na RO/RW skripty.
Img	Složka s obrázky a ikonami
Inc	Složka, kde jsou uloženy soubory, které se jen přiřkládají (include). V tomto případě deklarace DKA, jazykové mutace a další soubory pro fungování aplikace.
js	Složka s javascripty
styles	Složka s CSS styly

Ve složce *inc* jsou obsaženy i soubory *settings.php* a *functions.php*. Slovo *settings* znamená v češtině nastavení. V tomto souboru najdeme tedy nastavení celé aplikace. To zahrnuje definice konstant, statických polí, která se používají například pro rolovací menu, vložení všech potřebných souborů, které jsou v této složce a pak samotné nastavení prostředí, jako přihlašovací údaje k databázi, nastavení údajů o uživateli a nakonec vytvoření objektů pro databázi, správu chyb a HTML prvky. *Settings.php* je vkládána v každé stránce, protože vytvářené objekty jsou globální a jsou využívány v rámci celé aplikace.

Podobný případ je i se souborem *functions.php*. Ten obsahuje globální systémové funkce a vkládá se v souboru *settings.php*. Funkce jsou tedy dostupné v celém kontextu aplikace. Obsahuje 4 základní funkce pro generování jednotného vzhledu každé stránky. Názvy funkcí začínají znakem „_“.

Tab. 3. Soupis funkcí pro generování HTML výstupu

Název funkce	Popis
_print_header	Vypíše hlavičku HTML souboru. Parametrem je potom název stránky, dále pak pole CSS stylů a poslední pole javascriptů.
_top	Vypisuje blok, ve kterém se zobrazuje přihlášený uživatel, aktuální stránka a čas do konce session
_menu	Vypisuje menu na levé straně obrazovky. V případě potřeby umožňuje jednoduchou změnu v položkách menu.
_tail	Zakončení HTML stránky

Ověřování stavu přihlášení a uživatelská práva jsou kontrolována systémovými funkcemi. Začínají předponou *sys_* a jsou rovněž v souboru *functions.php*. Níže je uveden stručný soupis.

Tab. 4. Soupis systémových funkcí

Název funkce	Popis
sys_Login	Zajišťuje přihlášení uživatele a vytvoření session pro uživatele
sys_LoadPrivileges	Načítá oprávnění do session pro přihlášeného uživatele. Znovunačítá i práva pro uživatele, kterým byla změněna v průběhu session.
sys_Translate	Jenoducjá funkce, která vrací řetězec podle volby jazyka pro aplikaci
sys_CheckLogin	Ověřuje na každé stránce, jestli je session ještě platná
sys_AllowAction	Ověřuje oprávnění pro požadovanou akci

4.2 Aplikační třídy a popis jejich funkcí

Aby byla vnesena do aplikace jistá jednotnost a také z důvodů další údržby kódu, byly funkce podobného typu zapouzdřeny do tříd. Těch je v aplikaci přítomných 5. Každá třída je v separátním souboru, které jsou umístěny ve složce *class* a mají i stejnojmenný prefix souboru. V případě potřeby se příkládají ke stránce, kde se s nimi má pracovat. Obsahují funkce pro práci s databází, správu chyb, třídy pro připojení k zařízení a jednoduché funkce pro vytváření formulářových prvků. Databázová třída a třída pro správu chyb je vkládána vždy a objekty jsou vytvářeny automaticky.

Každé funkci a prvku ve třídě musí být nastavena viditelnost v rámci aplikace tzn. že každý element ve třídě musí být uvozen jedním z klíčových slov *private*, *public* nebo *protected*. Stručně popsáno – *public* znamená, že funkce se může volat odkudkoliv z aplikace. *Protected* značí, že elementy mohou být využívány v rámci třídy a ve zděděných třídách. *Private* omezuje využití pouze na třídu samotnou, kde je element definován. Každá třída musí obsahovat i konstruktor. Je to speciální funkce, která vytváří instanci dané třídy a inicializuje proměnné.

4.2.1 Třída pro operace s databází

Různé typy dotazů do databáze jsou v aplikaci velmi časté. Proto bylo třeba vytvořit třídu funkcí, která by ty nejčastější volání funkcí zastřešila a umožnila jednotný způsob volání. Třída pro operaci nad databází je v souboru *class_DB.php*. Veškeré proměnné jsou deklarovány jako *private*, protože jsou využívány jen v rámci třídy a jejich inicializace je prováděna v konstruktoru. Většina funkcí je však definována jako *public*, aby bylo možné funkce používat libovolným způsobem. Jedinou funkcí, která nelze volat přímo je funkce

pro připojení k databázi. Ta je volána pouze interními funkcemi a není důvod, aby ji uživatel používal i v aplikaci.

Tab. 5. Soupis funkcí databázové třídy

Název	Viditelnost	Popis
Connect	private	Zajišťuje připojení k databázi
Query	public	Posílá dotaz na databázi. Připojení si zajišťuje sama přes funkci Connect.
FetchAssoc	public	Vrací pole, kde každý řádek je asociativní pole dle databázového dotazu
NumRows	public	Vrací počet řádků v předešlém dotazu
GetLastID	public	Vrací ID řádku, které bylo vytvořeno v posledním dotazu. Použitelné pouze, pokud předešlý dotaz byl typu „INSERT“

4.2.2 Třída pro řízení chyb

Každá aplikace musí mít systém pro zachyt a reportování chyb. Tato není výjimka. Proto byla vytvořena třída, která se stará o výpis a logování veškerých uživatelských vstupů. Je uložena v souboru *class_ERR.php*. Původně obsahovala třída funkce pro přímý výpis chyby a pro návrat chybového řetězce. Funkce pro výpis byla nakonec odstraněna, protože v průběhu ladění aplikace docházelo k nesprávným výpisům v místech, kde k nim nemělo dojít. To je problém zejména, pokud nastane chyba před vytvořením session. Session lze totiž vytvořit, pouze pokud před tím nebyly zaslány http hlavičky a žádný další výstup. Dále je nutné většinu akcí i logovat do DB, aby bylo možné zpětně dohledat, kdo spustil jakou akci, či jinak ovlivnil systém. Standardně je systém nastaven tak, aby se ukládala i méně důležitá hlášení. Například vytvoření nového zařízení, úprava šablony apod. Vždy je k dané události přidáno datum a uživatel, který akci inicioval.

Třída obsahuje i funkce pro dočasné přenastavení systémového nastavení pro zápis do databáze. Využívá se zejména dočasné vypnutí logování do DB. To se využívá v situacích, kdy ještě není ustaveno databázové spojení a chybová třída by generovala opětovnou chybu, protože nemůže zapsat předešlou chybu do DB. Nicméně, některé typy chyb nemohou být zpracovány touto třídou, protože nastávají ještě před vytvořením globálních objektů. Jsou to chyby, které většinou ovlivní funkčnost aplikace, proto se zobrazuje chybové hlášení s informací, aby uživatel kontaktoval administrátora aplikace, který zjedná nápravu.

Tab. 6. Soupis funkcí třídy Error

Název	Viditelnost	Popis
UseDB	private	Vybírá databázi, kam se budou ukládat logy
ReportSilently	public	Vrací chybovou zprávu a zapisuje log do DB
Log2DB	private	Zapisuje log do DB
SetDBLogging	public	Manuálně povolí zápis do DB
UnsetDBLogging	public	Manuálně zakáže zápis do DB

4.2.3 Třída pro vytváření HTML elementů

Třída pro HTML prvky je jen uskupení funkcí, generující HTML formulářové prvky. Ty jsou v aplikaci poměrně hojně zastoupeny a tak bylo zapotřebí definovat jednotný vzhled a funkčnost. Obsahuje však také funkce, které usnadní generování formulářových prvků ze systémových dat, zejména prvky pro výběr hodnoty (SELECT). Třída je uložena v souboru *class_HTML_object* a všechny funkce jsou definovány jako veřejné.

Tab. 7. Soupis funkcí třídy HTML elementů

Název	Viditelnost	Popis
TextField	public	Vypisuje textové pole
Password	public	Vypisuje textové pole typu password. Není tedy vidět obsah
Select	public	Vypíše menu (tag SELECT). Data přijímá jako asociativní pole, kde musí být definovány názvy sloupců, které použije jako vypsaný text a hodnotu. Vhodné pro data z databázových dotazů.
SelectFromArray	public	Vypíše menu (tag SELECT). Jako parametr přijímá asociativní pole, které obsahuje pouze jeden sloupec – konstantní data, není třeba definovat vybírané sloupce.
Button	public	Vypisuje tlačítko. Povinné jsou text a identifikátor. Může ale obsahovat i další parametry.
Checkbox	public	Vypisuje zaškrtačací pole.
Textarea	public	Vypisuje textovou oblast – textové pole pro delší text

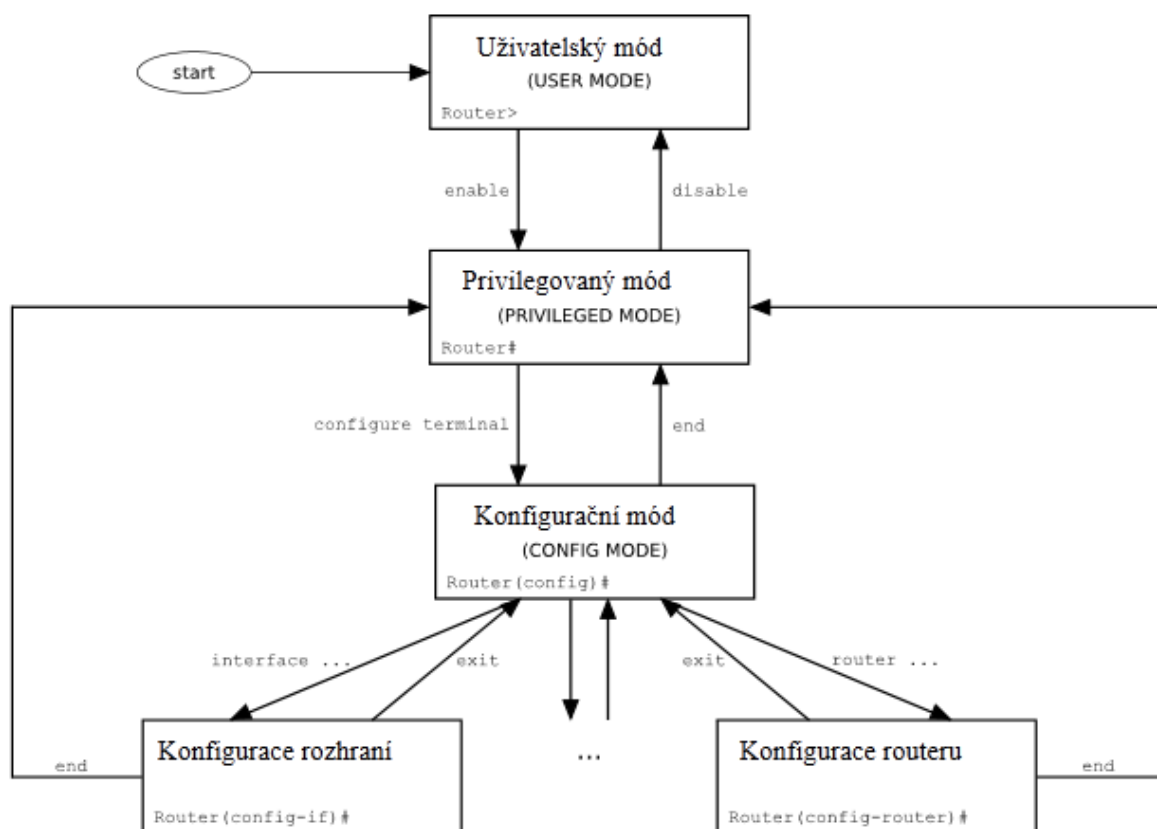
4.2.4 Třídy pro připojení k zařízení

Nejběžnější způsob připojení k zařízení je přes protokoly SSH a Telnet. Tyto 2 protokoly bylo tedy třeba začlenit do aplikace. Přenos dat pomocí SSH, jak z názvu vyplývá je zabezpečený. To znamená, že komunikace není vidět v TCP protokolu jako běžný text při případném záchytu. Dá se téměř s jistotou tvrdit, že pokud dané zařízení podporuje SSH, používá se vždy jen tato metoda spojení. Telnet je pak obvykle k dispozici jen pro velmi stará zařízení, jednoduchá zařízení nebo ta, která SSH podporu nemají zabudovány.

Aby SSH fungovalo v PHP, bylo nutné změnit jeho konfiguraci a přidat knihovnu podporující SSH funkce. Detaily jsou popsány v 3.2. Pro Telnet žádná knihovna funkcí není k dispozici. Není v podstatě ani třeba. Tím, že je telnet nešifrovaný, je možné použít běžné síťové funkce pro čtení a zápis datových streamů. Je jen třeba v těchto funkcích

použít číslo portu 23, který je pro něj standardně určen. Protokol telnet má několik řídicích příkazů, které však v našem případě mohou být ignorovány. Eventuální chybové výstupy jsou pak ošetřeny samotnou třídou.

Pro kontrolu CISCO zařízení a správnou funkčnost celého systému je nezbytné získat po připojení k zařízení prompt¹². Ten vždy obsahuje název zařízení a je zakončený znakem „>“. Pokud se nacházíme v konfiguračním módu, je prompt zakončen znakem „#“. V promptu mohou být mezi názvem zařízení a koncovým znakem další řetězce v závislosti na tom, zda jsme v nějaké specifické části konfiguračního módu. Pro lepší přehled je přiložen obrázek:



Obr. 9. Architektura systému cisco IOS

Aby bylo možné využít i konfigurační mód, je třeba mít nastavena správná systémová oprávnění. Ta jsou spravována administrátorem a jsou uložena obvykle na

¹² Prompt (výzva) indikuje, že příkazový řádek je připraven zpracovat další příkaz. Je zobrazen vždy po dokončení předchozího příkazu.

serveru TACACS, nebo RADIUS (Remote Authentication Dial In User Service). Protože se může stát, že ke každému zařízení bude mít uživatel různá oprávnění (k některému zařízení jen čtecí, k jinému i konfigurační), je třeba detekovat prompt automaticky a vstoupit do enable módu jen v případě potřeby.

Funkčnost obou tříd je obdobná. Mají podobné funkce i jejich skladbu. Vždy je třeba se připojit k zařízení, získat prompt a následně posílat příkazy a získávat výstup. Pro fungování SSH je však třeba po připojení vytvořit interaktivní shell¹³ na připojovaném zařízení a vyhradit pro něj stream. Přihlášení probíhá v obou případech tak, že je nejdříve učiněn pokus o připojení. Zařízení zašle zprávu, kde je třeba zadat uživatelské jméno a to samé se opakuje i pro heslo. Pokud se uživatel není schopen přihlásit, je vrácena chyba s touto informací. Přihlašovací informace jsou uloženy v globálním nastavení aplikace pro každého uživatele.

Po úspěšném přihlášení je prompt definován a uložen jako poslední řádek výstupu. Z těchto dat je poté získáno jméno zařízení a dle posledního znaku je určeno, zda je přihlášený uživatel pouze v uživatelském módu, nebo v privilegovaném módu. Pokud je pouze v uživatelském módu, program se pokusí vstoupit do privilegovaného módu pomocí enable hesla. Pokud není definováno, pak pokračuje jen v případě, že uživatel spustil jen čtecí šablonu a nepokouší se vstoupit do konfiguračního módu. Jméno zařízení je nutné získat proto, aby se při dalších voláních funkcí mohl správně definovat výstupní prompt. Jak bylo zmíněno výše, ten je využíván pro ukončení načítaného výstupu ze zařízení. Pouhé jméno zařízení, nebo jen poslední znak není možné použít, protože se tyto znaky a řetězce mohou běžně objevovat i při výstupu některých příkazů a načítání by tak bylo nesprávně ukončeno předčasně.

Při přihlášení k terminálu přes externí aplikaci je vidět, že v případě množství dat většího než 80 řádků, což je standardní délka terminálu, je třeba stisknout mezerník pro další výstup. V aplikaci by toto chování bylo možné nasimulovat zasláním ASCII hodnoty 32, která značí právě mezerník. Pro CISCO však existuje elegantnější řešení, které umožní získat výstup najednou bez potřeby stisku nějakých dalších znaků. Po přihlášení k zařízení je jako první odeslán příkaz „terminal length 0“ a dále se pokračuje uživatelskými příkazy.

¹³ Jedná se o typ CLI. Každý shell má jiné vlastnosti a různé podporované příkazy. Jako překlad by bylo možné použít „příkazový procesor“

Obě třídy obsahují funkci, která odešle příkaz a načte jeho výstup. Pro tuto činnost je k dispozici i vestavěná funkce v PHP. Její naprosto zásadní nevýhoda však je, že po odeslání příkazu a zpracování výstupu je ukončeno spojení. CISCO zařízení bohužel neumějí přijmout více příkazů současně. Tento problém by tak musel být řešen opakovaným přihlášením k zařízení a neustálým zasiláním příkazů, což je velmi nešťastné.

Čtení dat ze zařízení se provádí v nekonečném cyklu, kde program při každé iteraci ověří, zda jsou v bufferu přítomna nějaká data. Při příkazech, kde zařízení generuje větší množství dat a trvá tedy delší dobu jeho odpověď je třeba vyčkat nějakou dobu na příchod dat. Proto když nejsou v bufferu žádná data, čeká program 100ms a poté znovu ověří přítomnost dat. Pokud tato situace trvá 5 sekund, je skript přerušeno a je navržena chybová hodnota Timeout. Jako příklad lze použít spuštění příkazu pro zobrazení konfigurace na zařízení, které obsahuje stovky kB textu, kde samotné generování může trvat i několik sekund. Pokud se nevyskytne žádná chyba, je čtení ukončeno získáním promptu.

Tab. 8. Seznam funkcí třídy SSH

Název	Viditelnost	Popis
SSH_Connect	public	Připojuje se k zařízení přes SSH
_createShell	private	Vytváří interaktivní shell, aby mohly být poslány příkazy
_sendCommand	private	Posílá příkaz na zařízení a získává výstup
_getFirstPrompt	private	Získává prompt po přihlášení
SSH_enable	public	Umožňuje přejít do enable módu
SSH_Exec	public	Uživatelská funkce, která posílá příkaz na zařízení
SSH_IsPrivileged	public	Testuje, zda je přihlášený uživatel v privilege módu
getRAWOutput	public	Vrací celkový výstup jako asociativní pole, kde klíče jsou vlastní příkazy.
getArrCommandOutput	public	Vrací výstup jediného příkazu
SSH_Disconnect	public	Odpojení od zařízení

Tab. 9. Seznam funkcí třídy Telnet

Název	Viditelnost	Popis
_Telnet_Connect	private	Připojuje se k zařízení pomocí telnetu
_Telnet_Login	private	Přihlášení k zařízení
Telnet_Exec	public	Uživatelská funkce, která posílá příkaz na zařízení
Telnet_Enable	public	Umožňuje přejít do enable módu

4.3 Systém překladů

Aplikace byla od začátku koncipována jako vícejazyčná, a proto bylo třeba najít systém, který by vyřešil překlady textu. Nabízela se 3 řešení. První řešení bylo použít databázi pro ukládání jazykových mutací pevně daných textů. Databázové řešení by však bylo poměrně pomalé, protože by se každý text musel načítat ze serveru a navíc by bylo třeba vytvořit

rozhraní pro správu jazykových verzí. Jako další řešení bylo možné načítat překlady ze souboru. To by obnášelo vytvořit strukturu souboru, který by bylo možné zpracovávat. Načítání ze souboru je však ještě pomalejší, než načítání z databáze. Ze souborového řešení se však vycházelo.

Jako ideální model se jevil použít soubor PHP, který se vloží jako konstantní pole. Výhodou tohoto řešení je, že je velmi snadné vytvořit novou jazykovou verzi pouhým přepsáním řetězců a navíc se vyhneme načítání obsahu souboru a tím je razantně zvýšena rychlost načítání.

Všechny texty, které je třeba překládat, jsou získávány pomocí funkce *sys_Translate*, kde parametrem je hodnota asociativního klíče. Ta vrací textový řetězec, který se poté použije v aplikaci. Po úvodním načtení je následně celé pole s překlady uloženo v uživatelské session. Tím je docíleno globálního dosahu všech řetězců. Toto však neplatí pro obsah, který se generuje až na straně klienta pomocí AJAXu. Ten je třeba získat dynamicky. Jak bylo zmíněno výše, pro navrácení textového řetězce pro vybraný jazyk existuje funkce *sys_Translate*. Pokud by byla použita pro tento případ, musela by být volána mnohonásobně opět pomocí AJAXu. Tím by ovšem vznikla zbytečná režie při přenosu dat, nemluvě o tom, že tento způsob práce není nejrychlejší. Z toho důvodu byla vytvořena jednoúčelová stránka *ajax_get_translation.php*, která vrátí všechny požadované přeložené řetězce ve formátu JSON. Všechny požadované řetězce musí být definovány v javascriptu jako pole řetězců.

4.4 Správa a uživatelů a systémové zabezpečení

4.4.1 Systémová oprávnění a jejich popis

Uživatelská oprávnění jsou rozdělena na kategorie a každá tato kategorie je potom ještě dále rozdělena na dílčí části a to pro přidání / vytvoření elementu, změna elementu a výmaz elementu. V aplikaci je přítomen po instalaci uživatel s názvem admin, který obsahuje standardně práva na všechny činnosti. Pokud je tedy uživatel přihlášen pod tímto uživatelem, může přidávat v aplikaci nové uživatele, vytvářet zařízení a další. Současně umožňuje měnit práva všech stávajících uživatelů.

Po aplikační stránce je systém oprávnění implementován jako sada konstant v PHP, které jsou mapovány na názvy sloupců v databázi. Níže je uveden příklad definice oprávnění pro přidání nových zemí: *define('PRIV_COUNTRY_A', 'country_add')* V tomto

případě je *PRIV_COUNTRY_A* konstanta, která je poté použita v programu a *'country_add'* je název sloupce v databázi uchováající oprávnění. Oprávnění jsou načítána ihned po přihlášení uživatele a jsou uložena v session pro globální přístup. Vlastní ověřování se tedy již neprovádí dotazy do databáze ale pouhým zavoláním funkce *sys-AllowAction*, kde parametr je typ oprávnění, který je kontrolován. V průběhu testování však bylo zjištěno, že takto postavený systém neumožňuje jednoduše zablokovat některého uživatele, pokud je aktuálně přihlášený (má platnou session). V další verzi systému bude tento nedostatek odstraněn přepracováním řešení na ryzí online kontrolu. Verifikace uložená v session bude tedy kompletně odstraněna a nahrazena kontrolou přes databázi, kam se poté budou ukládat informace o přihlášeném uživateli.

Tab. 10. Mapování aplikačních práv na sloupce v DB tabulce

Konstanta	Sloupec v tabulce `users`	Popis
PRIV_COUNTRY_A	country_add	Přidání země
PRIV_COUNTRY_M	country_modify	Editace země
PRIV_COUNTRY_D	country_del	Výmaz země
PRIV_SITE_A	site_add	Přidání lokace
PRIV_SITE_M	site_modify	Editace lokace
PRIV_SITE_D	site_del	Výmaz lokace
PRIV_CUSTOMER_A	customer_add	Přidání zákazníka
PRIV_CUSTOMER_M	customer_modify	Editace zákazníka
PRIV_CUSTOMER_D	customer_del	Výmaz zákazníka
PRIV_DEVICE_C	device_create	Přidání zařízení
PRIV_DEVICE_M	device_modify	Editace zařízení
PRIV_DEVICE_D	device_del	Výmaz zařízení
PRIV_TEMPLATE_C	template_create	Přidání šablony
PRIV_TEMPLATE_M	template_modify	Editace šablony
PRIV_TEMPLATE_D	template_del	Výmaz šablony
PRIV_EXEC_RO_TEMPLATE	exec_ro_template	Spuštění čtecí šablony
PRIV_EXEC_RW_TEMPLATE	exec_rw_template	Spuštění konfigurační šablony
PRIV_USERS_C	users_create	Vytvoření uživatele
PRIV_USERS_M	users_modify	Editace uživatele (jeho práv)
PRIV_USERS_D	users_del	Výmaz uživatele
PRIV_COMMANDS_C	commands_create	Přidání příkazů
PRIV_COMMANDS_M	commands_modify	Editace příkazů
PRIV_COMMANDS_D	commands_del	Výmaz příkazů
PRIV_LAYOUT_C	layout_create	Vytvoření layoutu
PRIV_LAYOUT_M	layout_modify	Editace layoutu
PRIV_LAYOUT_D	layout_del	Výmaz layoutu

4.4.2 Zabezpečení aplikace

Daný typ oprávnění je vždy kontrolován v části aplikace, ke které se váže. Pokud uživatel prochází část v GUI, pak stačí, aby měl alespoň jedno z dané trojice oprávnění. To mu umožní vidět obsah dané sekce. Pokud dané oprávnění nemá, je mu zobrazena chybová hláška. Pro příklad: Uživatel má pouze oprávnění pro vytváření nové šablony. Po kliknutí na položku v menu se mu zobrazí stránka pro kompletní správu šablon. Vytvoření nové šablony je umožněno dle oprávnění. Pokud by se však pokusil změnit již existující, nebo některou vymazat, bude mu zobrazena chybová hláška a celá akce bude rovněž uložena v logu aplikace. Při pokusu o přístup do jiné sekce, např. správy zařízení, bude mu zobrazena opět chybová hláška s informací, že do dané sekce nemá přístup. I tato operace bude uložena v systémovém logu.

HTML a javascript jsou volně dostupné k prohlížení jako zdrojový kód. Z tohoto důvodu je možné, že se některý uživatel pokusí zobrazit zdrojový kód stránek nebo javascriptu a přistoupit na stránky, které přímo pracují s databází tak, že stránku zavolá přímo s daty a parametry, které je stránka schopná zpracovat. Všechny operace jsou uloženy v souboru `exec_operations.php` ve složce `exec`. Pokud je tedy požadována operace nad databází, je volána právě tato stránka, kde je nastavena požadovaná akce a jsou přiložena data ke zpracování. Po zavolání stránky je nejprve ověřeno, zda je přihlášený uživatel a zda má platnou session. Pokud ne, je tento pokus přesměrován na přihlašovací stránku a je pořízen log. Pokud je uživatel přihlášený, ale pokouší se spustit akci, ke které nemá oprávnění, je zobrazena chyba a opět je pořízen záznam o této akci. Systémový administrátor se poté může rozhodnout, jak s daty naloží.

Spolu s operacemi musí být kontrolována i vlastní data. Prvotní kontrola se provádí již na straně klienta. V případě, že data mají být unikátní je již při vyplňování pole ověřováno v databázi, zda se v ní vyskytuje zadaná hodnota. To eliminuje uživatelské překlepy a zmenšuje čas potřebný pro opětovné vyplnění dat. Dále je pak rovněž kontrolován typ zadávaných dat a jejich délka. Při výskytu chyby není ani možné data odeslat. Po úspěšném odeslání je ještě provedena hloubková kontrola dat přímo v prováděcím skriptu. Jsou zde ověřovány všechny parametry dat a také zda jsou vyplněny i hodnoty, které nelze uživatelsky změnit jako různé identifikátory elementů, podle kterých je možné přímo určit měněný záznam.

Ověřování dat, která nejsou zapisována do DB, ale je třeba je z DB získat, probíhá přes dotazy v souboru `ajax.php`. Ten je opět nutné volat s parametry (podobně jako v `exec_operations.php`), kde jedním z nich je akce a druhým jsou data, která má verifikovat. Obvykle se jedná o jména šablon, layoutů, či IP adresy. Soubor obsahuje i řadu verifikačních funkcí.

4.5 Správa zařízení

Databáze zařízení je nezbytností pro fungování aplikace. U zařízení je možné uchovávat celou řadu parametrů a vlastností, pomocí kterých je můžeme rozlišit a současně sdružit do skupin, nad kterými chceme provádět požadované operace. Takovou skupinou mohou být například pouze routery s vybranou verzí systému. Cílem systému bylo, aby vzájemná kombinace několika parametrů umožnila získat co nejpřesněji danou skupinu zařízení bez dalších komplikovaných činností.

Parametry, které jsou v aplikaci ukládány, jsou následující: jméno zařízení, IP adresa, model, sériové číslo (nepovinné), verze OS, typ, status zařízení, země, lokace a zákazník. Kromě nepovinných polí musí být všechny prvky vyplněny. Stejně jako v případě správy uživatelů, i zde probíhá online kontrola zadávaných dat pro pole s IP adresou. Ta musí mít správnou syntaxi a navíc, musí být v rámci celého systému unikátní. Současně je podporován pouze typ IPv4¹⁴. Typ a status jsou generovány ze statických polí s hodnotami, která jsou definována v `settings.php`. Pole se seznamem zákazníků je generováno přímo z databáze z již uložených zákazníků. Pokud databáze žádného zákazníka neobsahuje, je zobrazena informace o tomto stavu a následně je uživatel vyzván, aby nějakého zákazníka vytvořil v příslušné sekci. Stejný případ nastává i v případě zákazníka. Zde je však ještě aktualizováno menu s lokacemi. To je získáno AJAX voláním stránky `exec/ajax.php`. Ostatní pole jsou omezena pouze délkou řetězce. IP adresa může mít délku v rozmezí 7 – 15 znaků. Maximální délka jména zařízení je 40 znaků a sériového čísla 25. Verze systému se ukládá do vyhrazené tabulky. Při ukládání zařízení je ověřeno, zda verze systému již existuje a pokud ano, pak se do záznamu zařízení uloží identifikátor OS v tabulce. Pokud neexistuje, je vytvořen nový záznam a ten je pak použit u zařízení.

¹⁴ Adresování IPv4 je původní systém adresování známý již od roku 1980. Adresa má 32 bitů a rozsah byl kompletně vyčerpán již v roce 2011. V současné době se přechází na systém IPv6, kde adresa má délku 128 bitů, avšak není dosud plně podporována napříč internetem.

Toto řešení je použito z toho důvodu, že je takto možné spustit šablonu i na danou verzi systému a díky unikátním identifikátorům OS při tom nemůže vzniknout žádná nesrovnalost. Současně tak vzniká databáze tak, jak ji uživatelé vytváří a není třeba udržovat seznam všech dostupných verzí OS.

Stránka se správou zařízení obsahuje kompletní výpis všech zařízení. Na pravé straně jsou pak umístěny ikony umožňující editaci a výmaz zařízení. Ty jsou zobrazeny bez ohledu na to, zda uživatel má k dané akci oprávnění či nikoliv. Po kliknutí na ikony je daná akce ověřena, a pokud má uživatel oprávnění pro provedené akce, je i vykonána. V případě, že uživatel oprávnění nemá, je mu zobrazeno chybové hlášení a událost uložena v logu.

4.6 Správa zemí, lokací a zákazníků

Správa zemí, lokací a zákazníků je velmi podobná. Po kliknutí na položku v menu se vždy zobrazí tabulka s výpisem již uložených hodnot a volby k editaci a smazání každé položky. Ověření práv pro editaci a výmaz se provádí stejným způsobem jako v případě zařízení. Akce je tedy ověřována až před vykonáním.

Úprava a vložení nové položky se provádí pomocí jednoduchého formuláře, který je ve formě modálního dialogu. Pokud položku editujeme, pak je již předvyplněna hodnotami z řádku, který chceme editovat. Hodnoty jsou načítány přímo z dané stránky pomocí javascriptu, a nikoliv znovu z databáze.

Při přidávání nebo editaci země nebo zákazníka je třeba zadat název položky a třímístnou zkratku. Název může sestávat jen z velkých a malých písmen a z mezer. Zkratka země může obsahovat pouze znaky a musí být přesně 3 znaky dlouhá.

Lokace musí obsahovat název, opět třípísmennou zkratku a současně musí být vybrána země ze seznamu. Ten je opět generován z databáze a musí obsahovat alespoň jednu hodnotu. Proto je třeba seznam zemí vytvořit předtím, než jsou vytvářeny lokace.

4.7 Správa šablon a příkazů

V aplikaci existují 2 druhy šablon. Jsou to konfigurační (config) a kontrolní (read-only). Oba druhy se upravují a spravují stejným způsobem. V horní části okna (oranžový blok) lze nastavit základní údaje o šabloně. Těmi jsou: název šablony, typ šablony a aplikační rozsah šablony. Poslední pole se mění v závislosti na tom, jaký aplikační rozsah je vybrán.

The image shows a web-based interface for updating a template. The main window is titled "Update template" and contains a form with the following fields:

- Template name: CONF
- Template type: Config
- Template scope: Site based
- Site: Milano

Below the form is a "Save" button. To the right of the "Save" button is a list of commands, each in a green box:

- test2
- conf-enable
- conf-username
- enter config
- exit
- ggg

At the bottom of the dialog is an "Add command" button. At the bottom right of the entire window are "Update" and "Close" buttons.

Obr. 10. Okno pro vytvoření a editaci nové šablony

V další části šablony jsou viditelné všechny dostupné příkazy pro daný typ šablony. Ty jsou umístěny v pravé části okna a jsou zobrazeny na zeleném podkladu. Seznam obsahuje všechny příkazy, které byly zadány do systému libovolným uživatelem. Každý příkaz musí mít jako vlastnost nastaven typ. Ten je shodný s typem šablony, tedy konfigurační a kontrolní. Seznam příkazů je aktualizován v závislosti na typu šablony. Jakmile je v průběhu editace, nebo při vytváření nové šablony vybrán jiný typ, seznam je okamžitě obnoven na korektní seznam příkazů a pokud byly některé příkazy již vybrány, jsou v tento moment odstraněny. Pro získání všech příkazů daného typu je volán pomocí AJAXu soubor `ajax.php` s typem akce `GET_ALL_COMMANDS_BY_TYPE`.

Při editaci šablony, je seznam již použitých příkazů získán z databáze před vlastním zobrazením okna opětovným zavoláním `ajax.php`, tentokrát však s typem akce `GET_ALL_TEMPLATE_COMMANDS`. Po získání hrubých dat je zavolána na straně klienta javascriptová funkce, která prověří, jaké příkazy jsou v šabloně použity a ty skryje v seznamu nepoužitých. Tento systém je použitý pro případ, že uživatel některý příkaz odstraní z použitých. Ten je následně pouze zobrazen a připraven pro opětovné použití. Není tak třeba vše znovu načítat z databáze.

4.7.1 Využití jQuery UI

Než bude popsáno vlastní využití jQuery, je nutné vysvětlit několik pojmů, se kterými se bude v další části práce pracovat. Toto je výčet jen použitých funkcí. Kompletní referenci lze nalézt na internetu na [18]

- Draggable – vlastnost HTML bloku (v aplikaci je to typicky DIV element), která zajistí, že celý element je možné přesouvat myší. U objektu lze nastavit, k jaké oblasti ho půjde připojit.
- Droppable – oblast (opět typicky DIV), do které je možné umístit. Objekt pouze s touto vlastností v aplikaci nenajdeme.
- Sortable – jedná se o blok s vlastnostmi jako Droppable, ve kterém je však možné bloky ještě řadit a přesouvat uvnitř bloku. Umožňuje nastavit do ní umístěvané bloky tak, že je možné je přesouvat jen v rámci stejné skupiny
- Resizable – umožňuje měnit velikost elementu přetahováním myši

Všechny bloky s příkazy a to jak použité, tak volné jsou transformovány na Draggable bloky a jsou spárovány s oblastí, kam je možné je umístit (blok na levé straně). Popis v bloku u příkazu je jen interní označení, které bylo zadáno při vytváření příkazu. Po najetí myši je zobrazen na krátkou chvíli celý příkaz. V průběhu přesunu jsou pak získány informace o objektu, jako rozměry, jméno a hlavně identifikátor. Ten musí být unikátní, jinak není možné s objektem, který má duplicitní identifikátor, pracovat. Z tohoto důvodu musí být identifikátor nového prvku drobně pozměněn a to tak, že je před původní název přidán prefix „used_“. Tak je možné extrahovat v případě potřeby i původní název elementu pouhým odstraněním tohoto prefixu.

Po přesunu příkazu do použitých jsou změněny CSS třídy daného elementu, podklad zbarví do světle žluté barvy a současně je zobrazena ikona pro odstranění z použitých

příkazů. Celá oblast je definována jako Sortable. To tedy umožňuje nejen přidávat nové bloky, ale umožňuje i řadit jednotlivé příkazy přetahováním myši uvnitř Sortable oblasti. Z této oblasti nemůže být blok přesunutý zpět. Musí být odstraněn ikonou.

Přímo v dialogu je možné vytvořit nový příkaz. Tlačítko pro jeho vytvoření je umístěno ve spodní části dialogu. Po úspěšném vytvoření je obnoveno celé okno. Před přidáním příkazu je nutné změny uložit, jinak budou po obnovení stránky ztraceny. Je třeba dbát i na posloupnost příkazů tak, jak jdou za sebou, protože i v tomto pořadí budou vykonávány na požadovaném zařízení.

Při ukládání šablony se ověřuje, zda jsou zadány všechny nutné hodnoty. Jméno šablony může být maximálně 30 znaků dlouhé a musí být unikátní. Ostatní vlastnosti jsou voleny pomocí menu, nicméně při ukládání je ověřeno, zda typ a hodnota opravdu odpovídá. Vždy musí být použit alespoň jeden příkaz, jinak nebude možné šablonu uložit. Na veškeré operace je samozřejmě nutné mít potřebná oprávnění.

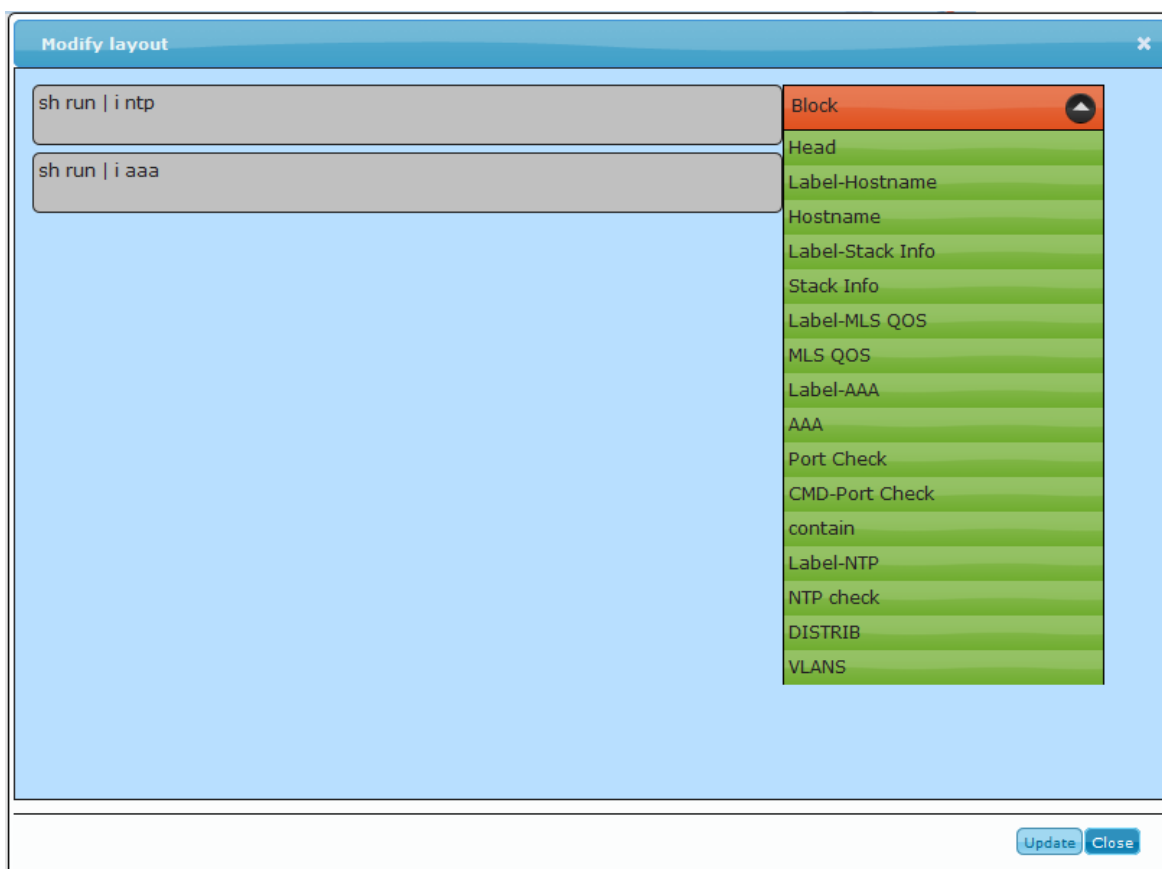
4.7.2 Správa příkazů

Příkazy se spravují stejně, jako ostatní položky. Po kliknutí na odpovídající položky menu je zobrazena tabulka se všemi příkazy, které jsou v systému uloženy spolu s možnostmi pro přidání, editaci a výmaz daného příkazu. Při vytváření a editaci příkazu musí být nastaven typ příkazu – konfigurační nebo kontrolní, popis příkazu, který je zobrazen ve výpisu příkazů v šabloně a který musí být v rámci systému unikátní a pak vlastní příkaz, který se pošle na zařízení. Popisek u příkazu může být maximálně 50 znaků a může obsahovat znaky, čísla a mezeru. Příkaz může mít délku až 255 znaků a skladba znaků není omezena. Je však možné zadat pouze jeden příkaz do položky. Celý řetězec se implementuje jako jeden příkaz.

V průběhu testování ukázalo, že uživatel by spíše ocenil, kdyby viděl jen příkazy, které vytvořil on sám. To stejné platí i o šablonách. V další verzi aplikace bude tedy ke každé položce v databázi přidána vlastnost s identifikátorem uživatele, který ji vytvořil. Současně bude u elementu možné nastavit, zda uživatel chce, aby element byl přístupný i pro jiné uživatele. Ti pak budou moci ovlivnit, zda uvidí jen svoje položky, nebo i položky ostatních uživatelů, kteří toto umožnili. Současně bude nutné i přidat vlastnost, aby daný uživatel mohl editovat, nebo mazat jen položky, které sám vytvořil. Tím se zamezí editaci a mazání položek jiných uživatelů.

4.8 Správa layoutů a výstup skriptu

Layouty tvoří jednu z klíčových částí aplikace a je na nich postavena značná část funkcionality. Jedná se o definici hodnot, které chceme zkontrolovat na jednom či více zařízeních a jejich rozložení na stránce. Výstup je poté zobrazen jako HTML stránka a označen dle kritérií, které si zde navolíme. Každý layout musí být spárován s kontrolní šablonou. U konfigurační šablony ho nemá z logiky věci smysl konfigurovat.



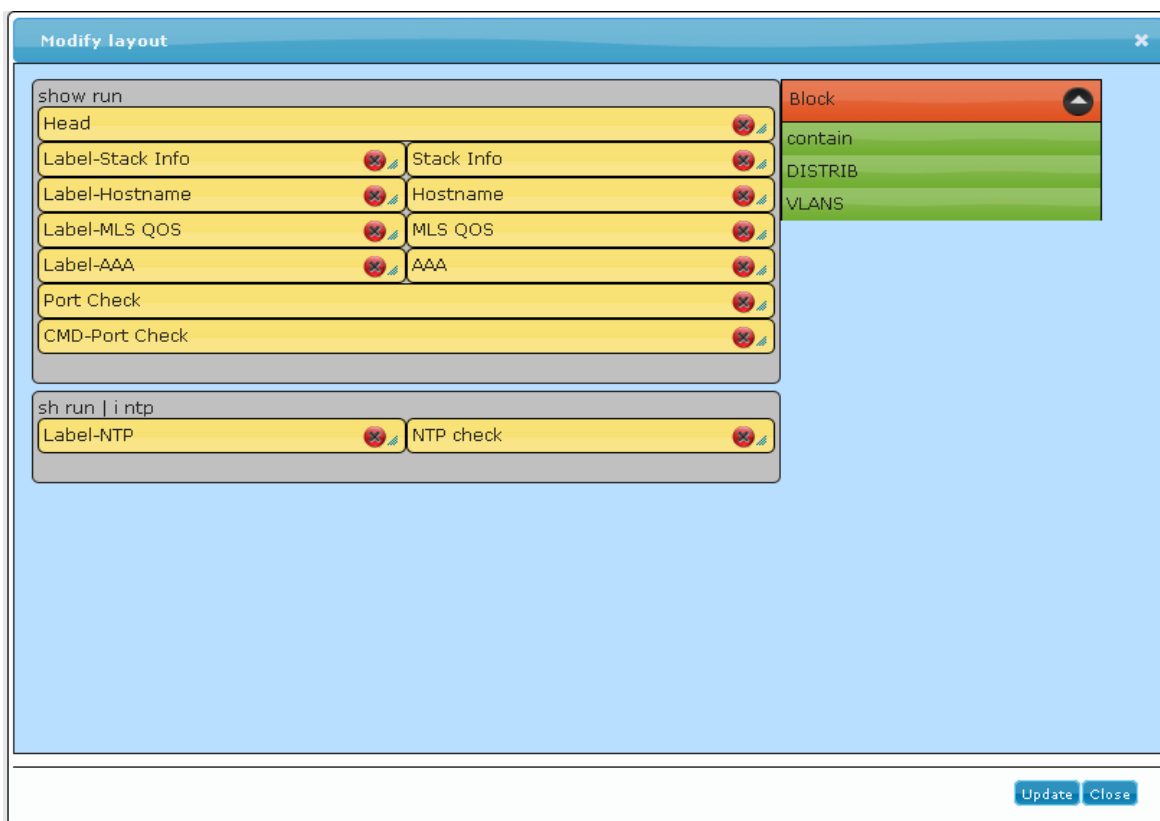
Obr. 11. Prázdný layout

Základním stavebním kamenem layoutu jsou bloky. Jsou to entity, které obsahují definici dat, která mají být v konfiguraci hledána a kontrolována. Jejich seznam je umístěn v pravé části okna. Chování bloků je podobné jako u příkazů při konfiguraci šablon. Bloky tedy jsou typu draggable a mohou být přesouvány do šedých oblastí. Na obrázku (Obr. 11) je výchozí zobrazení nového layoutu.

Každá šedá oblast reprezentuje příkaz v šabloně. Záhloví této oblasti obsahuje text příkazu, který je poslán na zařízení. Zde je tedy vidět, že šablona, se kterou je layout spárován obsahuje 2 kontrolní příkazy a to *show run / ntp* a *show run / aaa*. První vrátí ze zařízení všechny řádky obsahující „ntp“, což je konfigurace časového NTP (Network Time

Protocol) serveru a druhý vrátí ze zařízení všechny řádky obsahující „aaa“ což je autentizační protokol. Na každý z těchto výstupů lze aplikovat libovolné množství kontrolních bloků. Ty lze umístit do požadované oblasti přetáhnutím myši. Počet bloků, které je možné do oblasti přetáhnout není omezený. Můžeme tak zkontrolovat přítomnost, či naopak nepřítomnost některých klíčových řádků, či jejich částí a vypsat je.

Pro správné fungování bylo nutné nastavit několik věcí. Nejprve nastavit CSS třídy pro přetahování bloku a současně nastavit jiné třídy a vlastnosti, které budou aplikovány až po přetažení bloku do oblasti příkazu. Každý použitý blok má také nastavené v CSS obtékání zleva. To umožňuje, aby bloky byly automaticky řazeny tak, jak je vidět na obrázku a současně s nimi manipulovat tak, aby se vždy seřadily do požadovaného uskupení.

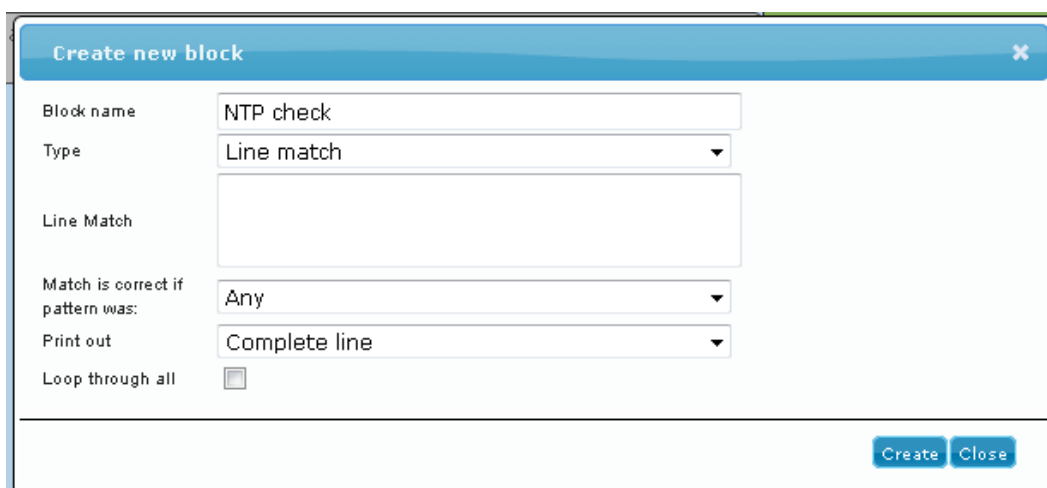


Obr. 12. Vyplněný layout

Standardně je možné bloky umístit ve 2 sloupcích. Takto vytvořené sloupce budou poté reflektovat i sloupce ve výpisové tabulce. Tímto řešením je eliminováno ruční vybírání umístění bloku a eventuální kolize ve finálním zobrazení. Současně je každý blok možné roztáhnout přes celou šířku okna. Javascriptem je omezeno, aby rozměr bloku přesáhl hranice oblasti. Šířka elementu může nabývat pouze 2 hodnot a to jedné poloviny

nebo celé šířky zobrazení. V případě úpravy šířky elementu je samozřejmostí přepočítání pozic všech ostatních elementů. Šířka přes celé okno je vhodná například pro nadpisy, nebo oddělovače sekcí apod.

Každá příkazová oblast je typu Sortable. Standardně je možné každé takové oblasti nastavit, aby do ní umístěné bloky bylo možné přesunovat i do jiných sortable oblastí. Tato vlastnost zde byla vypnuta. Možnost jak tohoto docílit je tak, že je použitý blok odstraněn a znovu vložen do jiné oblasti. Zamezí se tak nechtěnému přetažení bloku do jiné oblasti. Pro lepší orientaci, kam bude blok umístěn je vždy zvýrazněna oblast, do které se daný blok umístí.



Obr. 13. Okno pro vytvoření nového bloku

Aby nebyla stránka přeplněna ovládacími prvky, jsou nové bloky přidávány, upravovány a mazány přímo v aktuálním okně. Přidání nového bloku je možné dvojklikem na oranžové záhlaví seznamu bloků. Tento seznam je rovněž možné v případě potřeby skrýt pomocí ikony se šipkou. Editace použitého i nepoužitého bloku je možná dvojitým klikem na požadovaný blok. Informace o bloku se načítají online z databáze. Tlačítko pro výmaz bloku se objevuje po najetí myši na požadovaný blok. Při odstraňování se zobrazí ještě potvrzovací dialog. Stejný systém je použitý i při odstraňování použitého bloku. Po najetí myši na blok je zobrazena ikona pro odstranění bloku a ten je znovu zobrazen v seznamu nepoužitých.

4.8.1 Kontrolní bloky

Dialog pro vytvoření bloku je zobrazen na obrázku (Obr. 13). Všechny položky jsou povinné. Název bloku může být až 100 znaků dlouhý. Typ bloku může nabývat jedné

z pěti hodnot a to Statický, odpovídající celý řádek, řádek obsahující zvolenou hodnotu, víceřádkové porovnání a sekční příkaz. Další pole je textová oblast, kam je třeba vložit hledaný text. Popisek pole se mění dle toho, jaký typ bloku vybereme. Dále je třeba vybrat chování programu při výskytu hledaného řádku. Je třeba určit, zda je považováno za správné, když je řádek nalezen, nenalezen, nebo na tom nezáleží. V tom případě je vždy vypsán a označen jako správný. Takto nastavené chování pak umožní kontrolu nastavení, které je v konfiguraci považováno za výchozí (defaultní) a není tak vidět v konfiguraci. Poslední položka umožňuje nastavit, zda se mají ve výpisu objevit pouze řádky, které byly aplikací předtím označeny jako správné, chybné, nebo aby se řádek vypsál vždy, bez ohledu na to jestli byl správný či nikoliv.

Po testování byla ještě přidána další volba ve formě zaškrťovacího pole, kde je možné nastavit, aby se kontrola bloku provedla na celé konfiguraci a byly vypsány všechny výskyty odpovídající požadovanému vzoru. Standardně se totiž kontrola zastavila při prvním výskytu hledaného řádku, což bylo při návrhu považováno za vhodné. Praxe však ukázala, že možnost individuálního nastavení umožní daleko větší flexibilitu. Současný stav je tedy takový, že se hledají všechny výskyty u bloku typu „contains“ a první výskyt u ostatních typů. Toto nastavení není v současnosti možné ovlivnit. V budoucí verzi však bude možnost doplněna.

4.8.2 Typy bloků a jejich popis

Každý blok je vhodný k jinému typu kontroly. Zde jsou popsány všechny typy bloků, způsob použití a účel:

- Statický (static) – Statický blok je alias pro statický text. Je vhodný pouze k vypsání textu, který do něj zadáme. Je ideální jako oddělovač sekcí, nebo jako popisek pro data (tzv. label)
- Výskyt řádku (line match) – tento typ bloku vyhledává přesnou shodu textu v konfiguračním souboru. Hledání shody není citlivé na velká / malá písmena. V aktuální verzi aplikace je hledání ukončeno při prvním výskytu shody.
- Řádek obsahující zadaný text (Contains) – Typ Contains umožňuje vyhledávání částečných shod. Stačí tedy, aby se v konfiguraci nacházel text, který je předmětem vyhledávání a bude zahrnut do výsledků. Pro vyhledávání více možností současně je možné hledané hodnoty seskupit do společné množiny uvozené dvojitými složenými závorkami a oddělenými znakem | (pipe). Tyto množiny mohou být

opakovány. Výsledkem v aplikaci je potom kartézský součin všech částí množiny a textu mezi nimi, pokud je přítomen.

Příklad použití: *switch {{1/2/3/4}} priority*

tímto příkazem můžeme zkontrolovat přítomnost hned 4 řádků bez ručního vypisování. Konkrétní příkaz testuje, zda jsou přítomny switche zapojené do stacku.¹⁵ Výsledkem tohoto příkazu jsou 4 řádky obsahující kompletní řetězec, který je hledán, tedy:

switch 1 priority

switch 2 priority

switch 3 priority

switch 4 priority

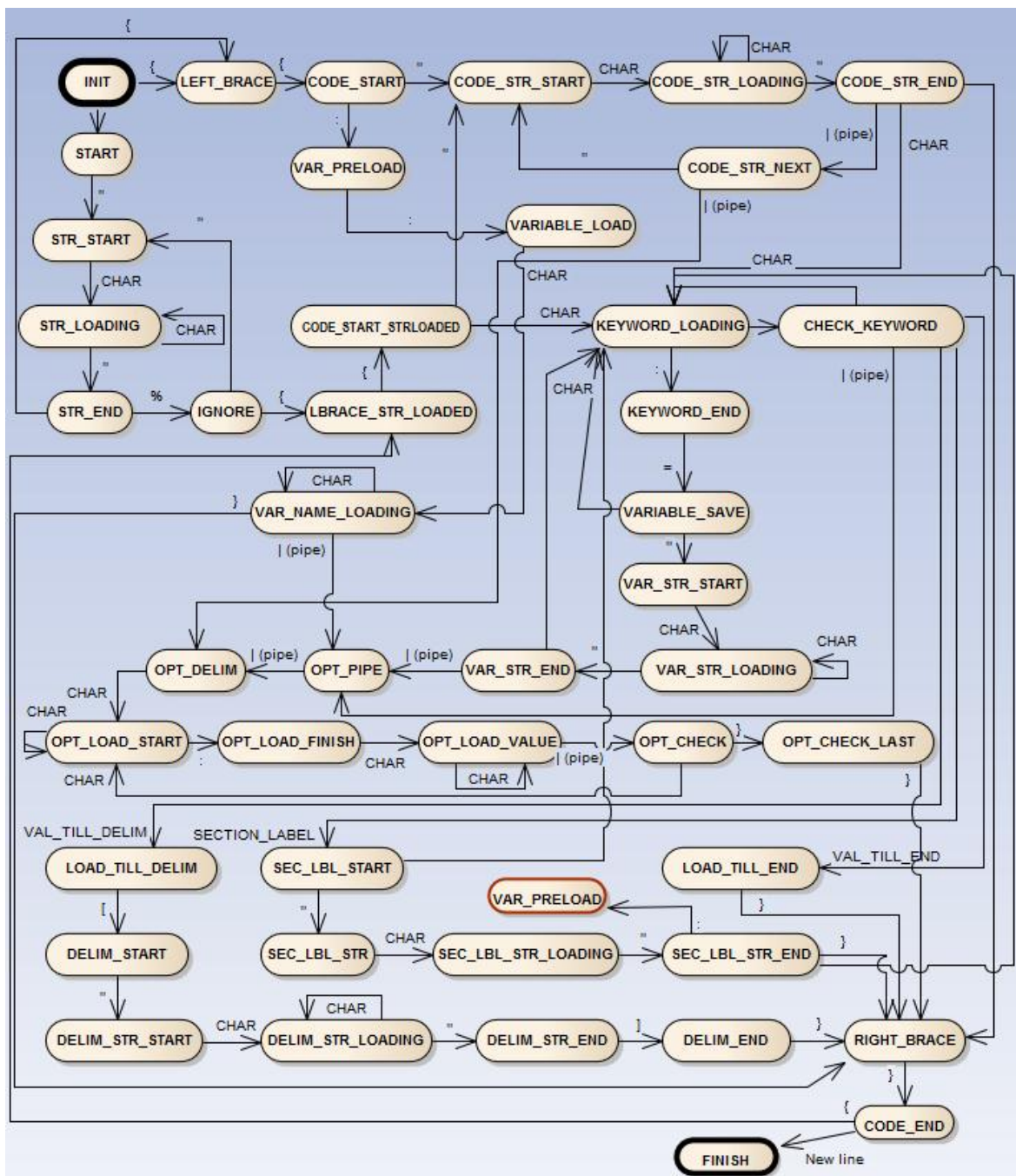
- Víceřádkové hledání (Multiline) – víceřádkové prohledávání funguje stejně jako řádkové s tím rozdílem, že je možné do hledaného textu více než 1 řádek. Každý nový řádek v textu reprezentuje řádek v konfiguračním souboru. To ušetří mnoho práce s vytvářením jednořádkových bloků a je rovněž praktičtější.
- Prohledávání sekcí (Section) – sekční příkaz je naprosto odlišný od všech předchozích. Umožňuje komplexní zpracování vstupu, generování vlastního výstupu s budoucí rozšiřitelností. Pro jeho zpracování byl vytvořen speciální automat, který je možné ovládat pomocí příkazů, tak aby generoval výstup, který chceme. Podrobné informace jsou v sekci 4.3.8

4.8.3 Sekční příkaz a implementace DKA

Účel sekčního příkazu je zpracování výstupu pro části výstupu, které se s drobnými obměnami opakují v kódu. Typickým příkladem je například kontrola konfigurace na zařízení pro jednotlivé porty, konfigurace OSPF, BGP a další. Obsahují totiž stejný typ a formát dat, pouze hodnoty pro jednotlivé řádky jsou odlišné. Toho lze využít k tomu, že každou takovou sekci lze zkontrolovat stejnou sadou příkazů a získat požadované výsledky relevantní každé sekci. K tomu je však třeba vyšší úroveň kontroly. Proto musel vzniknout

¹⁵ Switch stack: fyzické zapojení několika zařízení do jednoho logického celku.

system, který by toto umožnil, byl by snadno upravitelný a rozšiřitelný do budoucna. Z tohoto důvodu byl zvolen deterministický konečný automat (DKA) jako parser uživatelských programových sekvencí, tedy jednoduchý interpreter.



Obr. 14. DKA pro sekční příkaz.

4.8.4 Sekční DKA

Automat je definovaný jako uspořádaná pětice $M = (S, \Sigma, \delta, s, A)$, kde

- S je konečná neprázdná množina stavů

- Σ je konečná neprázdná množina vstupních symbolů (abeceda)
- δ je přechodová funkce popisující pravidla přechodu mezi stavy. Je ve formátu $S \times \Sigma \rightarrow S$
- s je počáteční stav, kde $s \in S$
- A je množina konečných stavů, kde $A \subseteq S$

DKA pro sekční příkazy je definován takto:

- $S = \{ \text{INIT, START, STR_START, STR_LOADING, STR_END, LEFT_BRACE, CODE_START, CODE_STR_START, CODE_STR_LOADING, CODE_STR_END, KEYWORD_LOADING, CHECK_KEYWORD, KEYWORD_END, VARIABLE_LOAD, VARIABLE_SAVE, SEC_LBL_START, SEC_LBL_STR, SEC_LBL_STR_LOADING, SEC_LBL_STR_END, LOAD_TILL_END, LOAD_TILL_DELIM, DELIM_START, DELIM_STR_START, DELIM_STR_LOADING, DELIM_STR_END, DELIM_END, RIGHT_BRACE, CODE_END, OPT_PIPE, OPT_DELIM, OPT_LOAD_START, OPT_LOAD_FINISH, OPT_LOAD_VALUE, CODE_STR_NEXT, OPT_CHECK, OPT_CHECK_LAST, LBRACE_STR_LOADED, CODESTART_STRLOADED, VAR_STR_START, VAR_STR_LOADING, VAR_STR_END, VAR_PRELOAD, VAR_NAME_LOADING, IGNORE, FINISH, FAILURE} \}$
- $\Sigma =$ všechny ASCII hodnoty
- Přechodové funkce jsou definovány v automatu na obrázku 15
- $s = \{ \text{INIT} \}$
- $A = \{ \text{FINISH} \}$

Jako abecedu lze použít všechny znaky v ASCII tabulce. Správný zápis by byl velmi zdlouhavý a nepřehledný. Automat tedy přijímá libovolný znak. Aby byla aplikace lépe čitelná a udržovatelná je každý znak je v aplikaci mapován na token. Naprostá většina znaků je mapována na token CHAR. Pro některé specifické znaky byly v aplikaci vytvořeny speciální tokeny, které rozlišují běžný znak od řídicího. Mapování znaků je v tabulce 11. Vstupem automatu je tedy skript, který definuje, jaké řetězce se mají hledat a jak se s nimi má zacházet. Počet řádků ve skriptu není omezen. Zpracování a příprava struktur pro kontrolu dat je však poměrně zdlouhavá, proto z hlediska výkonnosti je dobré nepřekračovat počet 15 – 20 řádků. Každý řádek skriptu může začínat řetězcem, nebo blokem kódu. Ten je uvozen dvojicí složených závorek na začátku i na konci. Je tedy ve formátu `{{ uživatelský kód }}`.

Tab. 11. Mapování znaků na tokeny

Načtený znak / řetězec	TOKEN
{	LBRACE
"	QUOTES
%	PERCENT
:	DOUBLEDOT
	PIPE
Načtený znak / řetězec	TOKEN
=	EQUALS
[L_SQL_BRACKET
]	R_SQL_BRACKET
VAL_TILL_DELIM	TILL_DELIM
VAL_TILL_END	TILL_END
SECTION_LABEL	SEC_LBL
}	RBRACE
\n	EOL
vše ostatní	CHAR

Jakmile program detekuje blok typu Section, Je celý skript načten z DB a zpracován funkcí DFA_ParseScript. Ta má za úkol projít všechny řádky skriptu a rozložit ho na části (struktury). Každá část má určen typ a nastaveny příznaky, pomocí kterých je možné blok aplikovat na data a zaručit správné chování. Tyto struktury jsou poté aplikovány na vlastní konfiguraci zařízení. Struktury jsou ukládány ke zpracování vždy, když je dokončeno zpracování úvodního řetězce, řetězce v kódu, nebo celého bloku kódu.

V cyklu je tedy volána funkce, která vrací aktuálně načtený token DFA_GetToken. Mapování načtených znaků na tokeny je uvedeno v tabulce 11. Ihned po navrácení tokenu je volána funkce DFA_GetState, která na základě předchozího tokenu aktualizuje stav automatu. Jako poslední je volána funkce DFA_Process. Ta pomocí stavu a tokenu určí chování stroje a modifikaci struktury, která se aplikuje na konfigurační soubor. V automatu je však vždy uložen i předchozí stav a token.

Pokud řádek začíná řetězcem, je použit pouze pro vyhledání řádku, který se bude zpracovávat a dále s ním není pracováno. Vždy však musí následovat blok kódu obsahující vlastní výkonný skript. V případě, že je hledán obsah, který je až za dlouhým textovým řetězcem, který navíc může být i proměnný, je možné využít znaku „%“. Ten je možné použít mezi dvěma řetězci, které slouží jako rozlišení začátku a konce textu. Tak můžeme skriptu říct, že vše co je mezi nimi má přeskočit, aniž by zkoumal obsah.

Parser obsahuje dále 3 klíčová slova, pomocí kterých můžeme řídit výstup a ovlivňovat chování. Prvním z nich je „SECTION_LABEL“. Pomocí něj je možné uložit popis dané sekce, který získáme přímo z kódu. To je velmi vhodné například pro získání názvu portu, který je aktuálně kontrolován. Dalšími klíčovými slovy jsou VAL_TILL_END a VAL_TILL_DELIM. První z nich vrátí zbytek aktuálně načítaného řádku, druhé vrátí text z aktuální pozice do zadaného oddělovače. Z toho vyplývá, že u druhého klíčového slova je nutné zadat oddělovač v hranatých závorkách. Všem klíčovým slovům musí předcházet řetězec, který parseru řekne, který řádek bude kontrolovat. Není tedy možné použít je samostatně.

Každému řádku je rovněž možné nastavit chování, které je považováno za správné a kdy se má řádek vypsát. Volitelné možnosti od kódu odděleny znaky ‚|‘ (dva znaky pipe za sebou). Nastavitelné možnosti jsou 2, a to REPORT (hlášení) a VALID (správnost). Hodnota pro danou možnost je pak oddělena znakem dvojtečka. Pokud jsou použity obě možnosti, je nutné je vzájemně oddělit dalším znakem | (pipe).

Hodnoty pro možnost REPORT:

- COMPLETE – Vypíše řádek vždy
- VALID_LINES – Vypíše pouze řádek označený jako platný
- INVALID_LINES – Vypíše pouze řádek označený za neplatný

Hodnoty pro možnost VALID:

- IFFOUND – Označí hledaný text jako správný, pokud je nalezen
- IFNOTFOUND – Označí hledaný text jako správný, pokud není nalezen
- ANY – Vždy označí hledaný text jako správný

Po dokončení zpracování uživatelského skriptu a načtení všech struktur je zahájena aplikace těchto struktur. V případě, že jsou v kódu přítomny řetězce, jsou tyto řetězce vypsány spolu s daným řetězcem a dalším požadovaným textem. Náhled celkového výstupu je možné najít v přílohách I a II.

V aplikaci byla rovněž připravena podpora pro ukládání hodnot do proměnných. Aplikační DKA již obsahuje stavy, které umožní načítat data z konfigurace do proměnných a dále pak s nimi pracovat. Bylo by tak možné provádět komplexnější operace nad konfigurací a získat tak širší možnosti jako například uložení všech SNMP serverů a jejich

následné další zpracování. Tato funkce však nebyla dostatečně otestována a proto se v aktuální verzi aplikace neobjevila. V následující verzi však již bude podpora připravena.

Vytváření aplikačních skriptů není bohužel příliš uživatelsky přívětivé. V dalších verzích proto bude přidáno grafické rozhraní pro vytváření skriptů a jejich přímé testování a přímá kontrola syntaxe. Ta je nyní zkontrolována až při spuštění kontrolní šablony. Celkově je třeba manipulace se sekčními příkazy významně vylepšit. Současný stav se dá považovat spíše za počáteční „beta“ verzi této funkce, kterou je třeba zdokonalit.

4.8.5 Příklad DKA skriptu.

```
"interface "{{SECTION_LABEL:=VAL_TILL_END}}
{{"ip arp inspection limit rate "VAL_TILL_DELIM[" "||REPORT:LINE_COMPLETE|VALID:IFFOUND
}}
{{"description "VAL_TILL_END||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"switchport mode "VAL_TILL_END||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"switchport access vlan "VAL_TILL_END||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"switchport voice vlan "VAL_TILL_END||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"speed 100"||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"duplex full"||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"srr-queue bandwidth share"VAL_TILL_END||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"priority-queue out"||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"no snmp trap link-status"||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"spanning-tree portfast"||REPORT:LINE_COMPLETE|VALID:IFNOTFOUND}}
{{"spanning-tree bpduguard enable"||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"spanning-tree guard loop"||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"switchport port-security"||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
{{"ip dhcp snooping limit rate 15"||REPORT:LINE_COMPLETE|VALID:IFFOUND}}
```

4.9 Spuštění kontroly a konfigurace zařízení

Spouštění šablon je rozděleno na 2 části. První z nich je spuštění konfigurační šablony, druhou pak spuštění kontrolní šablony. Seznam použitelných konfiguračních šablon je možné nalézt po otevření menu. Ve výpisu jsou potom šablony, které je možné spustit. Žádné další nastavení není třeba. Veškeré informace jsou zadávány při vytváření šablony. Pokud je tedy rozsah šablony typu „lokace“, pak jsou automaticky vybrána všechna zařízení, která spadají pod vybranou lokaci, a je na nich provedena požadovaná sada příkazů.

Spuštění kontrolní šablony je mírně odlišné. Při spuštění není vybrána šablona, ale layout, který se aplikuje na vybranou množinu zařízení. Layout je svázán se šablonou, a tak se vlastní příkazy získávají přímo z této šablony. Tyto příkazy pak mohou být poslány na libovolné zařízení. Kontrolu je tak možné provést na libovolném počtu a typu zařízení. Při spuštění kontrolní šablony je v aplikaci vypsán seznam všech zařízení. Pro lepší orientaci a

přehlednost byla vytvořena funkce, která umožní filtrovat seznam zařízení dle požadavků. Můžeme tak získat třeba seznam zařízení, který je kombinací určité lokace a zákazníka. Zaškrtnutím políček vedle jména zařízení je určeno, na která zařízení se má layout aplikovat.

Po spuštění kontroly je v aplikaci zobrazeno okno, ve kterém je zobrazen stav celé úlohy. Kontrola je spuštěna na separátní PHP stránce (na pozadí), ukládá aktuální výstup do textového souboru a tento soubor je periodicky kontrolován pomocí AJAXu na klientské straně. Po dokončení je v tomto okně zobrazeno tlačítko pro zobrazení grafického výstupu kontroly. Stejný výstup je možné kdykoliv poté zobrazit i pomocí menu „Zobrazit report“. Výstup je uložen v paměti do zavření okna prohlížeče, nebo do spuštění další kontroly.

4.10 Nastavení aplikace

Aplikace poskytuje i jednoduché možnosti nastavení. Aktuálně je možné v aplikaci změnit uživatelské heslo a dále pak nastavit přihlašovací jméno k zařízení spolu se standardním heslem a enable heslem. V návrhu aplikace se nepočítalo s ukládáním hesel pro jednotlivá zařízení, protože se ve velkých sítích používá TACACS, nebo RADIUS a hesla jsou tak pro všechna zařízení totožná. Při testování se toto však projevilo jako slabina, znemožňující jednoduché připojení k zařízení, které tyto ověřovací technologie nepodporuje. V případě potřeby kontroly takového zařízení je tedy pokaždé třeba změnit v nastavení přístupové údaje k tomuto zařízení.

V dalších verzích bude možné ovládat nejen chování aplikace (například změna výchozích hodnot pro uživatelské skripty), ale i nastavení vzhledu celé aplikace. K tomu bude třeba převést hodnoty CSS stylů do databáze a jejich následné načítání a změny.

4.11 Aplikační problémy a budoucí rozšíření aplikace

Přestože proběhlo základní testování celé aplikace, má aktuální verze celou řadu nedostatků a neodhalených problémů. Podrobnější testování přijde na řadu spolu s otevřeným testem. Jak bylo zmíněno výše, aplikace bude uvolněna k volnému použití a současně je očekáváno, že uživatelé, kteří aplikaci použijí, nahlásí objevené chyby, nestandardní chování či návrhy na vylepšení. V současné době jsou na seznamu vylepšení a oprav následující položky:

- Vylepšení uživatelských práv, jeho přehlednější zpracování a větší granularita práv (omezení uživatele na jednu zemi, lokaci apod.)
- Podpora Telnet knihovny. Ta byla z aplikace nakonec vynechána z důvodu minimálního otestování. V budoucí verzi bude přítomna pro podporu starších zařízení
- Specifické nastavení jména a hesla k každému zařízení
- Výběr typu připojení k zařízení (SSH, Telnet)
- Off-line kontrola zařízení (načtení zip souboru s konfiguracemi a aplikace layoutu)
- Systém intuitivní nápovědy
- Vylepšení systému správy příkazů a bloků. Každý uživatel uvidí pouze své příkazy a bloky spolu s těmi, které administrátor označí jako veřejné
- GUI pro sekční příkaz
- Rozšíření sekčního příkazu o možnosti ukládání do proměnných a podmínky
- Oprava načítání práv a přihlašování – přesun údajů se session do databáze
- Pokročilejší možnosti nastavení

ZÁVĚR

Přestože celý návrh a vývoj provázela celá řada problémů, podařilo se nakonec vytvořit požadovanou aplikaci, která je schopna zpracovat výstup ze zařízení, provést na tomto výstupu uživatelskou kontrolu, včetně pokročilejších konstrukcí konečného automatu, a zobrazit uživatelský výstup. Aplikace byla otestována v reálném prostředí na reálných zařízeních. Všechny nalezené problémy byly zaznamenány a některé z nich rovnou opraveny a implementovány. I přes testování aplikace však nebyly zcela jistě vyřešeny všechny problémy.

Původní myšlenka počítala se širší funkcionalitou, zejména s podporou telnetu. To by umožnilo i správu zařízení, která nepodporují SSH spojení. Implementace knihovny funkcí však nebyla řádně otestována a proto nebyla nakonec přidána do konečné aplikace. Podobná situace nastala i se systémem překladů. Při pokusu o změnu jazyka nebyly korektně zobrazeny některé řetězce, a proto se zatím podpora více jazyků odložila do další verze. Většina vytyčených cílů však byla splněna.

V budoucnu budou implementována veškerá rozšíření, která byla zmíněna v kapitole 4.11 a další funkce budou postupně přidávány. Finálním cílem je vydání celé aplikace pod opensource licencí a umožnit tak ostatním potenciálním uživatelům vytvořit si vlastní modifikaci celého řešení.

CONCLUSION

Despite the fact the entire design and development phase was accompanied by bunch of issues, the final application is able to process output from device, make a check on that output including more complex check by finite machine and show user results. The application was tested in real environment and on real devices. All found issues have been marked and some of them also immediately fixed and implemented. Although the application was tested, there still exist some other problems

The original idea counted with much wider functionality, especially with support of telnet protocol. This would allow management of devices which don't support SSH connection. The implementation of the function library was not fully tested and thus this library has not been added to final application. The similar situation happened also on translation system. During the change of the language some strings had not been correctly showed. Because of this, this functionality was put on hold and will be available in next versions. All other targets were successfully met.

In near future all extension mentioned in chapter 4.11 will be implemented and next functions progressively added. The final objective is to publish complete application under opensource license and enable to other people creation of their own system modification

SEZNAM POUŽITÉ LITERATURY

- [1] Shrubbery Networks, Inc. - RANCID. SHRUBBERY NETWORKS, Inc. *Shrubbery Networks - Unix and Network Consulting* [online]. © 1996-2006 [cit. 2013-05-13]. Dostupné z: <http://www.shrubbery.net/rancid/>
- [2] Automated Network Management: HP Official site. HP - United States | Laptop Computers, Desktops , Printers, Servers and more [online]. 2013 [cit. 2013-05-02]. Dostupné z: <http://www8.hp.com/us/en/software-solutions/software.html?compURI=1174702#tab=TAB1>
- [3] Nessus Vulnerability Scanner: Fueled by Nessus ProfessionalFeed. TENABLE. Tenable Network Security [online]. 2013 [cit. 2013-05-02]. Dostupné z: <http://www.tenable.com/products/nessus>
- [4] Cisco Prime LAN Management Solution: Products & Services. CISCO SYSTEMS, Inc. Cisco Systems, Inc. [online]. 2013 [cit. 2013-05-02]. Dostupné z: <http://www.cisco.com/en/US/products/ps11200/index.html>
- [5] Network Configuration Manager: NCM. SOLARWINDS INC. IT Management and Monitoring Software by SolarWinds [online]. 2013 [cit. 2013-05-02]. Dostupné z: <http://www.solarwinds.com/network-configuration-manager.aspx>
- [6] Network Device Configuration Reporting. TITANIA LTD. *Cyber Security Auditing Software: Titania* [online]. 2012 [cit. 2013-05-05]. Dostupné z: <https://www.titania-security.com/nipperstudio/configuration>
- [7] Libssh: The SSH Library!. *Libssh* [online]. 2013 [cit. 2013-05-13]. Dostupné z: <http://www.libssh.org/>
- [8] Source Browser. APPLE INC. *Open Source: Releases* [online]. © 2010 [cit. 2013-05-13]. Dostupné z: http://www.opensource.apple.com/source/remote_cmds/remote_cmds-22/telnet.tproj/
- [9] Apache friends: xampp. SEIDLER, Kai. *Apache friends* [online]. © 2002-2013, 01 Mar 2013 02:14:55 PM CET [cit. 2013-05-13]. Dostupné z: <http://www.apachefriends.org/en/xampp.html>
- [10] MySQL: Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. 23.2.2010, 7.3.2013 [cit. 2013-05-05]. Dostupné z: <http://cs.wikipedia.org/wiki/MySQL>

- [11] Porovnání DB Enginů dostupných v MySQL z hlediska funkčnosti. ČECÁK, Ondřej. *Čečák.cz* [online]. 2009, 12.1.2009 [cit. 2013-05-05]. Dostupné z: http://www.cecak.cz/fel/dba/referaty/mysql/porovnani_db_enginu_dostupnych_v_mysql_z_hlediska_funkcnosti_a_vykonosti
- [12] Plugins/livequery: jQuery Wiki. THE JQUERY FOUNDATION. *JQuery Wiki* [online]. © 2010 [cit. 2013-05-13]. Dostupné z: <http://docs.jquery.com/Plugins/livequery>
- [13] Apache friends: xampp for windows. *Apache friends* [online]. © 2002-2013 [cit. 2013-05-13]. Dostupné z: <http://www.apachefriends.org/en/xampp-windows.html>
- [14] *PhpMyAdmin* [online]. © 2003 - 2013 [cit. 2013-05-13]. Dostupné z: http://www.phpmyadmin.net/home_page/index.php
- [15] *Php_ssh2.dll: xampp-php-extensions* [online]. 2008 [cit. 2013-05-13]. Dostupné z: http://code.google.com/p/xampp-php-extensions/downloads/detail?name=php_ssh2.dll&can=2&q=
- [16] PHP 5.3: jak je to s VC6, VC9, Thread Safe a Non Thread Safe. VEČEŘA, Zdeněk. *Zdeněk Večeřa* [online]. 2010 [cit. 2013-05-05]. Dostupné z: <http://blog.zdenekvecera.cz/item/php-5-3-jak-je-to-s-vc6-vc9-thread-safe-a-non-thread-safe>
- [17] *Phpseclib: pure PHP implementations of SSH, SFTP, RSA and X.509* [online]. 2013 [cit. 2013-05-13]. Dostupné z: <http://phpseclib.sourceforge.net/>
- [18] THE JQUERY FOUNDATION. *JQuery UI* [online]. 2013 [cit. 2013-05-13]. Dostupné z: <http://jqueryui.com/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

RANCID	Really Awesome New Cisco confIg Differ
MTTR	Mean time to restore
IOS	Internetwork operating system
OS	Operační systém
EOL	End of Life – konec životnosti produktu
GUI	Graphical user interface
CLI	Command line interface
SCP	Secure Copy
PHP	Hypertext Preprocessor
HTML	Hypertext Markup Language
UI	User Interface
AJAX	Asynchronous JavaScript and XML
DLL	Dynamic Link Library
PECL	PHP Extension Community Library
XML	Extensible Markup Language
SO	Shared Object
DKA	Deterministický konečný automat
OSPF	Open Shortest Path First
BGP	Border Gateway Protocol
IIS	Internet Information Services
CSS	Cascade Style Sheet
URL	Uniform Resource Locator
TACACS	Terminal Access Controller Access-Control System
RADIUS	Remote Authentication Dial In User Service
NTP	Network Time Protocol
AAA	Authentication, Authorization and Accounting protocol
ISAM	Indexed Sequential Access Method
ACID	Atomicity, Consistency, Isolation, Durability
DOM	Document Object Model

SEZNAM OBRÁZKŮ

Obrázek 1: Nefunkcionální požadavky.....	20
Obrázek 2: Funkční požadavky	23
Obrázek 3: Diagram aktivit pro spuštění kontrolní šablony	24
Obrázek 4: Diagram aktivit pro spuštění konfigurační šablony	24
Obrázek 5: Diagram aktivit pro vytvoření nové šablony.....	25
Obrázek 6: Diagram aktivit pro vytvoření nového layoutu	25
Obrázek 7: ER diagram pro databázi HCTool.....	28
Obrázek 8: Kontrolní panel aplikace Xampplite	34
Obrázek 9: Okno aplikace phpmyadmin, kde je třeba vložit MySQL skript.	35
Obrázek 10: Architektura systému cisco IOS.....	43
Obrázek 11: Okno pro vytvoření a editaci nové šablony.....	51
Obrázek 12: Prázdný layout.....	54
Obrázek 13: Vyplněný layout	55
Obrázek 14: Okno pro vytvoření nového bloku	56
Obrázek 15: DKA pro sekční příkaz.....	59

SEZNAM TABULEK

Tabulka 1: Seznam MySQL tabulek.....	29
Tabulka 2: Seznam složek a popis jejich obsahu.....	39
Tabulka 3: Soupis funkcí pro generování HTML výstupu	39
Tabulka 4: Soupis systémových funkcí	40
Tabulka 5: Soupis funkcí databázové třídy.....	41
Tabulka 6: Soupis funkcí třídy Error	42
Tabulka 7: Soupis funkcí třídy HTML elementů.....	42
Tabulka 8: Seznam funkcí třídy SSH	45
Tabulka 9: Seznam funkcí třídy Telnet.....	45
Tabulka 10: Mapování aplikačních práv na sloupce v DB tabulce	47
Tabulka 11: Mapování znaků na tokeny	61

SEZNAM PŘÍLOH

P I ČÁST VÝSTUPU KONTROLNÍHO SKRIPTU

P II ČÁST VÝSTUPU SEKČNÍHO SKRIPTU

PŘÍLOHA I – ČÁST VÝSTUP KONTROLNÍHO SKRIPTU

CZ-PRG-SWA-01	
Allianz HC	
Stack Info	switch 1 priority switch 2 priority switch 3 priority switch 4 priority
Device name	hostname cz-prg-swa-01
QOS settings	mls qos map cos-dscp 0 8 16 24 32 40 48 56 mls qos sr-queue output cos-map queue 1 threshold 3 4 5 mls qos sr-queue output cos-map queue 2 threshold 1 2 mls qos sr-queue output cos-map queue 2 threshold 2 3 mls qos sr-queue output cos-map queue 2 threshold 3 6 7 mls qos sr-queue output cos-map queue 3 threshold 3 0 mls qos sr-queue output cos-map queue 4 threshold 3 1 mls qos sr-queue output dscp-map queue 1 threshold 3 32 33 40 41 42 43 44 45 mls qos sr-queue output dscp-map queue 1 threshold 3 46 47 mls qos sr-queue output dscp-map queue 2 threshold 1 16 17 18 19 20 21 22 23 mls qos sr-queue output dscp-map queue 2 threshold 1 26 27 28 29 30 31 34 35 mls qos sr-queue output dscp-map queue 2 threshold 1 36 37 38 39 mls qos sr-queue output dscp-map queue 2 threshold 2 24 mls qos sr-queue output dscp-map queue 2 threshold 3 48 49 50 51 52 53 54 55 mls qos sr-queue output dscp-map queue 2 threshold 3 56 57 58 59 60 61 62 63 mls qos sr-queue output dscp-map queue 3 threshold 3 0 1 2 3 4 5 6 7 mls qos sr-queue output dscp-map queue 4 threshold 1 8 9 11 13 15 mls qos sr-queue output dscp-map queue 4 threshold 2 10 12 14 mls qos queue-set output 1 threshold 1 1000 1000 100 1000 mls qos queue-set output 1 threshold 2 1000 1000 100 1000 mls qos queue-set output 1 threshold 3 1000 1000 100 1000 mls qos queue-set output 1 threshold 4 1000 1000 100 1000 mls qos queues-set output 1 buffers 15 20 55 1 mls qos
AAA config	aaa authentication login default group TACPLUS local aaa authentication enable default group TACPLUS enable aaa authorization console aaa authorization exec default group TACPLUS if-authenticated aaa authorization commands 1 default group TACPLUS if-authenticated aaa authorization commands 15 default group TACPLUS if-authenticated

PŘÍLOHA II – ČÁST VÝSTUPU SEKČNÍHO SKRIPTU

PORTS	
Loopback0	ip address ..
	description
	switchport mode
	switchport access vlan
	switchport voice vlan
	speed 100
	duplex full
	rr-queue bandwidth share
	priority-queue out
	no snmp trap link-status
	spanning-tree portfast
	spanning-tree bpduguard enable
	spanning-tree guard loop
	switchport port-security
	ip dhcp snooping limit rate 15
Loopback1	ip address
	description
	switchport mode
	switchport access vlan
	switchport voice vlan
	speed 100
	duplex full
	rr-queue bandwidth share
	priority-queue out
	no snmp trap link-status
	spanning-tree portfast
	spanning-tree bpduguard enable
	spanning-tree guard loop
	switchport port-security
	ip dhcp snooping limit rate 15
FastEthernet0/1	ip address ..
	description
	switchport mode access
	switchport access vlan 100
	switchport voice vlan
	speed 100
	duplex full
	rr-queue bandwidth share
	priority-queue out
	no snmp trap link-status
spanning-tree portfast	
spanning-tree bpduguard enable	