VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

Ostrava 2010 Bc. Kateřina Bambušková

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozvoj řídícího software Distribuované virtuální laboratoře počítačových sítí

Enhancements of Distributed Virtual Laboratory Control Software

Ostrava 2010 Bc. Kateřina Bambušková

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Kateřina Bambušková**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Rozvoj řídícího software Distribuované virtuální laboratoře počítačových sítí
Enhancements of Distributed Virtual Laboratory Control Software

Zásady pro vypracování:

Smyslem práce je implementovat rozšíření systému Distribuované virtuální laboratoře počítačových sítí.

1. Implementujte možnost specifikace zadání úloh ve více jazycích a jejich prezentaci podle jazyka nastaveného v preferencích přihlášeného uživatele.
2. Navrhněte mechanismus a implementujte možnost spolupráce uživatelů z více lokalit na společně rezervované úloze.
3. Implementujte možnost předkonfigurace laboratorních prvků (včetně varianty předkonfigurovaného prvku studentům nepřístupného).
4. Implementujte pomocné utility pro podporu administrace systému.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Grygárek, Ph.D.**

Datum zadání: 30.11.2008

Datum odevzdání: 07.05.2010

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr.Ing. Eduard Sojka
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Poděkování

Na tomto místě bych ráda poděkovala všem, kteří mi pomáhali, a se kterými jsem mohla spolupracovat. Poděkování patří celému vývojovému i provoznímu týmu Virtlabu.

Především pak koordinátorovi Virtlabového týmu a mému vedoucímu diplomové práce Ing. Petru Grygárkovi, Ph.D. za jeho obrovskou podporu a trpělivost.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně.

Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě dne ..... ……

Bc. Kateřina Bambušková

Abstrakt a klíčová slova

Abstrakt:

Hlavním cílem této práce je seznámit čtenáře se systémem Distribuované virtuální laboratoře počítačových sítí, včetně popisu návrhu a implemence rozšíření jeho řídícího softwaru dle zadání. Práce se postupně zabývá návrhem a implementací těchto rozšíření: podpora specifikace zadání laboratorních úloh ve více jazycích, rozšíření řídící aplikace o možnost spolupráce uživatelů z různých lokalit, návrhem a implementací systému pro předkonfigurace laboratorního vybavení a různých podpůrných utilit pro administraci systému.

Klíčová slova:

Virtlab, Virtuální laboratoř, počítačové sítě, PHP, SOAP, VirtIS, Generátor konfigurací.

Abstract and keywords

Abstract:

The main goal of this thesis is to introduce Distributed Virtual Laboratory of computer networks, to design and implement extensions of its control software according to requirements. This thesis designs and implements the following: Multilingual support for laboratory tasks, extension for the control applications on the possibility of collaboration of users from different sites, a system for pre-configuration laboratory equipment and various support utilities for system administration.

Keywords:

Virtlab, Virtual laboratory, computer networks, PHP, SOAP, VirtIS, Configs generator.

Seznam použitých symbolů a zkratek

ASSSK – Automatizovaný systém správy síťových konfigurací
CSS – Cascading Style Sheets – kaskádové styly
ERD – Entity Relationship diagram – diagram entit a jejich vztahů
HTML – HyperText Markup Language – značkovací jazyk pro tvorbu internetových stránek
HTTP – Hypertext Transfer Protocol – Iinternetový protokol určený pro výměnu dokumentů
PHP – Personal Home Page – Hypertext Preprocesor – skriptovací jazyk
SOAP – Simple Object Access Protocol – Protokol pro výměnu zpráv po síti
Virtlab – Virtuální laboratoř počítačových sítí
VirtIS – Informační systém Virtlabu
XHTML – Extensible HyperText Markup Language – rozšiřitelný značkovací jazyk pro hypertext
XML – Extensible Markup Language – Obecný značkovací jazyk

Obsah

1. Úvod

Původní projekt Virtuální síťové laboratoře – Virtlab (viz obrázek 1) vznikl před více než pěti lety a základní koncepce celého systému byla definována v roce 2005 v diplomové práci Pavla Němce [1].

Postupem času pak tento projekt získával čím dál víc na své oblíbenosti a popularitě. Až do dnešních dnů, ho postupně vylepšovalo několik desítek vývojářů ve formě diplomových a bakalářských prací nebo v rámci různých semestrálních projektů, případně jako svou volnočasovou aktivitu.

V posledních letech je plně v provozu distribuovaná verze Virtlabu a současně se aktivně používá při výuce zaměřené na počítačové sítě. O celý tento projekt pak projevilo zájem i několik dalších institucí včetně zahraničních.

Pouze „mrtvý projekt“ se dále nevyvíjí, nevylepšuje a nerozšiřuje, a protože Virtlab k „mrtvým projektům“ rozhodně nepatří, je cílem této práce dále rozvíjet řídící software Distribuované virtuální laboratoře počítačových sítí.

V následujícím textu jsou shrnuty nejpodstatnější informace o projektu Virtlab, dále je popsána základní architektura Virtlabu a popis jednotlivých komponent. Zájemce o hlubší seznámení a detailnější popis jednotlivých komponent odkazuji na oficiální stránky projektu http://www.virtlab.cz [2].

obrázek 1: logo projektu Virtlab

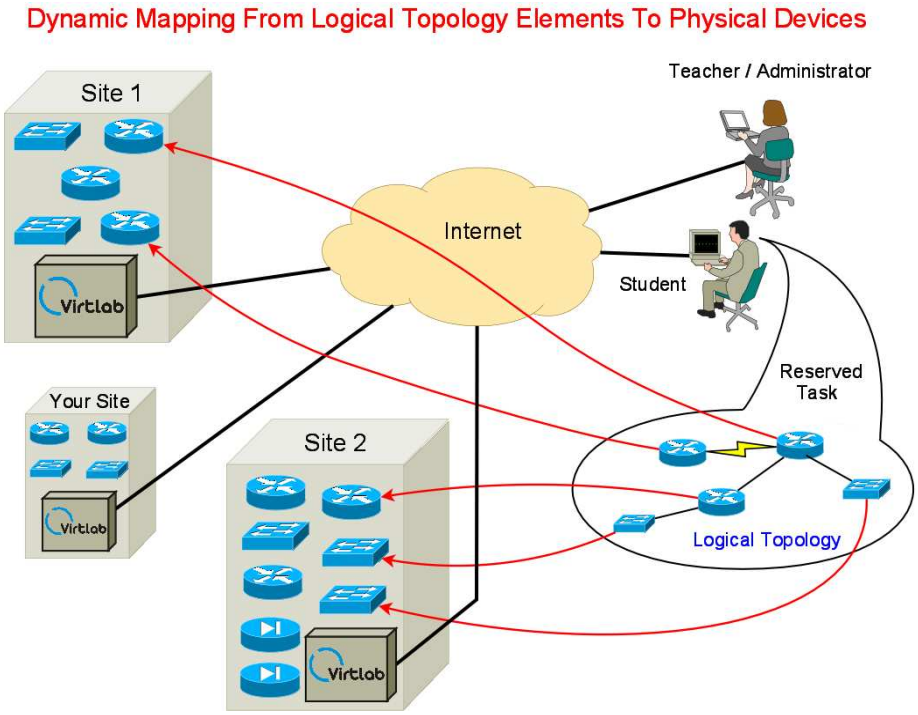
1.1. O virtuální laboratoři počítačových sítí

Hlavním smyslem celého projektu virtuální laboratoře je vzdáleně (přes Internet) zpřístupnit studentům poměrně drahé síťové prvky (laboratorní zařízení), převážně pro praktickou výuku zaměřenou na počítačové sítě. Studenti si tak mohou přes webové rozhraní řídící aplikace zarezervovat laboratorní zařízení na jimi definovanou dobu, a poté k jednotlivým síťovým prvkům přistupovat přes svůj webový prohlížeč.

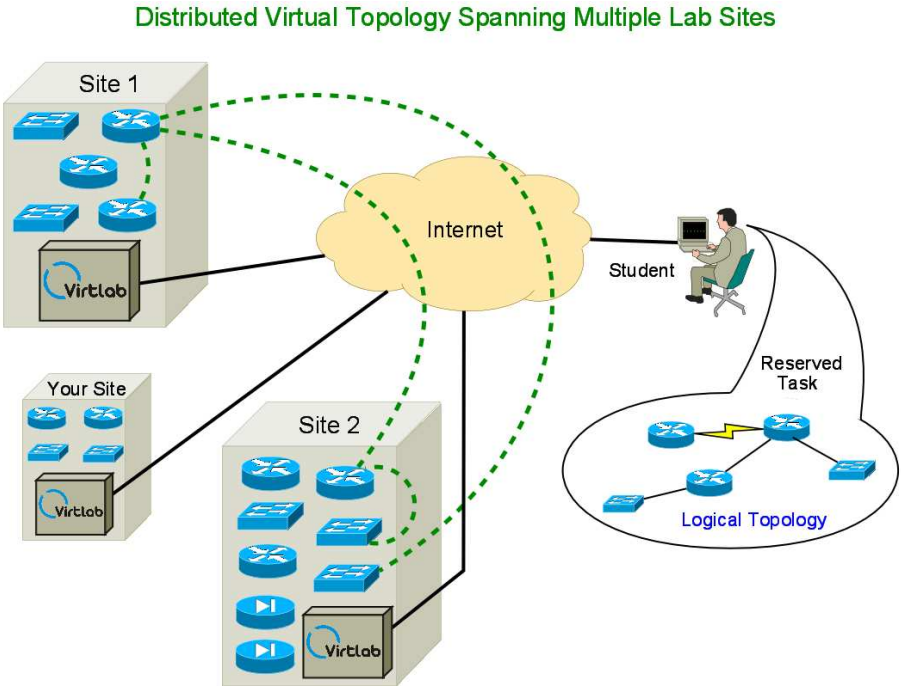
Při výběru konkrétní úlohy nebo vlastní topologie se výběr požadovaného zařízení provádí automaticky (viz obrázek 2), a to z dostupného vybavení, které se může nacházet v různých lokalitách (viz obrázek 3).

Následné propojení požadované topologie se provádí plně automatizovaně. Pokud se jedná o tzv. „školní úlohu“, mohou si studenti svá řešení nechat otestovat systémem pro

automatické hodnocení konfigurací, které implementoval ve své diplomové práci Zdeněk Filipec [3].



obrázek 2: dynamické mapovaní logické topologie na fyzická zařízení



obrázek 3: distribuovaná topologie mezi lokalitami

1.2. Architektura virtuální laboratoře

Distribuovaná verze Virtlabu se může skládat z několika samostatně funkčních virtuálních laboratoří, které pak nazýváme lokality. Každá taková lokalita musí obsahovat hlavní Virtlab Server, který se skládá z jednotlivých komponent (řídící server, rezervační server, tunelovací server, konzolový server, mazací server, databáze, a dalších systémových utilit) a je připojen k Internetu nebo do interní sítě, pokud nechceme, aby byl součásti veřejného Internetu.

Jednotlivé komponenty, jako jsou konzolový server nebo tunelovací server, se mohou nacházet na různých fyzických strojích (serverech), ale vzhledem k jejich nenáročnosti na výpočetní výkon se mohou nacházet na jednom fyzickém stroji (serveru).

Pokud lokalita obsahuje laboratorní vybavení, musí být každé zařízení připojeno alespoň k jednomu ze dvou propojovacích prvků: ASSSK nebo VLMUX.

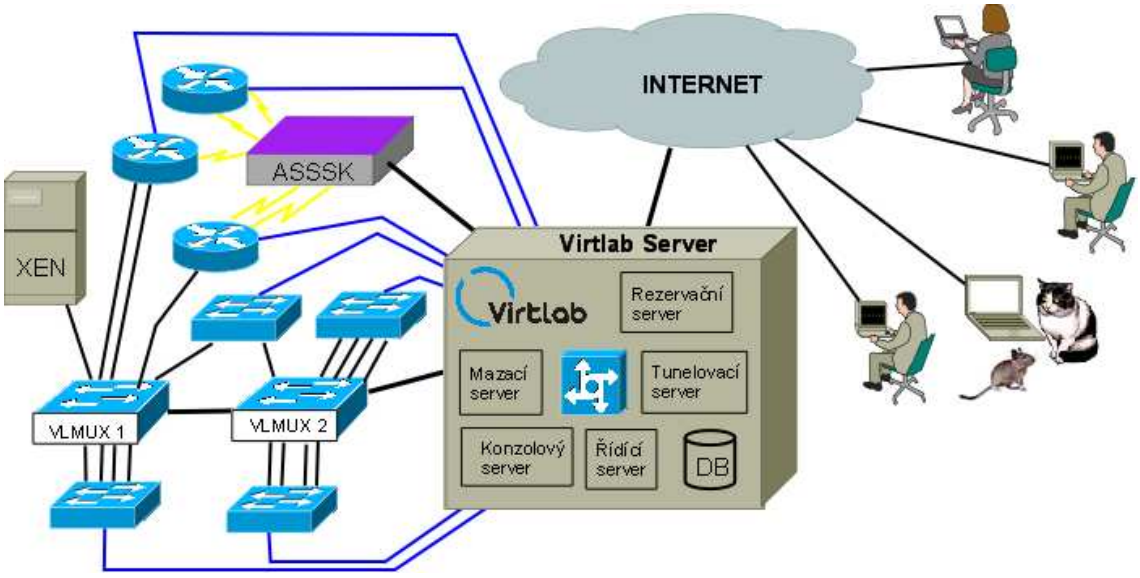
ASSSK (neboli automatizovaný systém správy síťových konfigurací) je označení pro vlastní prototypy zařízení pro automatizované spojování dvojic WAN portů směrovačů na bázi různých typů spojovacích polí (analogové, digitální s použitím FPGA). Ovládání je realizováno jednoduchým příkazovým jazykem z řídící konzole připojené přes rozhraní RS-232.

VLMUX je pak označení pro spojovací přepínače podporující technologii QinQ postavenou na normě 802.1Q, které zajišťují propojení zařízení přes jejich ethernet porty.

Ke konzolovému serveru je třeba přes rozhraní RS-232 připojit jednotlivé řídící konzole laboratorního vybavení. Vzhledem k většímu počtu zařízení, z důvodu nízkého počtu sériových rozhraní na standardních základních deskách serverů, jsou jednotlivá zařízení připojena přes multiportové sériové komunikační karty od společnosti MOXA.

XEN – virtualizační server, na kterém je provozována paravirtualizace XEN a tvoří sadu instancí virtuálních počítačů, které nahrazují fyzické PC.

Architektura Virtlabu je znázorněna na následujícím obrázku (viz obrázek 4).



obrázek 4: architektura Virtlabu

1.2.1. Řídící server

Řídícím serverem nazýváme především webové rozhraní, které slouží pro práci uživatelů se systémem Virtlab. Je implementován v PHP a slouží ke správě uživatelů, jejich práv a zařazení do skupin, dále ke správě úloh, souborů, rezervací a umožňuje přístup k podpůrným nástrojům, jako jsou: validace XML, monitor serverů, import úloh a dalších.

Při rezervaci úlohy nebo topologie, komunikuje řídící server s rezervačním serverem, kterého se ptá na seznam dostupného laboratorního vybavení a v době aktivní rezervace poskytuje rozhraní pro navázání spojení s konzolovým serverem.

Součástí řídícího serveru je rozsáhlá databáze dvldb-www obsahující velké množství tabulek. První verze řídícího serveru, pro distribuovanou variantu Virtlabu, byla navržena a implementována v diplomové práci Jana Vavříčka [4].

1.2.2. Rezervační server

Rezervační server reaguje na požadavky z řídícího serveru nebo na požadavky přicházející z rezervačních serverů jiných lokalit. Na základě dostupného laboratorního vybavení, v rámci časového rozvrhu ve své lokalitě, posílá seznam dostupného laboratorního vybavení ve formátu XML.

Rezervační server, který se nachází v lokalitě, ze které vzešel požadavek, jednotlivé seznamy poskládá a vrátí řídícímu serveru. V případě úspěšné rezervace si ve své databázi dvldb-reser uloží informaci o použitém vybavení.

Pokud z řídícího serveru přijde požadavek na zrušení rezervace, opět musí rezervační server navázat spojení na rezervační servery z ostatních lokalit.

1.2.3. Tunelovací server

Tunelovací server je komponenta, na kterou se kladou největší nároky ohledně rychlosti a spolehlivosti. Jeho úkolem je zajistit propojení různých VLAN mezi jednotlivými lokálními virtuálními laboratořemi a také různých VLAN v rámci jednoho ethernetového segmentu.

Fyzický server, na kterém běží tunelovací server, musí mít dvě ethernet rozhraní, jedno je připojeno k Internetu a druhé je připojeno trunkem k propojovacímu prvku VLMUX.

Při aktivaci rezervace dojde k nakonfigurování tabulky (tabulka přesměrování), ve které je uvedeno vždy VLAN ID rámce z lokání sítě, nové VLAN ID kam bude rámec poslán a ip adresa vzdáleného (nebo i lokálního) tunelovacího serveru.

Tunelovací server na trunk linkce odchytává VLAN tagované ethernetové rámce a kontroluje, jestli jsou definovány v tabulce přesměrování a podle ní je přečíslovává a posílá zpět s jiným číslem VLAN nebo přes Internet na IP adresu odpovídajícího tunelovacího serveru. Tímto způsobem dochází k virtuálnímu propojení ethernetu – ethernetový tunel, a to i vzdáleně (přes Internet).

Pokud je v dané lokalitě i ASSSK, pak je opět připojen k tomuto serveru, a při aktivaci rezervace mu tunelovací server posílá informace o propojení sériových WAN portů.

1.2.4. Konzolový server

Úkolem konzolového serveru je zprostředkovat uživatelům přístup k sériovým konzolím síťových zařízení nebo telnetové připojení na zařízení. Pro zobrazení komunikace, probíhající prostřednictvím jednoduchého protokolu nad TCP/IP, se využívá Java Applet, který běží ve webovém ovládacím rozhraní, což umožňuje uživateli jednoduchý přístup k síťovým zařízením.

Pro přístup na zařízení ve vzdálených lokalitách, slouží konzolový server zároveň jako proxy server, který zprostředkovává přístup k zařízením, která jsou připojena ke vzdáleným konzolovým serverům, kam nemá lokální uživatel přímý přístup.

Prostřednictvím speciálního PHP skriptu verify.php se ověřuje, jestli jsou požadavky uživatele oprávněné a tím zamezuje neautorizovaný přístup na zařízení.

Dalším bezpečnostním opatřením je kontrola vstupu ze strany uživatele a následné testování, zda se nejedná o některý ze zakázaných příkazů. Některé příkazy, jako jsou třeba různá hesla pro přístup na zařízení, nebo ukládání konfigurace by mohli způsobit, že se na zařízení nedostanou další uživatelé. Konzolový server má veškeré zakázané příkazy uloženy ve své databázi cmd_disabled.

1.2.5. Mazací server

Mazací server slouží k přípravě laboratorního zařízení do výchozího stavu, tak aby bylo připraveno pro další uživatele, kteří si dané zařízení rezervovali po předchozí rezervaci.

K této akci dochází na konci stávající rezervace, kdy už uživatelé nemají na laboratorní zařízení přístup, ale zařízení je na ně ještě rezervováno. Impuls k mazací akci dostává server od systémového nástroje at démona (obdoba nástroje CRON).

V současnosti se používají dvě metody k dosažení výchozího stavu laboratorního zařízení:

1) **Odeslání mazací sekvence** – pomocí vhodných příkazů odeslaných na zařízení, se toto zařízení restartuje, a po nabootování se nachází ve výchozím stavu.
2) **Hardwarový restart** – některá zařízení jsou připojena ke zdroji elektrické energie přes speciální vzdáleně ovladatelnou napájecí lištu – APC Power Switch. V případě, že se zařízení vzdáleně odpojí od zdroje elektrické energie, dojde k jeho vypnutí a při následném znovu nabootování se opět nachází ve výchozím stavu.

1.3. Specifikace požadavků na rozšíření virtuální laboratoře

Zkušenosti z pilotního provozu distribuované virtuální laboratoře počítačových sítí ukázaly potřebu rozšíření o řadu dalších funkcí a vylepšení. Následuje seznam jednotlivých požadavků na rozšíření virtuální laboratoře, které plynou se zadání.

1.3.1. Zadání úloh ve více jazycích

Jedná se o potřebu mít možnost specifikovat zadání k jednotlivým úlohám ve více jazycích a jejich zobrazení uživatelům podle preferovaného jazyka. Tento požadavek vznikl z důvodu neustále narůstajícího počtu zahraničních studentů.

Podrobně se touto problematikou zabývá kapitola Zadání úloh ve více jazycích, kterou můžete najít na straně 12.

1.3.2. Spolupráce uživatelů z více lokalit

Systém virtuální laboratoře umožňuje spolupráci více uživatelů na jedné zarezervované úloze v rámci jedné lokality, ale časem se ukázalo, že chybí možnost spolupráce uživatelů z různých lokalit. Tato funkčnost by mohla být zajímavá při řešení složitějších úloh, kdy spolu s uživateli jedné lokality by mohli na úloze spolupracovat i zkušenější uživatelé z jiných lokalit.

Postup při řešení tohoto problému naleznete v kapitole Spolupráce uživatelů z více lokalit na straně 16.

1.3.3. Předkonfigurace laboratorních prvků

Jako další velmi užitečná funkce se ukázala možnost mít některé laboratorní prvky předem nakonfigurované, nebo alespoň částečně předkonfigurované. Tato funkčnost byla plánována převážně pro prvky studentům nepřístupné, případně pro složitější úlohy, kde je cílem procvičit specifičtější problematiku a použitím částečné předkonfigurace ušetřit čas při základní konfiguraci.

Více o této problematice můžete najít v kapitole Předkonfigurace na straně 21.

1.3.4. Archív uživatelských konfigurací

Některé školní úlohy vyžadují odevzdávání po částech, proto na základě požadavků uživatelů, vznikl nápad vytvořit archív uživatelských konfigurací. Uživatelé by totiž uvítali možnost pokračovat při opětovné rezervaci úlohy, v konfiguraci laboratorního zařízení tam, kde naposledy skončili. Konkrétní řešení je popsáno v kapitole Archív uživatelských konfigurací na straně 26.

1.3.5. Pomocné utility a další rozšíření

V neposlední řadě by se hodila nějaká utilita, pro správu mnoha konfiguračních souborů jednotlivých komponent z různých lokalit a dalších informací, včetně seznamu laboratorních prvků v jednotlivých lokalitách. Tento problém, s velkým množstvím konfigurací, byl vyřešen návrhem generátoru konfigurací. Více o návrhu a implementaci tohoto systému se dozvíte na straně 31.

Během provozu se dále objevila potřeba hlásit a zaznamenávat chyby, případně další návrhy na zlepšení ze strany uživatelů, kteří se systémem pracují. Proto byl původní informační systém pro Virtlab – VirtIS rozšířen o bug tracking systém. Více o integraci najdete na straně 37.

2. Zadání úloh ve více jazycích

Při posledním návrhu webového rozhraní Distribuované virtuální laboratoře počítačových sítí se počítalo s podporou jazykových mutací, ale pouze v rámci GUI řídícího systému. Bohužel se v tomto návrhu nebrala v úvahu možnost specifikovat alternativní jazykové verze zadaní pro laboratorní úlohy. Veškeré informace o původním návrhu řídící aplikace můžete najít v diplomové práci Jana Vavříčka [4].

Vzhledem k neustálému nárůstu počtu zahraničních studentů a také vzrůstajícímu zájmu dalších univerzit, se velmi záhy ukázalo, že je nutné tento nedostatek odstranit a rozšířit stávající systém i o možnost specifikovat zadání jednotlivých úloh ve více jazycích.

V další části textu je podrobněji popsána analýza stávajícího systému včetně návrhu, dále pak samotná implementace navrženého řešení a v neposlední řadě také otestování funkčnosti.

2.1. Analýza stávajícího systému

Řídící systém virtuální laboratoře počítačových sítí je vybaven podporou pro zobrazení GUI v jednotlivých jazykových mutacích. Toho je docíleno využitím tabulky `language` v databázi řídícího serveru, která obsahuje jednotlivé jazyky. Společně s třídou, která zajišťuje mapování mezi jednotlivými jazyky a odpovídajícími texty, které se mapují podle jednoznačného identifikátoru a uživatelem zvoleného jazyka.

Každý uživatel si pak může ve svém osobním nastavení zvolit, jaký jazyk se má pro zobrazení používat. V současné době se využívá pouze jazyka českého a anglického, ale velmi snadno můžeme přidat další jazykové verze.

Ovšem správce úloh má v původním systému možnost specifikovat pouze jeden soubor se zadáním pro konkrétní úlohu, tudíž tu není žádný prostor pro možnost jednotlivých zadání v různých jazycích.

2.2. Návrh řešení

Velmi jednoduchým řešením je rozšíření databáze řídícího systému o vazební tabulku. V tabulce se budou vázat jednotlivá zadání s alternativními jazyky definovanými v rámci systému. Díky tomuto rozšíření je možno velmi snadno definovat ke každé úloze několik zadání v různých jazycích. Úloha přitom nemusí obsahovat všechna zadání ve všech systémem definovaných jazycích.

Kromě rozšíření databáze bude nutno změnit současný způsob zobrazování úloh, stejně jako způsob jejich vytváření a editace. Upravit se bude muset i původní způsob importu a exportu celých úloh, kterým systém disponuje.

Jednotlivé úpravy jsou podrobněji rozepsány v jednotlivých podkapitolách.

2.2.1. Rozšíření databáze

Jediným rozšířením databáze řídícího serveru bude tedy vazební tabulka, která má velmi jednoduchou strukturu, neboť nám stačí pouze tyto tři atributy: jednoznačný identifikátor úlohy, jednoznačný identifikátor jazykové verze, jednoznačný identifikátor souboru se zadáním.

Všechny tyto atributy jsou povinné a jsou zároveň cizími klíči na konkrétní již existující tabulky s úlohami, jazyky a soubory.

Konkrétní SQL skript pro vytvoření nové vazební tabulky pro zadání v různých jazycích (viz Příloha A).

2.2.2. Rozšíření Relax schématu pro import úloh

Jak jsme se již zmínili dříve, ve stávajícím systému existuje funkce pro import úloh, která umožňuje velmi elegantně celou úlohu se všemi potřebnými soubory umístěnými ve společném archívu importovat do systému. Vzhledem k tomu, že jsme upravili způsob, jakým budeme k úlohám definovat soubory se zadáním, musíme upravit i způsob importování úloh.

Specifikace jednotlivých souborů náležejících k dané úloze se provádí pomocí XML souboru s popisem úlohy – task.xml. Tento úlohu specifikující XML soubor musí být validní podle Relax schématu taskupload.rng (viz Příloha B), který byl upraven následovně:

1) přidání elementu specifikujícího několik souborů se zadáním, přitom úloha nemusí obsahovat žádné zadání:

```
<!-- Popis zadani -->
<element name="specification">
     <zeroOrMore>
          <ref name="file" />
     </zeroOrMore>
</element>
```

2) rozšíření elementu **file** specifikujícího soubory o nepovinný atribut **lang** definující konkrétní jazyk:

```
<optional>
     <attribute name="lang">
          <data type="string">
               <param name="minLength">1</param>
               <param name="maxLength">5</param>
          </data>
     </attribute>
</optional>
```

2.3. Implementace

Samotná implementace spočívá v upravení systému řídícího serveru v PHP. Je především třeba upravit veškeré výpisy, které zobrazují úlohy a jejich zadání, jako je například detail úlohy, její editace nebo vytváření úlohy nové. Dále je třeba upravit zobrazování aktivní zarezervované úlohy, kde se rovněž zobrazuje zadání k úloze.

Vzhledem k tomu, že úloha může obsahovat pouze zadání, které se neshoduje s preferovaným jazykem uživatele, zvolili jsme následující řešení: při zobrazování detailu úlohy jsou k dispozici všechny dostupné jazykové mutace zadání (viz obrázek 5) a uživatel si tak může vybrat nejvíce mu vyhovující jazykovou verzi.

obrázek 5: jednotlivá zadání v různých jazycích

Správce úloh má během vytváření nebo editace ve formuláři předem připravená políčka podle systémem podporovaných jazyků (viz obrázek 6).

obrázek 6: editace zadání

Kromě změn při novém způsobu zobrazování jednotlivých variant zadání je třeba upravit i funkční části kódu, které se starají o ukládání dat v databázi nebo funkční kód pro import úlohy jako celku. Konkrétními implementačními úpravami se nebudeme dále zabývat.

2.4. Testování

Testování při konkrétní implementaci probíhalo průběžně, tak jak se postupně implementovaly jednotlivé funkce systému, a spočívalo především ve vizuální kontrole očekávaného chování. Samotné testování by se pak dalo rozdělit na tři části:

1) kontrola správného importu laboratorní úlohy jako celku,
2) kontrola správného vytváření a editace laboratorní úlohy v rámci systému,
3) kontrola správného zobrazování jednotlivých jazykových verzí zadání.

Po dokončení všech implementačních úprav a následným kompletním otestováním celé nové verze systému, byla tato verze nasazena na všechny produkční lokality virtuální laboratoře počítačových sítí.

3. Spolupráce uživatelů z více lokalit

Distribuovaný charakter Virtlabu umožňuje jednotlivým lokalitám mezi sebou sdílet své laboratorní vybavení, ale zároveň mohou lokality pracovat plně samostatně a nezávisle na ostatních.

Každá z lokalit má své uživatele a úlohy, ale pouze laboratorní vybavení může sdílet s ostatními lokalitami. To vedlo k tomu, že jsme přistoupili na rozšíření systému o spolupráci uživatelů z více lokalit na jedné zarezervované úloze.

Díky rozšíření systému o spolupráci uživatelů z různých lokalit můžeme sdílet i jejich laboratorní úlohy, které jsou mnohdy velmi hodnotné a stály nemalé úsilí několika vývojářů a diplomantů. Jedná se především o bakalářské práce Karla Zapletala [5], Miroslava Soludujeva [6] a Zbyňka Saloně [7].

Postup při analýze a návrhu řešení je popsán v následujících podkapitolách.

3.1. Analýza

Původní systém, tak jak byl navržen Janem Vavříčkem v jeho diplomové práci [4], obsahuje pouze možnost spolupráce uživatelů v rámci jedné lokality. Po zarezervování úlohy má uživatel možnost k úloze přidat spolupracující kolegy, ti jsou pak uloženi ve vazební tabulce ke konkrétní rezervaci v databázi řídící aplikace a na základě těchto záznamů mají k dané rezervaci v době její aktivace přístup.

Tvůrce rezervace stejně jako všichni spolupracující kolegové čerpají ze své kvóty v délce trvání rezervace. Výši limitu kvóty, která je přidělena v rámci týdenního plovoucího okna, určuje uživatel v roli administrátora systému. Každý ze spolupracujících kolegů může od rezervace kdykoliv odstoupit a ušetřit tak čerpání ze své přidělené kvóty.

3.2. Návrh řešení

Abychom mohli zrealizovat spolupráci uživatelů i z různých lokalit, potřebujeme mít nějakou možnost komunikovat mezi jednotlivými lokalitami. Celá koncepce a architektura virtuální laboratoře je navržena tak, aby jednotlivé lokality mohly fungovat nezávisle na sobě, například i v době výpadku kterékoliv z lokalit. Z tohoto důvodu není možno použít žádné centralizované řešení a je tak potřeba přímou komunikací získat z ostatních lokalit seznamy jejich uživatelů.

Jako další krok bude třeba rozšířit databázi řídícího serveru o tabulku vzdálených spolupracujících uživatelů současně s úpravou a rozšířením celého původního systému, který měl na starosti veškerou funkcionalitu ohledně spolupráce uživatelů na zarezervované úloze.

3.2.1. O technologii SOAP

SOAP - celým názvem Simple Object Access Protocol - je protokol založený na XML, pro výměnu zpráv mezi aplikacemi po síti. Technologie SOAP tvoří základní vrstvu komunikace mezi webovými službami a poskytuje prostředí pro tvorbu složitější komunikace.

Koncept webových služeb umožňuje protokolem HTTP volat jednotlivé funkce na vzdáleném serveru, předávat jim různé parametry a pracovat s vrácenými výsledky. Je tedy úplně jedno, v jakém programovacím jazyce je webová služba napsaná i z jakého jazyka ji voláme.

Díky existujícím rozšířením se v PHP s webovými službami pracuje velmi pohodlně, jsme totiž zcela odstíněni od přenosového formátu a jednoduše voláme funkce a zpracováváme výsledky podobně, jako to děláme s lokálními knihovnami.

Existuje několik různých druhů šablon pro komunikaci na protokolu SOAP. Nejznámější z nich je RPC šablona, kde jeden z účastníků komunikace je klient a na druhé straně je server, který odpovídá na požadavky klienta. Tato šablona je vhodná i pro náš případ, vzhledem k tomu, že všechny lokality se budou dotazovat na seznamy uživatelů a zároveň budou jednotlivé lokality na tyto dotazy odpovídat, je třeba, aby každá z lokalit disponovala SOAP serverem i SOAP klientem.

3.2.2. Virtlab SOAP server

Jak již bylo zmíněno výše, je třeba, aby každá lokalita obsahovala SOAP server, který bude odpovídat na požadavky SOAP klientů z ostatních lokalit. Tento SOAP server pak musí obsahovat veškeré funkce pro obsluhu jednotlivých klientů. Mezi základní požadované funkce patři především:

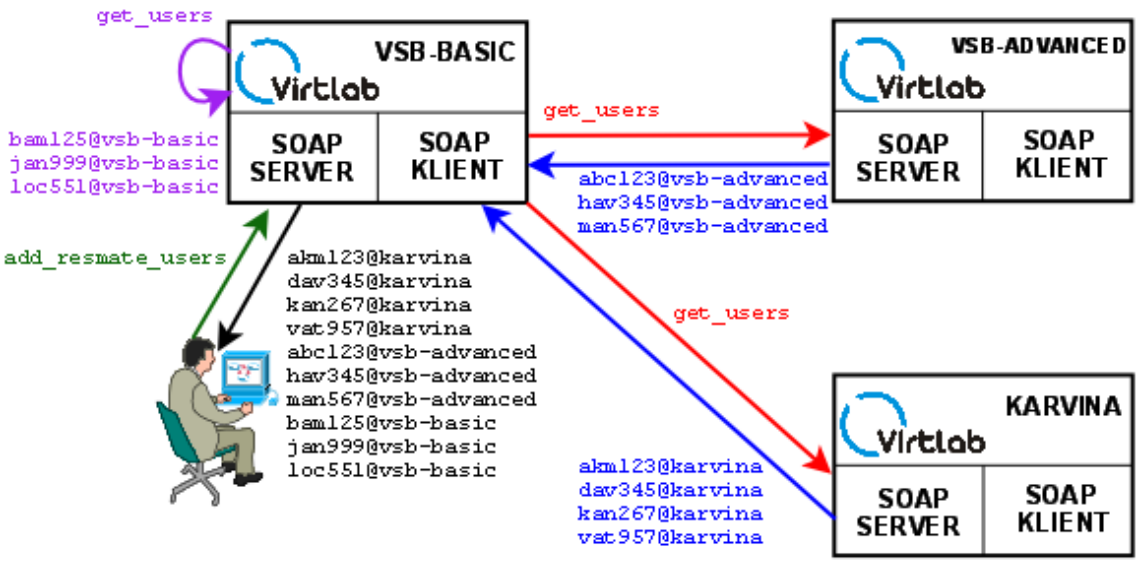
- seznam uživatelů
- ověřování uživatelů
- získání vzdálených rezervací
- získání spolupracujících kolegů
- poskytnutí údajů pro aktivní rezervace včetně detailu úlohy
- odstoupení od vzdálené rezervace

SOAP server je univerzální komponentou systému a postupně ji bude možno velmi snadno rozšířit o další funkce pro vzdálenou komunikaci mezi jednotlivými lokalitami.

3.2.3. Virtlab SOAP klient

Druhou komponentou do páru je SOAP klient, který se podle potřeby doptává SOAP serverů jednotlivých lokalit, na výše uvedené požadavky, a dále může získané informace na straně domovské lokality zpracovávat.

Komunikace SOAP klienta s jednotlivými SOAP servery při požadavku na přidání spolupracujících kolegů je znázorněna na následujícím obrázku (viz obrázek 7).



obrázek 7: komunikace SOAP klienta se SOAP servery

Na obrázku je znázorněna situace, při které uživatel z lokality VSB-BASIC přidává spolupracující kolegy pomocí funkce add_resmate_user. Lokálně se zavolá funkce get_users, která vrátí seznam uživatelů v lokatitě VSB-BASIC. V dalším kroku SOAP klient z lokality VSB-BASIC odesílá požadavek get_users na SOAP servery vzdálených lokalit. Na tento požadavek SOAP servery odesílají své seznamy uživatelů. Seznamy ze všech lokalit se následně zobrazí uživateli. Ten z nich pak může vybrat své spolupracující kolegy.

3.2.4. Rozšíření databáze

Pro zaznamenávání vzdáleně spolupracujících uživatelů vytvoříme novou vazební tabulku `reservation_remote_users` v databázi řídícího serveru `dvldb-www`. Tato tabulka vychází z původní tabulky spolupracujících uživatelů `reservation_users` a je rozšířena o jednoznačný identifikátor sítě.

Konkrétní SQL skript pro vytvoření nové vazební tabulky pro vzdálené uživatele (viz Příloha A).

3.3. Implementace

Pro implementaci SOAP serveru a SOAP klienta, v programovacím jazyce PHP v rámci řídícího serveru virtuální laboratoře počítačových sítí, bylo využito PHP dokumentace viz http://www.php.net/manual/en/book.soap.php [8].

3.3.1. Implementace SOAP serveru

Celá implementace Virtlab SOAP serveru se nachází v souboru **soapserver.php**, který obsahuje deklaraci objektu $server typu SoapServer a jednotlivé funkce, kterými bude obsluhovat jednotlivé SOAP klienty. Veškeré definované funkce musí být následně k serveru registrovány.

```
$server = new SoapServer(null, array('uri' => 'adresa_server'));
$server->addFunction('jméno_funkce');
```

Seznam jednotlivých funkcí:

1) **get_users($array_reser_users)** – vrací vygenerovaný html kód se seznamem všech uživatelů v lokalitě. Pokud je argumentem této funkce pole spolupracujících uživatelů, jsou tito v select boxu označeni jako již vybráni.
2) **get_resers($user_id,$site_id,$all)** – vrací seznam rezervací, kterých se daný uživatel předaný argumentem $user_id z lokality dané argumentem $site_id účastní. Pokud nenastavíme argument $all na hodnotu 1, vypíší se pouze budoucí rezervace.
3) **get_task_name($taskid)** – podle předaného argumentu $taskid, který jednoznačně identifikuje úlohu, vrací její název.
4) **get_topology_name($topid)** – podle předaného argumentu $topid, který jednoznačně identifikuje topologii, vrací její název.
5) **get_resmate_users($resid,$user)** – vrací seznam spolupracujících uživatelů, kteří spolupracují na rezervaci předané argumentem $resid zarezervované uživatelem předaným argumentem $user.
6) **get_active_resers($lang,$site,$user,$sid,$tutor)** – vrací vygenerovaný html kód aktivní rezervace. Jazyková verze se zvolí na základě argumentu $lang. Aktivní rezervace se vyhledávají podle argumentů $site a $user, kteří jednoznačně identifikují uživatele. Argumenty $sid a $tutor jsou pro vygenerování části kódu s formulářem pro spouštění Java Appletu.
7) **dismate_users($resid,$user,$site)** – na základě jednoznačného identifikátoru rezervace předaného argumentem $resid a jednoznačného identifikátoru uživatele určeného argumenty $user a $site odstraní daného uživatele ze seznamu spolupracujících uživatelů.
8) **get_task_detail($lang,$taskid)** – vrací vygenerovaný html kód detailu úlohy s jednoznačným identifikátorem předaným v argumentu $taskid a v jazyce určeném argumentem $lang.
9) **get_verify($did,$tm,$resid)** – vrací počet ms do konce rezervace určenou argumentem $resid, a slouží pro vzdálenou kontrolu oprávnění přístupu k zařízení určené argumentem $did. Pokud je hodnota atributu $tm rovna 1 jedná se o tutor mód.

3.3.2. Implementace SOAP klienta

V jednotlivých modulech řídící aplikace Virtlabu, ve kterých potřebujeme využít volání SOAP serveru, deklarujeme objekt $client, který je typu SoapClient a následně voláme požadovanou funkci. Jedná se o jednotlivé funkce, které jsme definovaly v SOAP serveru.

```
$client = new SoapClient(null, array('location' => 'soap_server',
'uri' => 'uri_server'));
$client->název_funkce(argumenty_funkce);
```

Implementace SOAP klienta se týká těchto modulů:

1) **reser-mine.php** – zobrazuje seznam rezervací včetně rezervací vzdálených a zajišťuje přidávání spolupracujících kolegů ze všech lokalit.
2) **reser-active.php** – zobrazuje aktivní rezervace včetně všech jejich náležitostí.
3) **tasks-list.php** – zobrazuje seznam laboratorních úloh včetně jejich detailních informací.
4) **verify.php** – zajišťuje konzolovému serveru ověření přístupu uživatele k laboratornímu vybavení.

3.4. Testování

Vývoj a samotné testování jednotlivých kroků implementace probíhalo mezi virtuálními lokalitami most.virtlab.cz a zlin.virtlab.cz. Nejdůležitějším krokem bylo otestování správné komunikace mezi SOAP klientem a serverem. Následně se jednalo spíše o vizuální kontrolu očekávaného chování.

4. Předkonfigurace

U náročnějších laboratorních úloh, které mají za úkol procvičit specifičtější problematiku, vede základní konfigurace ke zbytečnému odvádění od problému, a také k určité ztrátě času, který by mohl být využit efektivněji. Tento problém by mohl být velmi snadno vyřešen, pokud by byla nějaká možnost mít laboratorní prvky předem předkonfigurované.

Dalším velmi užitečným využitím předkonfigurací může být nakonfigurování laboratorních prvků, které jsou studentům nepřístupné a plní určitou funkci. Může se jednat například o poskytovatele ISP, který provozuje různé služby nebo je třeba nežádoucí, aby studenti měnili konfiguraci tohoto konkrétního laboratorního zařízení.

4.1. Analýza

Stěží bychom přinutili a hlavně ohlídali, aby všichni uživatelé na konci své rezervace, jednotlivé laboratorní vybavení vymazali, nebo vrátili do výchozího stavu. Proto na konci každé rezervace dochází k automatickému mazání použitého laboratorního vybavení, tak aby další uživatelé rezervující si dané laboratorní vybavení, měli toto vybavení ve výchozím stavu.

O veškeré akce, týkající se mazání jednotlivého laboratorního vybavení, se stará komponenta jménem mazací server, která se spouští na konci rezervace v době pětiminutového okna, které se automaticky přidává k uživatelem definovanému času rezervace. Toto pětiminutové okno bylo zvoleno s dostatečnou rezervou.

Doba potřebná na vymazání nebo navrácení do výchozího stavu použitého laboratorního vybavení je závislá na konkrétním typu prvku a na metodě, která se pro tuto akci použije. V současné době bychom mohly tyto metody rozdělit do tří skupin:

- otáčení XEN instancí sloužících pro simulaci PC a ISP
- otáčení směrovačů a přepínačů pomocí APC
- posílání sady příkazů pro reload směrovačů a přepínačů

Při návrhu a implementaci původní aplikace řídícího serveru v PHP se počítalo, že k jednotlivým úlohám budou přidružený i soubory s počátečními a ukázkovými cílovými konfiguracemi.

4.2. Návrh

Tak jako se na konci rezervace laboratorní vybavení maže, mohli bychom použit obdobné řešení i pro nahrávání konfigurací v době před začátkem rezervace. Uživatel tak bude mít na začátku své rezervace veškeré zarezervované vybavení předkonfigurované a dále bude moci pokračovat v řešení zadané úlohy.

Pro ukládání předkonfigurací bude nejlépe využít již stávající implementaci a databázovou strukturu podporující ukládání k jednotlivým úlohám soubor s počáteční konfigurací.

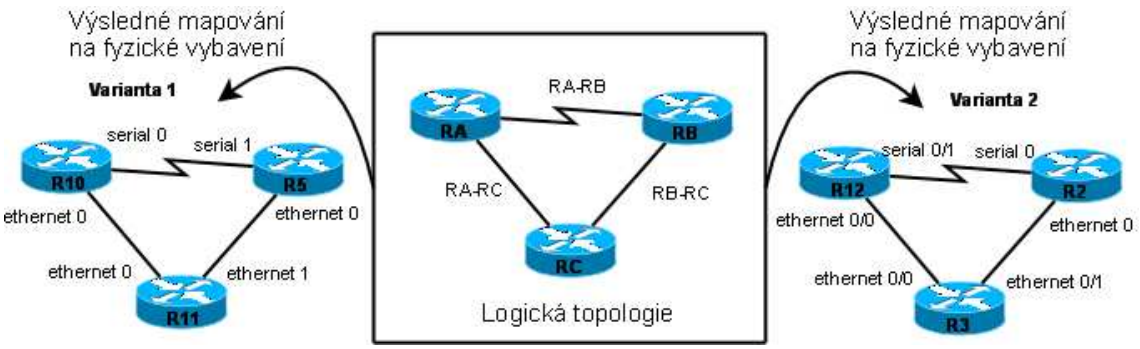
Pro samotné nahrávání na jednotlivé laboratorní zařízení, které si uživatel zarezervoval, opět nebudeme vymýšlet žádnou novou komponentu, ale využijeme již navrženou komponentu, která nám poskytuje přístup k jednotlivým laboratorním prvkům. Jedná se o konzolový server, který poskytuje přístupové rozhraní ke všem prvkům.

4.2.1. Soubor s předkonfigurací

Vzhledem k návrhu ponechat původní implementaci řídící aplikace v PHP spolu s původní databázovou strukturou, je třeba navrhnout způsob, jakým se do jednoho souboru s počáteční konfigurací, zaznamenají jednotlivé konfigurace.

Každá úloha typicky obsahuje několik laboratorních zařízení a je nutno odlišit, která počáteční konfigurace patří na které zařízení. Jedno z možných řešení by mohlo být rozdělení souboru do jednotlivých sekcí podle jednotlivých laboratorních zařízení.

Dále nám situaci významně komplikuje skutečnost, že při opakovaných rezervacích konkrétní úlohy dochází k různým výsledkům mapovaní úlohou definovaného logického vybavení na fyzické vybavení dostupné v konkrétním čase. Vzhledem k této vlastnosti systému není možno používat názvy konkrétních fyzických rozhraní. Příklad této situace demonstruje obrázek 8.



obrázek 8: mapování logické topologie na topologii fyzickou

Topologie k úloze v popisném XML formátu je daná logickými názvy laboratorního zařízení spolu s jejich vlastnostmi a jednotlivými linkami mezi těmito prvky, které jsou jednoznačné určené svým jménem a opět mohou mít různé vlastnosti a omezení. Na základě těchto informací vybírá mapovací algoritmus nejvhodnější dostupné fyzické laboratorní vybavení. Podrobnější informace o mapovacím algoritmu lze najít v diplomové práci Jana Vavříčka [4].

Na obrázku s mapováním logické topologie na topologii fyzickou (viz obrázek 8) můžete vidět příklad toho, jak mapovaní dopadlo pokaždé jinak a v případě, že bychom měli v našich konfiguračních souborech názvy fyzických rozhraní napevno, konfigurace by se nemusela korektně provést.

Vzhledem k této skutečnosti, je třeba i v předkonfiguračních souborech udávat logické názvy laboratorního zařízení. Pro logické názvy rozhraní byl navržen formát ve tvaru: logický_název_prvku:jednoznačný_idenentifikátor_linky. Pro lepší vyhledávání, při následném přemapování, se navíc vkládá mezi znaky #. Příklad: #RA:RA-RB#, kde RA je logický název laboratorního zařízení a RA-RB je jednoznačný identifikátor linky.

Jednotlivé konfigurace pro laboratorní zařízení z úlohy se pak vkládají mezi elementy <logický_název_zařízení> a </logický_název_zařízení>. Příklad konkrétního souboru s předkonfiguracemi je součástí přílohy viz Příloha C.

4.2.2. Přemapování a rozklad na dílčí soubory

Při samotné konfiguraci laboratorních zařízení potřebujeme jejich skutečná jména včetně názvů jejich fyzických rozhraní. Výsledek mapování je znám v okamžiku úspěšně zarezervované úlohy a hned poté můžeme přemapovat veškerá logická jména na fyzická. Při této příležitosti je vhodné připravit z původního souboru se všemi předkonfiguracemi jednotlivé dílčí soubory s předkonfiguracemi pro jednotlivá laboratorní zařízení, které se odešlou konzolovému serveru na začátku aktivace uživatelské rezervace.

4.3. Implementace

Při implementaci bylo postupováno podle navrhovaného řešení. Jako první vznikl modul VirtlabPreconfig, který je zodpovědný za správné přemapování souboru s předkonfiguracemi z logických na fyzické názvy laboratorních zařízení a jejich rozhraní.

4.3.1. VirtlabPreconfig

V době kdy si uživatel rezervuje úlohu a ta je úspěšně zarezervováná, dochází k volání modulu VirtlabPreconfig, který je implementován jako další PHP třída v rámci hlavní řídící aplikace. Tato třída se nachází v souboru virtlabPreconfig.php.inc a obsahuje funkci generatePreconfig, jejímž argumentem je id rezervace, na základě kterého se vyhledá odpovídající mapování logických názvů na fyzické názvy zařízení a rozhraní.

Následně se všechny logické názvy v souboru s konfigurací přemapují na odpovídající fyzické názvy a celý soubor se rozloží, na základě jednotlivých sekcí, na jednotlivé konfigurační soubory pro konkrétní laboratorní zařízení. Ty se vzápětí uloží jako soubory s fyzickými názvy zařízení do adresáře s názvem id rezervace, který se vytvoří v adresáři s předkonfiguracemi virtlab/data/pre-config a jsou připraveny na aktivaci rezervace.

4.3.2. Nahrávání předkonfiguračních souborů

O spouštění aktivačního skriptu activator.sh, který zajistí veškerou konfiguraci tunelovacího serveru a správné propojení fyzické topologie, se stará systémový nástroj at démon (obdoba nástroje CRON). Více o informací o tunelovacím serveru a aktivačním skriptu najdete v bakalářské práci Václava Bortlíka [9].

Na konec aktivačního skriptu activator.sh přidáme příkaz pro spuštění skriptu pre-configurator.sh s parametrem id rezervace. Tento skript pak na základě id rezervace projde odpovídající adresář s předkonfiguračními soubory a ty pak jeden po druhém pošle s odpovídajícími parametry na konzolový server.

Zde je prostor pro nahrávání konfigurací na laboratorní zařízení, které je později studentům nepřístupné.

4.3.3. Využití konzolového serveru

Konzolový server, na základě předaných parametrů, vybere odpovídající laboratorní zařízení a dále se postará o odeslání konfigurace.

Při navázání spojení s konzolovým serverem, který pomocí skriptu verify.php ověřuje oprávněnost přístupu, využíváme toho, že požadavek přišel přímo z řídícího serveru Virtlabu.

Podrobné informace o funkci konzolového serveru můžete najít v diplomové práci Radka Nováka [10].

4.4. Testování

Samotné testování bychom mohli rozdělit na několik následujících částí:

1) Správná funkčnost přemapování logických názvů na fyzické názvy – jedná se o jednoduché ověření, zdali se všechny názvy správně přemapovaly.
2) Správné vytvoření jednotlivých souborů s předkonfiguracemi – kontrola jednotlivých dílčích souborů, zdali se globální soubor se všemi konfiguracemi rozdělil podle odpovídajících sekcí na soubory s fyzickými názvy zařízení.
3) Komunikace s konzolovým serverem – jedná se především o korektní navázání spojení a předání odpovídajících parametrů.
4) Ověření nahrané konfigurace přímo na zařízení – na závěr zbývalo ověřit, jestli se všechny konfigurace úspěšně nahrály až na samotné zařízeni.

Nejvíce problémů způsobovalo nahrávání konfigurací na samotná laboratorní zařízení. Většina zařízení jsou cisco směrovače a přepínače, pro které je typické občasné „zatuhnutí“ konfigurační konzole a pro obnovení funkčnosti je třeba poslat několik sekvencí klávesy Enter, případně vyčkat několik desítek sekund.

Tento problém by měl být vyřešen v nové implementaci konzolového serveru realizované v diplomové práci Radka Nováka [10].

Další problém vznikal při příliš rychlém posílání konfigurací na zařízení, ten byl ale úspěšně vyřešen vložením 200 ms pauzou mezi znaky.

5. Archív uživatelských konfigurací

Většina systémů ať už hardwarových nebo softwarových umožňuje nějakým způsobem uložit stávající konfiguraci nebo stav ve kterém se právě nachází, tak aby se uživatel mohl kdykoliv k nějakému stavu nebo konfiguraci vrátit. Případně pokračovat ve své práci, tam kde naposledy skončil.

Lepší systémy pak navíc umožňují ukládat a archivovat i různé verze konfigurací, případně si mezi nimi libovolně přepínat.

5.1. Analýza

Uživatel Virtlabu, který by si rád svou práci uložil, má v podstatě jen jednu možnost. Připojit se postupně na všechna zařízení, získat jednotlivé konfigurace a ty si následně uložit někde u sebe. Tato metoda jistě splní svůj účel, ale je zdlouhavá a pro uživatele jistě velmi nepříjemná a nepohodlná.

Další zdlouhavá a nepohodlná činnost nastává v okamžiku, kdy chce uživatel uložené konfigurace použít. Opět se musí postupně připojit na všechna zařízení a každou konfiguraci jednu po druhé na jednotlivá zařízení nahrát. Ovšem nesmíme zapomenout, že díky dynamickému výběru laboratorního zařízení, může docházet k nekompatibilitě fyzických názvů rozhraní, stejně jako jsme se s toto vlastností setkali u předkonfigurací.

V následujících podkapitolách je popsán postup při návrhu a implementaci archívu uživatelských konfigurací, který původní zdlouhavý postup zautomatizuje, a velmi tak usnadní práci uživatelům Virtlabu.

5.2. Návrh řešení

Konfigurace celé úlohy se skládá z konfigurací na jednotlivých laboratorních zařízeních, proto je třeba ukládat všechny dílčí konfigurace. Znamená to postupně se připojit na všechna laboratorní zařízení a získat jejich konfiguraci.

Vzhledem k tomu, že uložené konfigurace bude třeba později na zařízení zase nahrát, bude vhodné je ukládat ve formátu použitelném pro předkonfigurace. Znamená to opět přemapovat všechny fyzické názvy rozhraní na logické názvy a jednotlivé dílčí konfigurace vložit do sekcí podle logického názvu zařízení.

Takto vzniklý soubor s konfiguracemi uložíme do databáze. A abychom umožnili uživatelům ukládat více různých verzí konfigurací k jedné úloze, zajistíme jim ukládání konfigurací pod vlastními názvy. Pokud si uživatel bude znovu rezervovat stejnou úlohu, dostane na výběr seznam svých uložených konfigurací k dané úloze, a tudíž možnost jak pokračovat tam, kde naposledy skončil.

Pro nahrávání konfigurací využijeme již implementovaného modulu pro předkonfigurace. Tím zajistíme, že uživatel bude mít na začátku své rezervace jednotlivé laboratorní vybavení nakonfigurované ve stavu, ve kterém si ho uložil.

5.2.1. Rozšíření databáze o archív konfigurací

Pro ukládání všech uživatelských konfigurací vytvoříme novou tabulku `user_config_archive`, která bude součástí databáze řídícího serveru `dvldb-www`. Jako primární klíč použijeme automaticky generovaný jednoznačný identifikátor, další atributy jako jsou: identifikátor uživatele, úlohy a rezervace budou cizí klíče na tabulky: uživatelů, úloh a rezervací. Posledními atributy v tabulce jsou jméno a samotný obsah souboru s konfigurací.

5.2.2. Rozšíření GUI a modulu pro předkonfigurace

Aby si uživatelé mohli konfigurace ukládat, bude třeba rozšířit uživatelské rozhraní o funkci ukládání konfigurací. Dále bude třeba upravit způsob rezervace úlohy, aby si uživatelé mohli vybrat, zdali se jim má některá z jimi uložených konfigurací na zařízení nahrát. Současně bude třeba rozšířit modul pro nahrávání předkonfigurací tak, aby dokázal pracovat kromě původního souboru s předkonfiguracemi (definovaného k úloze) i s konfiguracemi uloženými uživateli.

5.3. Implementace

Díky využití stávající funkcionality modulu pro nahrávání předkonfigurací se nám samotná implementace archívu značně zjednoduší, a v podstatě se omezí pouze na implementaci získávání a ukládání konfigurací z laboratorních zařízení a rozšíření uživatelského rozhraní.

5.3.1. Načítání konfigurací

Pro načítání konfigurací z laboratorního zařízení využijeme konzolového serveru, ke kterému se připojíme přes socket implementovaný v PHP [11]. Proto, abychom se mohli připojit k zařízení, musíme znát: IP adresu lokálního konzolového serveru, jméno zařízení včetně lokality ve které se nachází a id rezervace. Na základě těchto informací můžeme navázat socketové připojení na konzolový server.

Všechna laboratorní zařízení, momentálně používaná ve Virtlabu, jsou od společnosti Cisco Systems, a tak lze využít jednotného způsobu získávání konfigurací. Ten spočívá ve výpisu aktuálně běžící konfigurace příkazem: **`show running-config`**. Proto zapíšeme na socket tento příkaz, a následně ze socketu načteme vypsanou konfiguraci. Vzhledem k tomu, že konfigurace může být obsáhlá, a ve výchozím nastavení se vypisuje po částech, použijeme

příkaz: **`terminal lenght 0`**. Tím zajistíme, že se konfigurace vypíše celá najednou. Proto, abychom mohli tento příkaz použít, musíme se nacházet v privilegovaném režimu, do kterého se přepneme použitím příkazu: **`enable`**. Tento režim ovšem může být přístupný pouze po zadání hesla. Jediné povolené heslo, které není mezi zakázanými příkazy, je **`cisco`**. Celá sekvence příkazů odeslaných na zařízení bude vypadat následovně:

```
enable
cisco
terminal lenght 0
show running-config
```

Následně budeme ze socketu číst tak dlouho, dokud nenarazíme na **`end`** (výraz označující konec výpisu konfigurace), nebo dokud při načítání nedojde k timeoutu.

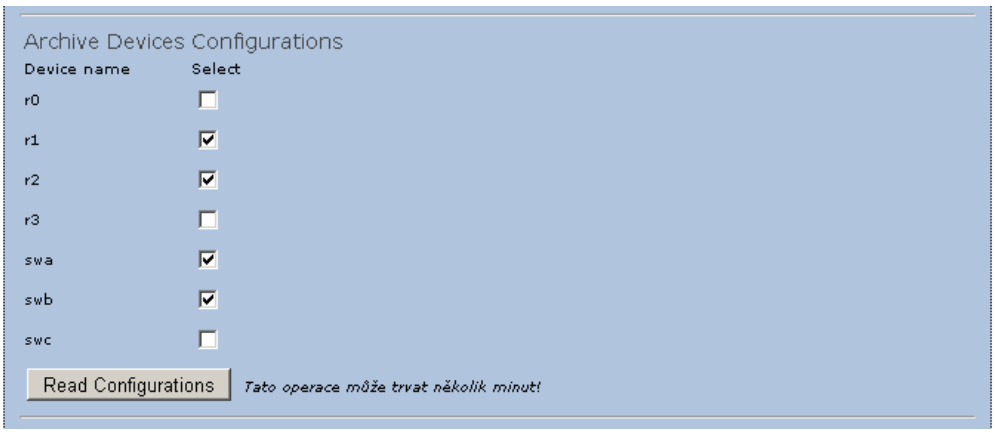
5.3.2. Ukládání konfigurací

Poté co načteme jednotlivé konfigurace, musíme přemapovat všechny fyzické názvy rozhraní na logické názvy, opět ve formátu #název_zařízení:jednoznačný_identifikátor_linky#. Dílčí konfigurace pak vložíme do sekcí označených podle logického názvu zařízení <logický_název_zařízení></logický_název_zařízení>.

U takto upravených konfigurací už nezbývá nic jiného, než je pod uživatelským názvem uložit do databáze.

5.3.3. GUI pro načítání a ukládání konfigurací

Abychom uživateli umožnili výběr jen některých laboratorních zařízení, použijeme formulář s checkboxy, kde si uživatel zaškrtnutím vybere, ze kterých zařízení chce konfigurace načíst (viz obrázek 9).



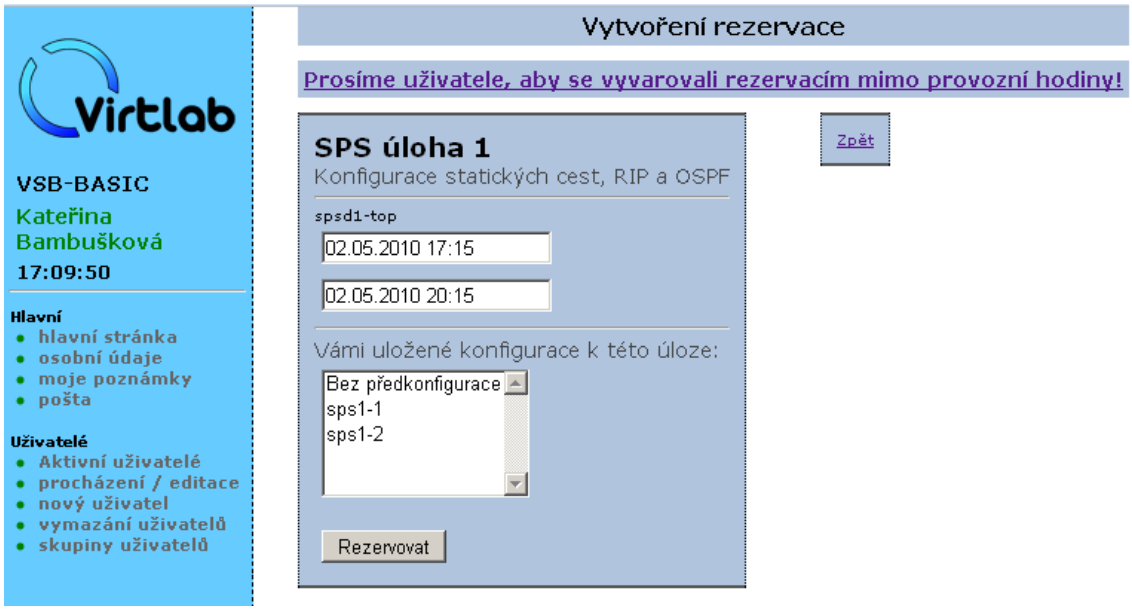
obrázek 9: volba zařízení k uložení konfigurací

Následně se každá konfigurace načte do textového pole, a uživatel má ještě před uložením možnost konfiguraci zkontrolovat a případně upravit (viz obrázek 10).

obrázek 10: načtené konfigurace

5.3.4. Výběr uložené konfigurace

V databázi řídícího serveru máme v tabulce archívu konfigurací uloženy všechny uživatelské konfigurace. V případě, že se uživatel rozhodne si zarezervovat úlohu, kterou již dříve konfiguroval a uložil si k ní i konfiguraci, zobrazíme mu seznam všech jeho konfigurací k dané úloze. Uživatel si pak může vybrat, jestli bude mít laboratorní vybavení nakonfigurováno některou jeho variantou, nebo bez konfigurace ve výchozím stavu (viz obrázek 11).



obrázek 11: výběr konfigurace při rezervaci úlohy

5.4. Testování

Samotné testování bychom opět mohli rozdělit na několik následujících částí, tak jak se postupovalo při implementaci:

1) Komunikace řídící aplikace v PHP s konzolovým serverem – navázání socketového spojení s konzolovým serverem a čtení ze socketu.
2) Správná funkčnost přemapovávání fyzických názvů na logické názvy – jde o inverzní funkci k té, kterou jsme použili při přemapovávání předkonfigurací. Opět se jedná jednoduché ověření, zdali se všechny názvy správně přemapovali.
3) Ukládání konfigurací do databáze – kontrola správného spojení dílčích konfigurací a uložení do databáze.
4) Ověření nahrané konfigurace přímo na zařízení – na závěr zbývalo ověřit, jestli se všechny konfigurace úspěšně nahrály až na samotné zařízeni.

Poté co byly úspěšně naimplementovány a otestovány jednotlivé časti archívu uživatelských konfigurací, byla dopsána do uživatelského manuálu kapitola týkající se práce s tímto archívem a dalšího testování se tak mohli zúčastnit samotní uživatelé.

Celý uživatelský manuál je součástí přiloženého CD, část týkající se archívu uživatelských konfigurací je součást přílohy (viz Příloha D).

6. Další utility a rozšíření

Již během nasazování systému Virtlab do pilotního provozu se ukázala řada nedostatků například v souvislosti se správou konfiguračních souborů, které obsahuje každá komponenta v relativně velkém množství.

Ze začátku provozu systému Virtlab se objevovaly různé chyby, případně jiné nedostatky. Ovšem v té době měli uživatelé jedinou možnost, a to nahlásit chybu emailem. To vedlo později k tomu, že byl navržen a implementován systém pro správu chyb – Bug tracking systém.

Při samotném provozu produkčního prostředí, bylo ještě naimplementováno několik podpůrných skriptů, které měly za cíl usnadnit práci administrátorům. Jednalo se především o instalační skript a další skripty pro spuštění, zastavení nebo restart jednotlivých komponent systému. Jednotlivé skripty se nacházejí na přiloženém CD, ale vzhledem k tomu, že byly implementovány nad rámec zadání, nejsou v textu podrobně rozepisovány.

6.1. Generátor konfigurací

Ručně spravovat velké množství konfiguračních souborů jednotlivých komponent je velmi náročné, a také velmi náchylné na nekonzistenci a zanesení různých chyb.

Další nevýhodou je, že každý správce lokality potřebuje mít znalosti ohledně formátu a struktury jednotlivých konfiguračních souborů, proto se uvažovalo o nějaké utilitě, která by práci s konfiguračními soubory zjednodušila.

6.1.1. Analýza stávajících konfiguračních souborů

Ještě než došlo k samotnému návrhu generátoru konfigurací byly zanalyzovány veškeré konfigurační soubory jednotlivých komponent a sepsán konfigurační manuál, který měl být nápomocen všem administrátorům jednotlivých lokalit.

Manuál můžete najít v jeho původní formě na přiloženém CD.

6.1.2. Návrh generátoru konfigurací

Většina konfiguračních souborů je závislá na seznamu laboratorního vybavení, které je aktuálně pro studenty k dispozici. Součástí systému tedy musí být i možnost evidovat veškeré laboratorní vybavení a jejich vlastnosti, spolu s informací o připojení ke konkrétnímu spojovacímu prvku (ASSSK nebo VLMUX).

Další skupina konfiguračních souborů obsahuje informace o vzdálených komponentách v okolních lokalitách. Jedná se především o IP adresy jednotlivých serverů.

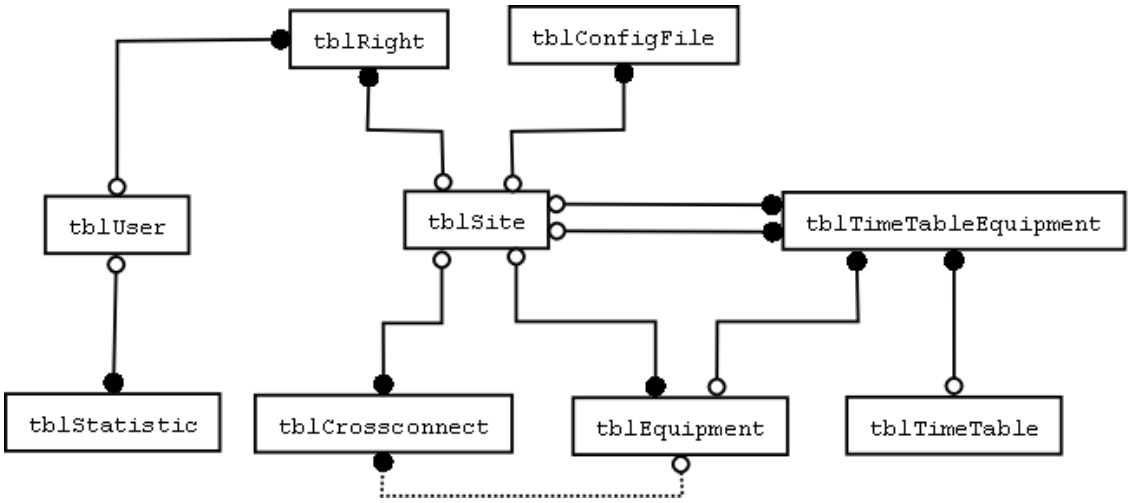
Vzhledem k předpokládanému nárůstu počtu lokalit, je také třeba definovat kdo může editovat jaké informace. Z tohoto důvodu bude veškerá administrace povolena pouze po přihlášení a pouze pokud bude mít přihlášený uživatel na konkrétní akci oprávnění.

Dalším bezpečnostním opatřením bude sledování a zaznamenávání veškerých akcí, které modifikují záznamy v databázi.

Veškeré vygenerované konfigurační soubory se budou rovněž ukládat do databáze, a to se vzrůstajícím číslem verze. V případě, že by se ukázal nějaký nedostatek u nejnovější verze, budou tak kdykoliv zpětně přístupné.

Na základě těchto požadavků byl proveden následující návrh databáze:

Základní E-R diagram



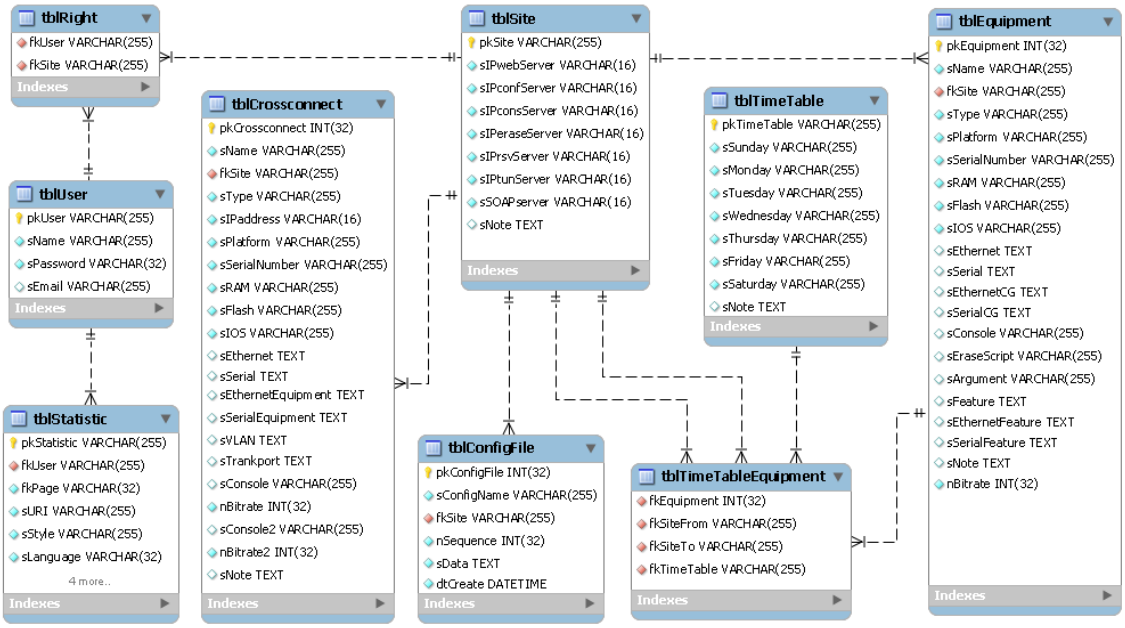
obrázek 12: E-R diagram generátoru konfigurací

Popis jednotlivých tabulek

tblSite – tabulka jednotlivých lokalit obsahující informace o lokalitách a IP adresy jednotlivých serverů.
tblUser – tabulka uživatelů (administrátorů), kteří s generátorem konfigurací pracují, včetně jejich emailového kontaktu.
tblRight – vazební tabulka obsahující informace o uživatelích, kteří mají oprávnění modifikovat záznamy jednotlivých lokalit.
tblStatistic – tabulka evidující jednotlivé přístupy uživatelů, včetně jejich chování v systému.
tblConfigFile – tabulka obsahující jednotlivé vygenerované konfigurační soubory.
tblEquipment – tabulka s laboratorním vybavením, včetně veškerých informací o zařízení.

tblCrossconnect – tabulka jednotlivých propojovacích prvků, včetně veškerých informací o připojení laboratorního vybavení k portům spojovače.
tblTimeTable – tabulka časových rozvrhů pro sdílení laboratorního vybavení mezi lokalitami.
TblTimeTableEquipment – vazební tabulka definující která lokalita (from), které lokalitě (to), půjčuje jaké laboratorní vybavení.

E-R diagram se všemi atributy



obrázek 13: podrobný E-R diagram

6.1.3. Implementace

Celý systém generátoru konfigurací byl implementován v jazyce PHP, který generuje jednotlivé HTML stránky a využívá databázi MySQL jako úložiště dat. Odpovídajícího vzhledu je dosaženo za pomocí CSS stylů.

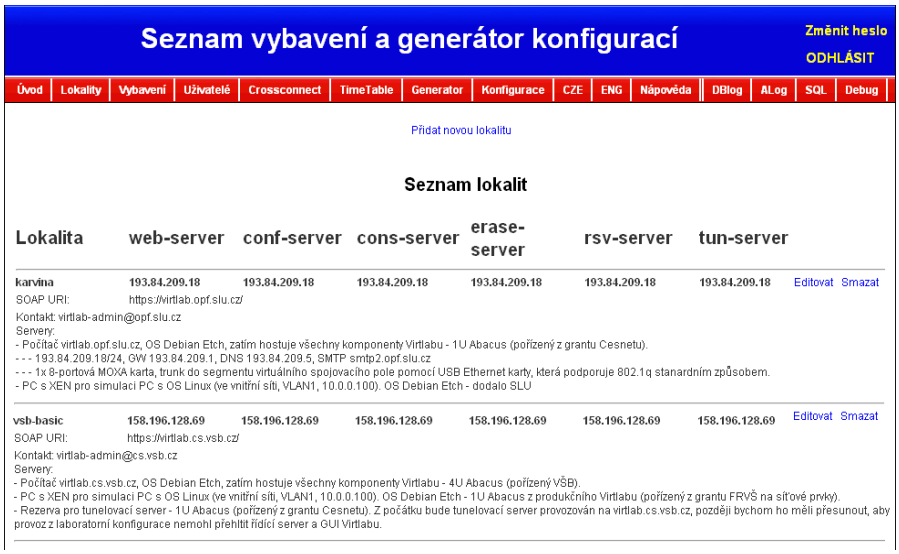
Požadovaná funkcionalita byla rozdělena do několika modulů, podle toho, s jakými informacemi se pracuje nebo jaké akce vykonává. Každý modul je v samostatném souboru a využívá řadu funkcí, které jsou opět rozděleny do samostatných souborů v adresáři s funkcemi.

Popis jednotlivých modulů a funkcí:

user.php (user_fce.php, site_fce.php) – evidence uživatelů (administrátorů), kteří mají přístup do systému. Umožňuje editaci uživatelských údajů a práv přístupu k jednotlivým lokalitám, případně resetovat přístupové heslo (viz obrázek 14).

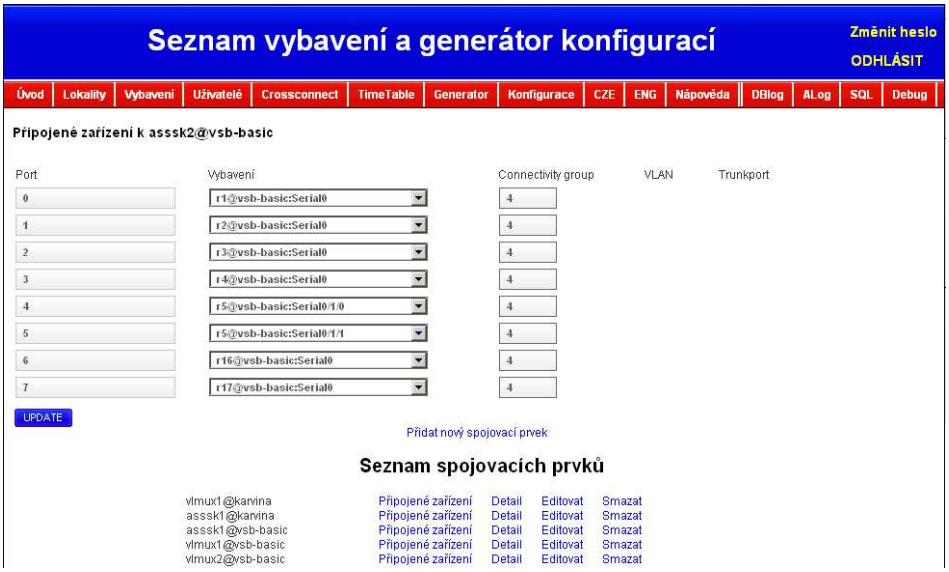
obrázek 14: práce s uživateli

site.php (site_fce.php) – evidence lokalit, IP adresy serverů na kterých běží jednotlivé komponenty, URI SOAP serverů, případně další poznámky týkající se konkrétní lokality (viz obrázek 15). Soubor funkcí pro práci s lokalitami využívají i jiné moduly.



obrázek 15: evidence lokalit

crossconnect.php (crossconnect_fce.php, site_fce.php, equipment_fce.php) – evidence všech propojovacích prvků typu ASSSK nebo VLMUX, jejich vlastností a rozhraní, včetně připojeného laboratorního vybavení (viz obrázek 16).



obrázek 16: zařízení připojená ke spojovacímu prvku

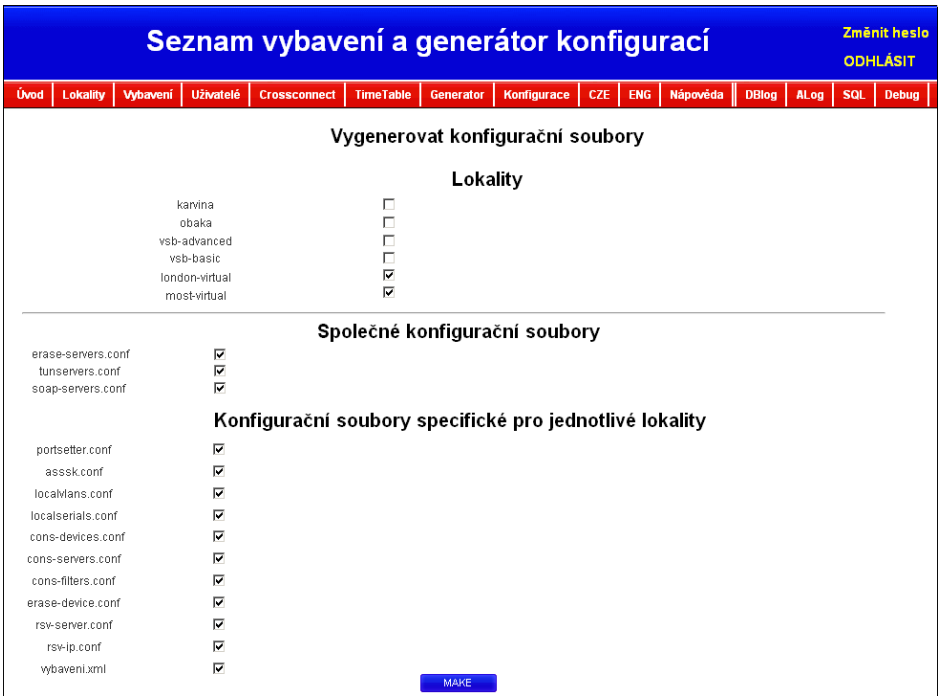
timetable.php (timetable_fce.php, site_fce.php, equipment_fce.php) – evidence časových rozvrhů pro sdílení laboratorního vybavení mezi jednotlivými lokalitami.

equipment.php (equipment_fce.php, site_fce.php) – evidence veškerého laboratorního vybavení, včetně detailního seznamu různých vlastnosti, jako jsou: platforma, sériové číslo, velikost pamětí, verze IOSu, konzolový port, rozhraní a další definované vlastnosti.

accesslog.php a **dblog.php** (log_fce.php) – sledování a zaznamenávání aktivity uživatelů, který pracují se systémem generátoru konfigurací.

configfile.php (site_fce.php) a **getfile.php** – slouží pro práci s vygenerovanými konfiguračními soubory.

generator.php (generator_fce.php, site_fce.php, equipment.php) – obsahuje soubor funkcí pro generování konkrétních konfiguračních souborů pro vybranou skupinu lokalit (viz obrázek 17).



obrázek 17: generování konfiguračních souborů

Další důležité součásti nezbytné pro správnou funkčnost systému:

function_inc.php – sada obecných funkcí, pro práci se systémem
connection_inc.php – nastavení přístupu k databázi
inicialization_inc.php – nastavení systémových proměnných
skelet_inc.php – definice kostry GUI

6.1.4. Config-downloader

Pro snadnější manipulaci s konfiguračními soubory byla navržena utilitka pro stahování vygenerovaných konfiguračních souborů na jednotlivé lokality. Jedná se o bashový skript **config-downloader.sh**, který najdete v příloze (viz Příloha E).

Syntaxe pro spuštění skriptu:

`config-downloader.sh <sitename>`
- stáhne poslední vygenerovanou verzi všech konfiguračních souborů pro vybranou lokalitu.

`config-downloader.sh <sitename> <configname> [sequence number]`
- stáhne konkrétní konfigurační soubor pro vybranou lokalitu, pokud nezadáme sekvenční číslo, stáhne se naposledy vygenerovaný soubor.

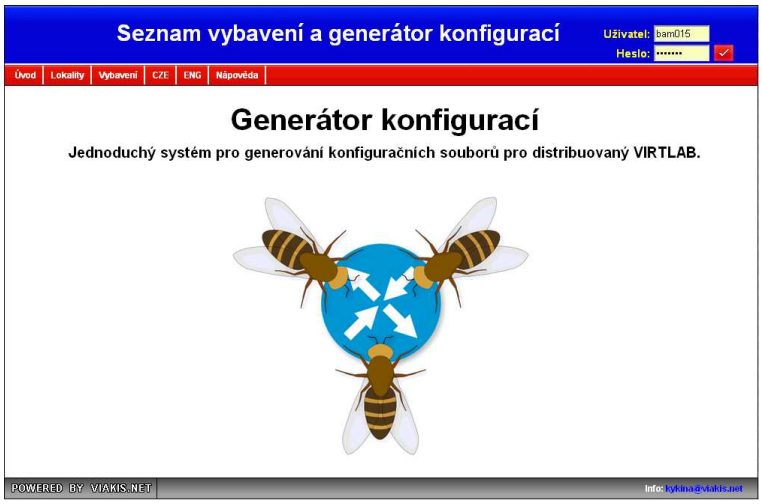
`config-downloader.sh help`
- zobrazí nápovědu k použití tohoto skriptu.

6.1.5. Testování a shrnutí pilotního provozu

Dílčí testování probíhalo postupně již při samotné implementaci, kdy se sledovalo, zdali se systém chová dle očekávání. Během provozu se projevilo pár nedostatků, převážně z pohledu ergonomizace uživatelského rozhraní, které byly postupně odstraněny.

Dále byl systém postupně rozšířen o možnost generovat řadu dalších konfiguračních souborů, které nově vznikly až po návrhu a implementaci generátoru konfigurací.

Samotný generátor konfigurací (viz obrázek 18), který pro nepřihlášené uživatele poskytuje seznam laboratorního vybavení a informace o jednotlivých lokalitách, můžete najít na stránkách: http://config.viakis.net, včetně odkazu na popis generátoru uvedený na oficiálních stránkách projektu [2].



obrázek 18: úvodní stránka generátoru konfigurací

6.2. Bug tracking systém

I ten nejlépe navržený a otestovaný systém občas obsahuje nějakou drobnou chybu, nebo uživatelé touží po nějakém vylepšení. Proto jsme se rozhodli zavést podporu pro hlášení chyb a návrhů na zlepšení. A to z důvodu, že předchozí způsob posílání chyb a námětů přes email se ukázal jako neefektivní.

6.2.1. Analýza požadavků

Požadavky na systém:

- jednoduchý způsob zadávání návrhů a chyb, automatické doplňování údajů
- možnost přiložit soubor (například screen obrazovky s chybou)
- napojení na existující systémem VirtIS a mít možnost přidělit chybu konkrétnímu vývojáři
- automatické zasílání emailu (při nahlášení, při vyřešení)
- možnost definovat stav, ve kterém se incident nachází

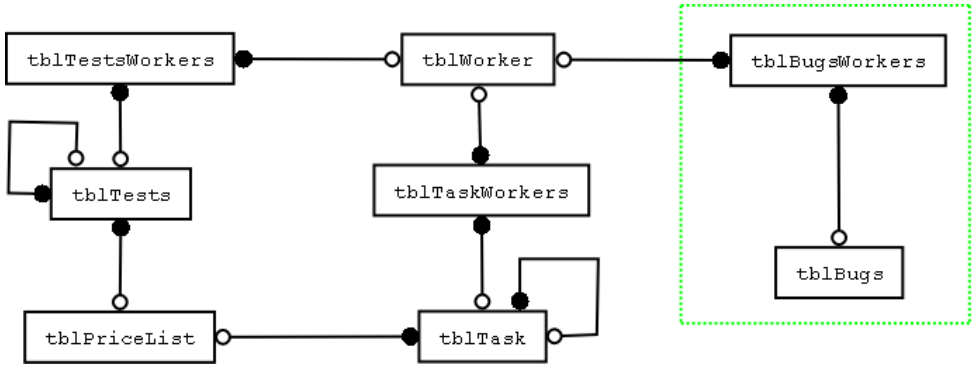
6.2.2. Návrh Bug tracking systému

Tak rozsáhlý projekt, jako je Virtlab, s velkým množstvím vývojářů si už dříve vyžádal navržení informačního systému – VirtIS. Zde se eviduje seznam všech vývojářů a různých úkolů, včetně termínů do kterých by měly být vyřešeny. Tento systém již podporuje zasílání emailu v případě přidělení vývojáře k nějakému úkolu, stejně tak jako zasílání emailu s upozorněním, že už se blíží termín odevzdání.

Jednoduchým rozšířením systému VirtIS o modul, který bude evidovat nahlášené chyby získáme Bug tracking systém, který vyhovuje zadaným požadavkům.

Pro usnadnění práce uživatelům, kteří budou návrhy a chyby zadávat, naimplementujeme odesílací formulář jako součást uživatelského rozhraní řídícího serveru Virtlabu.

Rozšíření stávající databáze

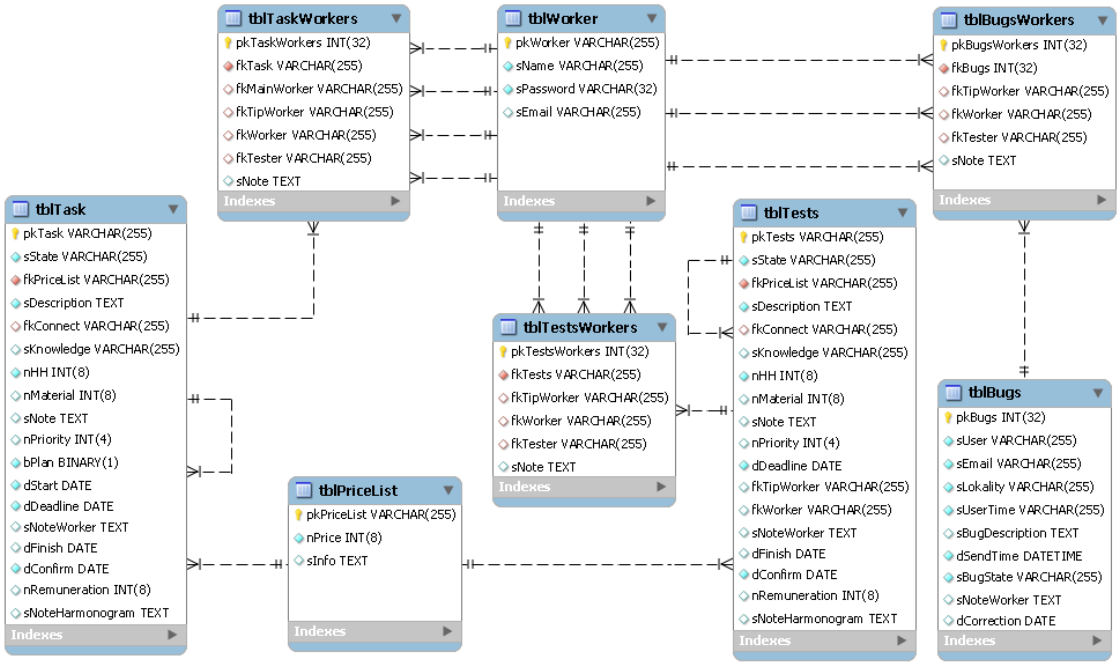


obrázek 19: E-R diagram rozšířené databáze

Popis jednotlivých tabulek

tblWorker – tabulka pracovníků (vývojářů a testerů), kteří se systémem VirtIS pracují, včetně jejich emailového kontaktu.
tblTask – tabulka jednotlivých úkolů obsahující veškeré informace jako jsou: stav, termín, typ úkolu, zdroj financování, datum dokončení, časová náročnost a další.
tblTaskWorkers – vazební tabulka obsahující informace o jednotlivých pracovnících, kteří se podíleli na řešení úkolu.
tblTests – tabulka jednotlivých testů obsahující veškeré informace jako jsou: stav, termín, zdroj financování, datum dokončení, datum schválení, časová náročnost a další.
tblTestsWorkers – vazební tabulka obsahující informace o jednotlivých testerech, kteří se na testování podíleli.
tblBugs – tabulka nahlášených incidentů obsahující veškeré informace jako jsou: uživatel, který incident nahlásil, lokalita ze které přišel, datum a čas výskytu, popis případně název přiloženého souboru, dále informace o datumu a čase přidělení, převzetí nebo vyřešení incidentu.
tblBugsWorkers – vazební tabulka obsahující informace o jednotlivých pracovnících, kteří se podíleli na vyřešení incidentu.
tblPriceList – tabulka s finančním ohodnocením jednotlivých typů úkolů a testů.

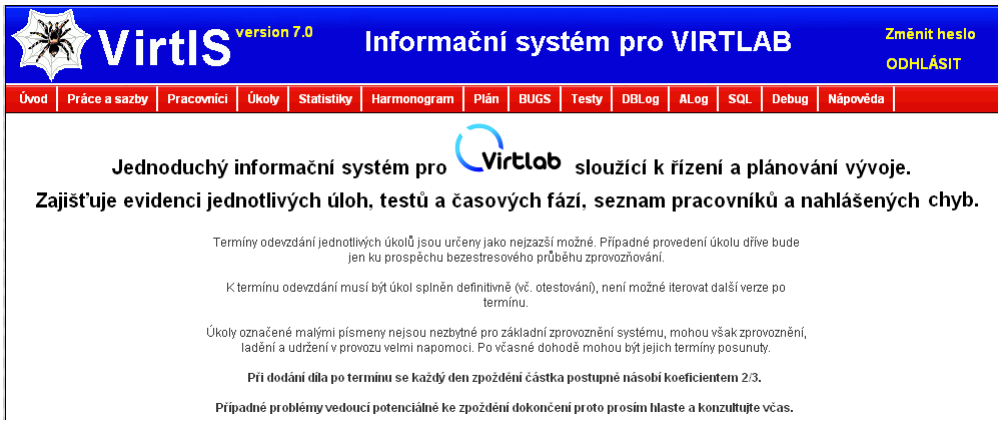
E-R diagram se všemi atributy



obrázek 20: E-R diagram se všemi atributy

6.2.3. Implementace

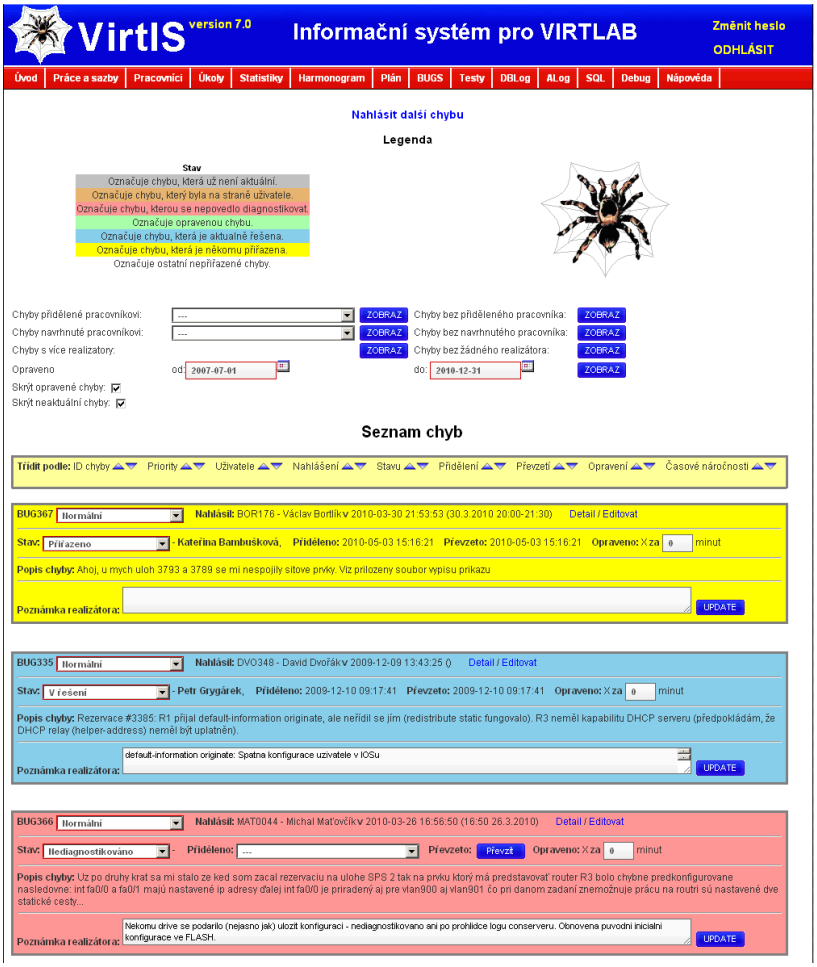
Informační systém VirtIS (viz obrázek 21) byl rovněž implementován v jazyce PHP. Rozšíření o Bug tracking systém spočívalo v doprogramování dalšího modulu. Ten se nachází v souborech bugs.php a bugsworkers.php. Obě tyto části využívají soubor funkcí bugs_fce_inc.php pro práci s chybami, dále soubor file_fce_inc.php, který obsahuje sadu funkcí pro práci se soubory a soubor worker_fce_inc.php, který obsahuje sadu funkcí pro práci s pracovníky (skuteční a navrhovaní realizátoři a testeři).



obrázek 21: VirtIS - informační systém pro Virtlab

Seznam chyb (viz obrázek 22) se eviduje obdobně jako seznam úkolů. Chyby se dají různě třídit podle jednotlivých atributů, zobrazovat pouze chyby přidělené konkrétnímu uživateli nebo chyby, které nejsou nikomu přiděleny. Dále se dají různě filtrovat již opravené nebo neaktuální chyby.

Po přihlášení mají jednotliví realizátoři možnost měnit stav chyb, případně k nim dopisovat poznámky. Při zobrazení detailu chyby realizátor navíc vidí: lokalitu, ze které byla chyba nahlášena, uživatelův email, kdyby bylo potřeba něco zkonzultovat, a v neposlední řadě přiložené soubory.



obrázek 22: VirtIS - výpis seznamu chyb

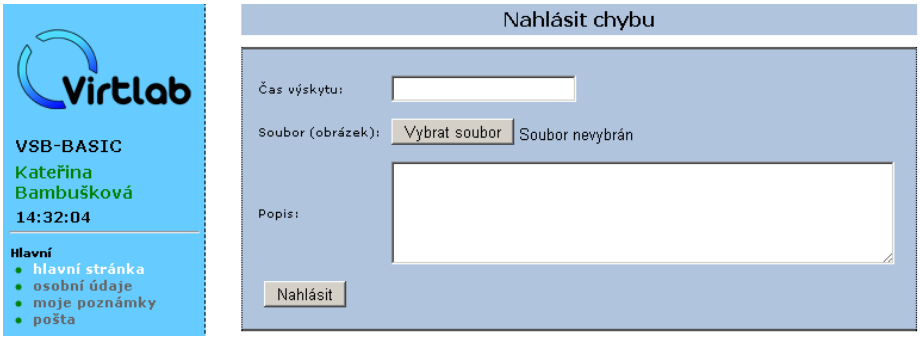
6.2.4. Propojení s GUI Virtuální laboratoře

Aby si uživatelé nemuseli zvykat na nové prostředí, byl odesílací formulář naimplementován do samotného uživatelského rozhraní Virtuální laboratoře (viz obrázek 23).

Jediné informace, které musí uživatel zadat jsou: čas výskytu – chyba se mohla vyskytnout mnohem dřív než se jí uživatel rozhodl nahlásit a uživatelův popis problému. Případně může přiložit soubor, například obrázek se screenem obrazovky.

Veškeré další informace, jako jsou: uživatelské jméno, email a lokalita se předá automaticky.

Všechny tyto informace se následně předají pomocí metody POST samotnému VirtISu, ten tyto informace zpracuje a pomocí metody GET odešle výsledek akce. Na základě toho vypíše řídící aplikace Virtlabu uživateli informaci o úspěchu či neúspěchu při nahlášení chyby.



obrázek 23: formulář pro chlášení chyb

6.2.5. Testování a nasazení do provozu

Po základním testování, které spočívalo převážně ve správné komunikaci mezi dvěma aplikacemi, byl celý systém nasazen do produkční verze Virtlabu.

Během provozu bylo do dnešních dnů nahlášeno 289 incidentů, z toho bylo 208 úspěšně opraveno, 4 již nejsou aktuální, ve 24 případech se jednalo o chybu uživatele, v 17 případech se chybu nepodařilo diagnostikovat, 16 incidentů je ještě v řešení, 8 jich bylo přiřazeno konkrétním vývojářům a zbylých 12 incidentů zatím nikomu nebylo přiděleno.

Nejvíce incidentů 239 bylo nahlášeno z Ostravské lokality, 2 incidenty přišli z lokality v Karviné a zbylých 48 bylo nahlášeno přes webové rozhraní VirtISu.

Celkem incidenty nahlásilo 123 různých uživatelů.

7. Závěr

Cílem této práce bylo rozšíření řídícího softwaru Distribuované virtuální laboratoře počítačových sítí o řadu nových funkcí a subsystémů.

Jednalo se zejména o možnost specifikovat zadání k laboratorním úlohám ve více jazycích. Tento bod byl splněn v plném rozsahu, včetně podpory prezentace zadání v jazyce preferovaném uživatelem.

Další část práce popisuje mechanizmus spolupráce uživatelů z různých lokalit na společně zarezervované úloze. Použitou technologii SOAP následně dále využívá Pavel Burda, při implementaci konektorů pro dynamické propojování topologií a Jiří Knapek, při integraci simulátoru „PacketTracer“ do Virtlabu. Oba v rámci svých diplomových prací [12] a [13].

Dále byl navržen a implementován systém pro nahrávání konfigurací do laboratorního vybavení před začátkem rezervace, a to včetně zařízení, která jsou studentům nepřístupné. Nad rámec zadání byl implementován archív uživatelských konfigurací, který volně navazuje na problematiku předkofigurací laboratorního vybavení.

Jednotlivá řešení byla implementována jako součást grantu Fondu rozvoje vysokých škol č. 1212/2008 s názvem: „Implementace nových prvků distribuované virtuální laboratoře počítačových sítí a rozšíření souboru laboratorních úloh“.

Pomocné utility pro podporu administrace systému, konkrétně se jedná o Generátor konfigurací a VirtIS s Bug tracking systémem, byly implementovány v rámci rozšíření navržených v projektu Fondu rozvoje sdružení Cesnet č.280/2008 s názvem: „Optimalizace správy distribuované virtuální laboratoře počítačových sítí“.

Nad rámec zadání byl dále zpracován podrobný uživatelský manuál a manuál pro tvůrce úloh. Tyto manuály jsou součást řešení grantu Fondu rozvoje vysokých škol č.1557/2009 s názvem "Rozšíření využití distribuované virtuální laboratoře počítačových sítí ve výuce".

Veškerá řešení, zde uvedená a řádně otestovaná, byla úspěšně nasazena do produkčního prostředí během let 2008-2010. Většina z nich se intenzivně používá při práci se systémem Distribuované virtuální laboratoře počítačových sítí.

Do budoucna plánuji i nadále spolupracovat s členy vývojového týmu, podílet se na dalším rozvoji Virtuální laboratoře, včetně setrvání v provozním týmu Virtlabu.

Literatura a informační zdroje

[1] NĚMEC, Pavel. Virtuální síťová laboratoř. [s.l.], 2005. VŠB-TU Ostrava. Diplomová práce.

[2] Oficiální stránky projektu Virtlab [online].
Dostupné na URL: http://www.virtlab.cz. (2009-2010).

[3] FILIPEC, Zdeněk. Automatizace hodnocení konfigurací v Distribuované virtuální laboratoři počítačových sítí. [s.l.], 2009. 69 s. VŠB-TU Ostrava. Diplomová práce.

[4] VAVŘÍČEK, Jan. Rozvoj řídícího software virtuální laboratoře počítačových sítí. [s.l.], 2007. 66 s. VŠB-TU Ostrava. Diplomová práce.

[5] ZAPLETAL, Karel. Soubor úloh pro virtuální laboratoř počítačových sítí. [s.l.], 2009. 59 s. VŠB-TU Ostrava. Bakalářská práce.

[6] SOLODUJEV, Miroslav. Technické zajištění pro výuku kurzů bezpečnosti počítačových sítí. [s.l.], 2009. 41 s. VŠB-TU Ostrava. Bakalářská práce.

[7] SALOŇ, Zbyněk. Technické zajištění pro výuku kurzu Network Security 2. [s.l.], 2008. VŠB-TU Ostrava. Bakalářská práce.

[8] Manuál k SOAP v PHP [online].
Dostupný na URL: http://www.php.net/manual/en/book.soap.php (2009).

[9] BORTLÍK, Václav. Distribuované spojovací pole pro virtuální laboratoř počítačových sítí. [s.l.], 2008. VŠB-TU Ostrava. Bakalářská práce.

[10] NOVÁK, Radek. Robustní reimplementace prototypového řešení serverových komponent systému Virtlab. [s.l.], 2010. VŠB-TU Ostrava. Diplomová práce.

[11] Manuál k socketům v PHP [online].
Dostupný na URL: http://php.net/manual/en/book.sockets.php (2010).

[12] BUDRA, Pavel. Implementace konektorů pro dynamické propojování distribuovaných virtuálních topologií v systému Virtlab. [s.l.], 2010. VŠB-TU Ostrava. Diplomová práce.

[13] KNAPEK, Jiří. Integrace distribuované laboratoře počítačových sítí se simulátorem PacketTracer. [s.l.], 2010. VŠB-TU Ostrava. Diplomová práce.

Přílohy

Příloha A

Konkrétní SQL skript pro vytvoření nové vazební tabulky pro zadání v různých jazycích:

```
CREATE TABLE `tasks_specification`
(
     `task_id` int(11) NOT NULL COMMENT 'task identifier',
     `lang` char(3) NOT NULL COMMENT 'language identifier',
     `file_id` int(11) default NULL COMMENT 'file identifier',
     KEY `file_id` (`file_id`),
     KEY `lang` (`lang`),
     KEY `task_id` (`task_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE `tasks_specification`
ADD CONSTRAINT `tasks_specification_ibfk_1` FOREIGN KEY (`task_id`)
REFERENCES `tasks` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT `tasks_specification_ibfk_2` FOREIGN KEY (`lang`)
REFERENCES `languages` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT `tasks_specification_ibfk_3` FOREIGN KEY (`file_id`)
REFERENCES `files` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Konkrétní SQL skript pro vytvoření nové vazební tabulky pro vzdálené uživatele:

```
CREATE TABLE `reservation_remote_users`
(
     `resID` int(11) NOT NULL default '0' COMMENT 'reservation ID',
     `user_id` varchar(20) NOT NULL default '' COMMENT 'user ID',
     `site_id` varchar(20) NOT NULL default '' COMMENT 'site ID',
     PRIMARY KEY  (`resID`,`user_id`,`site_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE `reservation_remote_users`
ADD CONSTRAINT `reservation_users_remote_ibfk_1` FOREIGN KEY (`resID`)
REFERENCES `reservations` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Konkrétní SQL skript pro vytvoření nové tabulky pro archív konfigurací:

```
CREATE TABLE `user_config_archive`
(
     `id` int(11) NOT NULL auto_increment COMMENT 'identifier',
     `user_id` varchar(20) NOT NULL default '' COMMENT 'user id',
     `resID` int(11) NOT NULL default '0' COMMENT 'reservation id',
     `task_id` int(11) NOT NULL COMMENT 'task identifier',
     `name` varchar(255) NOT NULL COMMENT 'user name configuration',
     `config` longtext NOT NULL COMMENT 'configurations',
     PRIMARY KEY  (`id`),
     KEY `user_id` (`user_id`),
     KEY `resID` (`resID`),
     KEY `task_id` (`task_id`)
)
ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE `user_config_archive`
ADD CONSTRAINT `user_config_archive_ibfk_1` FOREIGN KEY (`resID`)
REFERENCES `reservations` (`id`) ON DELETE CASCADE,
ADD CONSTRAINT `user_config_archive_ibfk_2` FOREIGN KEY (`user_id`)
REFERENCES `users` (`id`) ON DELETE CASCADE,
ADD CONSTRAINT `user_config_archive_ibfk_3` FOREIGN KEY (`task_id`)
REFERENCES `tasks` (`id`) ON DELETE CASCADE;
```

Příloha B

Taskupload.rng - Relax schéma pro upload úloh

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- RELAX NG schema pro popis ulohy -->
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
         xmlns:html="http://www.w3.org/1999/xhtml"
         datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

  <start><!-- zacatek -->
    <element name="task">
        <attribute name="name"><!-- atribut "name" -->
          <data type="string">
            <param name="minLength">1</param>
            <param name="maxLength">35</param>
          </data>
        </attribute>

        <optional><!-- nepovinny atribut "time" -->
          <attribute name="time">
            <data type="decimal">
              <param name="minInclusive">0</param>
              <param name="maxExclusive">10000000</param>
            </data>
          </attribute>
        </optional>

        <!-- Element popisující jméno v dlouhém formátu -->
        <element name="longname">
            <data type="string">
              <param name="minLength">1</param>
              <param name="maxLength">150</param>
            </data>
        </element>

        <element name="description"><!-- Element pro popis -->
            <data type="string">
              <param name="minLength">1</param>
              <param name="maxLength">250</param>
            </data>
        </element>

        <element name="specification"><!-- Popis zadani -->
          <zeroOrMore>
            <ref name="file" />
          </zeroOrMore>
        </element>

      <element name="picture"><!-- Popis obrazku -->
        <optional>
          <ref name="file" />
        </optional>
      </element>
```

```
      <element name="pre_conf"><!-- Popis preconf -->
          <optional>
          <ref name="file" />
        </optional>
      </element>
      <element name="post_conf"><!-- Popis postconf -->
        <optional>
          <ref name="file" />
        </optional>
      </element>
      <element name="topology"><!-- Popis topology -->
        <optional>
          <ref name="file" />
        </optional>
      </element>
      <element name="dia_picture"><!-- Popis dia obrazku -->
        <optional>
          <ref name="file" />
        </optional>
      </element>

    </element>
  </start><!-- konec hlavni casti -->

  <!-- nasleduje definice pouzitych elementu a atributu -->
  <define name="file">
    <element name="file">
          <attribute name="name">
            <data type="string">
              <param name="minLength">1</param>
              <param name="maxLength">49</param>
            </data>
          </attribute>
          <optional>
          <attribute name="lang">
            <data type="string">
              <param name="minLength">1</param>
              <param name="maxLength">5</param>
            </data>
          </attribute>
          </optional>
          <attribute name="filepath">
            <data type="string">
              <param name="minLength">1</param>
              <param name="maxLength">299</param>
            </data>
          </attribute>
          <attribute name="exists">
            <choice>
              <value>yes</value>
             <value>no</value>
            </choice>
          </attribute>
      </element>
  </define>

</grammar><!-- konec RELAX NG schematu -->
```

Příloha C

Ukázka souboru s předkonfiguracemi

```
<PC1>
root
ifconfig #PC1:line1# 111.1.1.5 netmask 255.255.255.0
route add default gw 111.1.1.1
</PC1>

<PC2>
root
ifconfig #PC2:line2# 222.2.2.5 netmask 255.255.255.0
route add default gw 222.2.2.1
</PC2>

<RA>
enable
conf t
interface #RA:line1#
ip address 111.1.1.1 255.255.255.0
no shutdown
exit
interface #RA:line2#
ip address 222.2.2.1 255.255.255.0
no shutdown
exit
</RA>
```

Příloha D

Část uživatelského manuálu týkající se archívu konfigurací.

Archív uživatelských konfigurací

Pokud řešíte složitější úlohu, která se skládá z více části, které jsou rozdělený na více rezervací, můžete využít funkce archívu uživatelských konfigurací. Při řešení úlohy tak můžete načíst jednotlivé konfigurace z laboratorního vybavení a ty nasledně uložit pod vlastním názvem. Takovýchto načtených konfigurací můžete mít libovolné množství. V případě, že si budete stejnou úlohu rezervovat znovu, máte k dispozici na výběr všechny vámi uložené konfigurace k dané úloze. Stejně tak můžete zarezervovat úlohu v původním stavu, pokud nezvolíte žádnou z vašich uložených konfigurací.

Načtení konfigurace

Pokud máte aktivní zarezervovanou úlohu (funkce archívu konfigurací prozatím **není dostupná** pro **topologie na přání**) v rámci dostupných úloh v systému, máte k dispozici funkci **Archive Devices Configurations** (viz Obrázek 21).

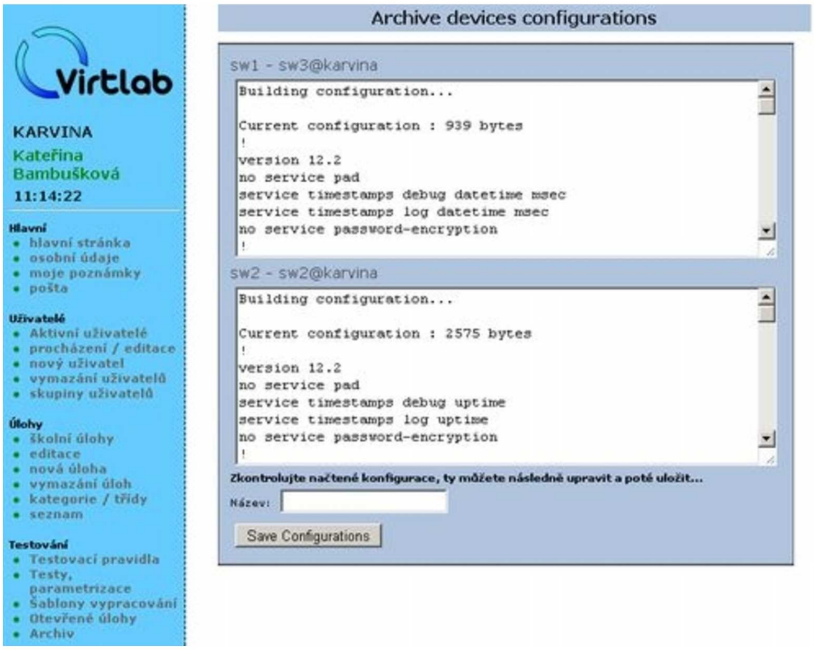
Obrázek 21

Pokud máte jednotlivé laboratorní prvky nakonfigurované a chcete si jejich konfiguraci uložit na příště, stačí si vybrat ty, ze kterých budete chtít vyčíst konfiguraci a operaci načtení spustíte tlačítkem **Read Configurations**.

Získat konfiguraci lze pouze z laboratorního vybavení typu **switch, router** a jedná se o výpis příkazu **show run.** Délka této operace je závislá na počtu zvolených laboratorních zařízení a může trvat od desítek sekund až po několik minut.

Editace a uložení konfigurace

Poté, co skončí operace načítaní konfigurací z jednotlivých laboratorních zařízení, můžete si před samotným uložením načtené konfigurace upravit (viz Obrázek 22). Při procesu ukládání konfigurace dochází k přemapování fyzických jmen rozhraní na logická jména, která se při nové rezervaci opět přemapují na fyzická jména rozhraní nově zarezervovaného laboratorního vybaveni.

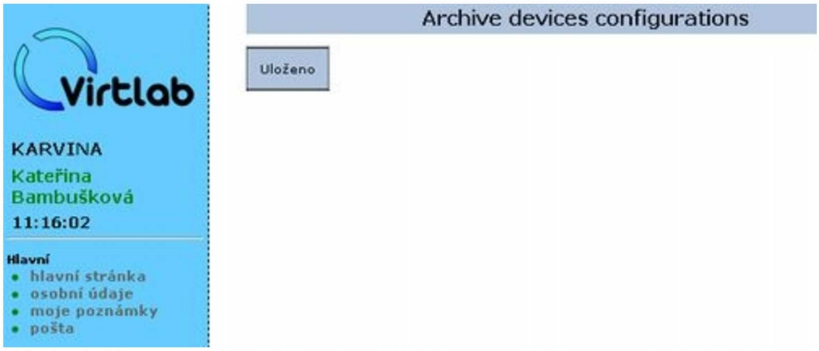


Obrázek 22

Může se také stát, že se některá konfigurace z laboratorního zařízení nepodaří načíst. V tomto případě doporučuji nejprve přistoupit na laboratorní zařízení přes JavaApplet a konzoli laboratorního zařízení „rozklepat“ a následně zkusit celou operaci načtení konfigurací zopakovat.

Jednotlivé konfigurace se ukládají hromadně ze všech zvolených laboratorních prvků do jednoho předkonfiguračního souboru pod vašim **uživatelským názvem** a jsou svázány s konkrétní úlohou. Je proto třeba zvolit všechna zařízení, která chcete mít příště předkonfigurovaná. Ke každé úloze můžete ukládat libovolné množství konfigurací.

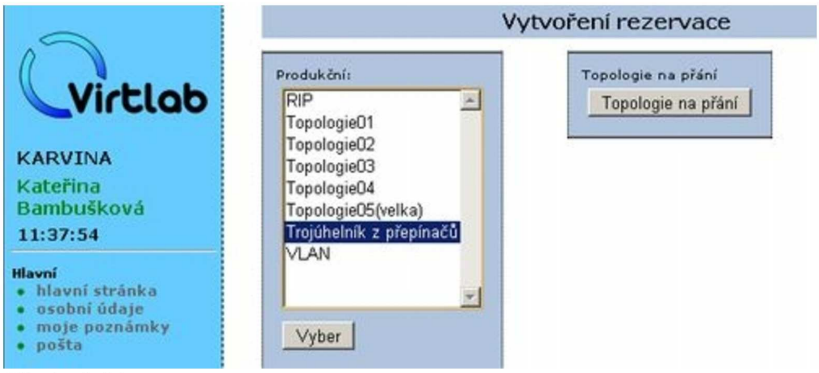
Konfigurace uložíte do archívu pomocí tlačítka **Save Configurations**. O výsledku operace budete informováni (viz. Obrázek 23).



Obrázek 23

Rezervace úlohy s uživatelskými konfiguracemi

Pokud si budete znovurezervovat nějakou úlohu některým ze standardních způsobů, například ze seznamu úloh (viz. Obrázek 24), budete mít následně k dispozici seznam vámi uložených konfigurací k této konkrétní úloze (viz Obrázek 25).



Obrázek 24

Obrázek 25

Úlohu rezervujete standardním způsobem, tak jak jste již zvyklí (od kdy, do kdy). Navíc máte k dispozici seznam **Vámi uloženách konfigurací ke zvolené úloze**.

Pokud žádnou předkonfiguraci na laboratorních prvcích nechcete, můžete vybrat možnost **Bez předkonfigurace** nebo nezvolit žádnou z uložených konfigurací. Stejným způsobem postupujte i v případě, že chcete použít standardní předkonfiguraci definovanou k úloze (pokud jí obsahuje).

Po zvolení vámi požadované možnosti můžete úlohu zarezervavat pomocí tlačítka **Rezervovat**. Z archívu konfigurací se připraví předkonfigurační soubory, které vznikají z vámi uložené konfigurace a přemapováním logických jmen rozhraní na fyzická jména rozhraní nově zarezervovaného laboratorrního vybavení. Ty jsou před začátkem rezervace poslány na laboratorní vybavení.

Příloha E

Skript pro stahování vygenerovaných konfiguračních souborů

```
#!/bin/bash

############################################################
#
#   C O N F I G   D O W N L O A D E R
#   ----------------------------
#   Description: Downloader config file for VIRTLAB
#   Author: Katerina Bambuskova
#
############################################################

# CONFIG BASE PATH
CONFIG_PATH="/etc/virtlab"

# CONFIG FILES
configarray=("conf-servers.conf"
             "erase-servers.conf"
             "tunservers.conf"
             "soap-servers.conf"
             "spoje.conf"
             "portsetter.conf"
             "asssk.conf"
             "localvlans.conf"
             "localserials.conf"
             "cons-devices.conf"
             "cons-servers.conf"
             "cons-filters.conf"
             "erase-device.conf"
             "uploads.conf"
             "rsv-server.conf"
             "rsv-ip.conf"
             "vybaveni.xml"
             "serialport.ini")

echo "             C O N F I G   D O W N L O A D E R"
echo "          ---------------------------------------"

if [ $# -lt 1 ]; then
  echo "Please restart CONFIG DOWNLOADER with correct syntax"
  echo "DOWNLOAD ->"
  echo "syntax: config-downloader.sh <sitename>"
  echo "syntax: config-downloader.sh <sitename> <configname> [sequence num]"
  echo "HELP ->"
  echo "syntax: config-downloader.sh help"
  echo "------------------------------------------------------------"
else
  if [ $1 == "help" ]; then
    echo "DOWNLOAD ->"
    echo "syntax: config-downloader.sh <sitename>"
    echo "syntax: config-downloader.sh <sitename> <configname> [sequence num]"
    echo "---------------------------------"
    echo "Config files name:"
    echo "---------------------------------"
    for config in ${configarray[*]}; do
      echo "$config"
    done
    echo "---------------------------------"
```

```
    echo "all - For ALL config files"
  else
    if [ $# -lt 2 ] || [ $2 == "all" ]; then
      for config in ${configarray[*]}; do
        if [ $# -lt 3 ]; then
          wget --no-check-certificate -O $CONFIG_PATH/$config
"http://config.viakis.net/getfile.php?config=$config&site=$1"
        else
          wget --no-check-certificate -O $CONFIG_PATH/$config
"http://config.viakis.net/getfile.php?config=$config&site=$1&sequence=$3"
        fi
      done
      echo "------------------------------------------------------------"
      echo "                    COMPLETED :-)"
      echo "------------------------------------------------------------"
    else
      test="false"
      for config in ${configarray[*]}; do
        if [ $2 == $config ]; then
          test="true"
        fi
      done
      if [ $test == "true" ]; then
        if [ $# -lt 3 ]; then
          wget --no-check-certificate -O $CONFIG_PATH/$2
"http://config.viakis.net/getfile.php?config=$2&site=$1"
        else
          wget --no-check-certificate -O $CONFIG_PATH/$2
"http://config.viakis.net/getfile.php?config=$2&site=$1&sequence=$3"
        fi
        echo "------------------------------------------------------------"
        echo "                      COMPLETED :-)"
        echo "------------------------------------------------------------"
      else
        echo "------------------------------------------------------------"
        echo "    $2 -> UNKNOWN CONFIG FILENAME !!!"
        echo "    HELP ->"
        echo "    syntax: config-downloader.sh help"
        echo "------------------------------------------------------------"
      fi
    fi
  fi
fi
```

Příloha F

Obsah přiloženého CD