VŠB - Technická univerzita Ostrava
Fakulta Elektrotechniky a informatiky
Katedra informatiky

Řízené sdílení popisů experimentů mezi lokalitami virtuální síťové laboratoře Virtlab

Controlled Sharing of Experiment Descriptions between Sites of Virtual Networking Laboratory Virtlab

2010 Bc. Roman Krhovják

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5.5.2010 . . . . . .

Mé poděkování patří především vedoucímu této diplomové práce Ing. Petru Grygárkovi za to, že mne trpělivě vedl od počátku práce až do jejího konce. Dále bych chtěl poděkovat některým členům vývojového týmu Virtlabu, kteří neváhali poskytnout mi v nouzi tolik potřebnou radu – a to především Bc. Kateřině Bambuškové a Ing. Zdeňku Filipcovi. Také nesmím opomenout poděkovat tvůrci nové řídící aplikace, se kterým jsem úzce spolupracoval, a to Martinu Gramesovi. Chci vyjádřit díky i rodině a přátelům, kteří mne podporovali jinými prostředky.

Abstrakt a klíčová slova

Abstrakt:

Cílem této práce je zanalyzovat, navrhnout řešení a implementovat do distribuované virtuální laboratoře Virtlab možnost sdílení úloh mezi lokalitami. Sdílení úloh je realizováno pomocí uživatelských skupin – každé skupině je přidělen seznam dostupných úloh. Každá lokalita má svůj strom skupin, ve kterém jsou jak skupiny lokální, tak i odkazy na skupiny ze vzdálených lokalit.

Je tedy třeba upravit tzv. řídící server Virtlabu. Ten je v tomto roce přepisován do nové podoby. Bylo dohodnuto, že tato úpravu bude implementována pro tuto novou podobu řídícího serveru.

Klíčová slova:

Virtlab, SOAP, Nette, PHP

Abstract and key words

Abstract:

Goal of this diploma thesis is to analyse, design and implement solution for possibility of task sharing between localities in virtual laboratory Virtlab. Sharing is realised by user groups – a list of available tasks is assigned to each group. Each locality has its own tree of user groups which includes local groups as well as groups from remote localities.

It is necessary to edit so called control server of Virtlab. The server is actually rewritten to new version. It was settled on that this diploma will be implemented for the new version of control server.

Key words:

Virtlab, SOAP, Nette, PHP

Seznam použitých zkratek a symbolů

AJAX	- Asynchronous JavaScript and XML
HTML	- Hyper-text Markup Language – značkovací jazyk pro tvorbu internetových stránek
URL	- Unique Resource Locator – jednoznačné určení zdroje
Virtlab	- distribuovaná virtuální laboratoř
PHP	- Hypertext Preprocessor, skriptovací programovací jazyk
SOAP	- Simple Object Access Protocol, protokol pro výměnu zpráv přes síť
ORM	- Object-relational mapping, objektově-relační mapování
MVP	- Model-View-Presenter, třívrstvý architektonický vzor aplikace
ERD	- Entity relationship diagram, diagram databáze
VŠB-TUO	- Vysoká škola báňská – Technická univerzita Ostrava
SEO	- Search Engine Optimization, optimalizace pro internetové full-text vyhledávače
DRY	- Don't repeat yourself, princip vývoje software snažící se omezit duplicitu kódu
KISS	- Keep it short and simple, princip upřednostňující jednoduchost před robustností
SQL	- Structured Query Language, strukturovaný dotazovací jazyk
XML-RPC	- eXtensible Markup Language – Remote Procedure Call
WSDL	- Web Service Definition Language
CSV	- Comma Separated Values
HTTPS	- Hypertext Transfer Protocol Secure

Obsah

1 Úvod

Distribuovaná virtuální síťová laboratoř (dále Virtlab) je z pohledu studenta internetová aplikace, která mu umožňuje rezervovat si dostupné síťové prvky a vzdáleně na nich pracovat - konfigurovat, testovat. Aby mohl student do Virtlabu přistupovat, musí mu být vytvořen v této aplikaci účet. Po přihlášení do systému si může zobrazit dostupné úlohy a některou si zarezervovat. Úloha je obvykle zadána svým popisem, obrázky a topologií. Po rezervaci úlohy se na zadaný čas vyhradí studentovi síťové prvky specifikované v úloze. V určený čas pak může student na úloze pracovat.

Slovo distribuovaná v celém názvu Virtlabu značí, že neexistuje jen jedna instalace tohoto programu. Původně byl Virtlab nasazen jen na VŠB-TUO, nyní se nasazuje i na jiných školách a uvažuje se i o nasazení v komerční sféře. Jedna tato distribuce Virtlabu se nazývá lokalita.

Lokality byly původně od sebe absolutně izolované. Nyní je možnost sdílet síťové prvky mezi lokalitami, studenti tak mají k dispozici větší množství prvků. Nebyla ovšem žádná možnost sdílet úlohy. Proto vznikla úprava, která toto umožňuje, ale jen dost omezujícím způsobem. Uživatel musí na své lokalitě rezervovat úlohu a poté má možnost pozvat člena cizí lokality ke spolupráci na této úloze.

Začalo být také problémem, že všichni uživatelé lokality mají přístup ke všem úlohám. Nebyla možnost skrýt před uživatelem určité úlohy. Úlohy jsou určité know-how a není snadné nějaké smysluplné vymyslet. Některé úlohy slouží také k testování samotného Virtlabu, ty uživatelé vidět nemusí.

Z pohledu administrátora jsou uživatelé členy uživatelských skupin. Seznam těchto skupin má stromovou strukturu. Z těchto faktů a požadavků vznikla specifikace mé diplomové práce – zajistit možnost sdílení úloh mezi lokalitami a také možnost upravovat přístupová práva uživatelů k úlohám.

Nejdříve jsem ve spolupráci se členy vývojového týmu Virtlabu sepsal specifikaci požadavků a udělal analýzu možných řešení. Pak jsem vytvořil návrh řešení, který jsem následně implementoval.

Ze specifikace vyplynulo, že upravovat práva každého uživatele k dané úloze by bylo příliš pracné. Proto se využívá právě uživatelských skupin. Výhoda je také stromová struktura seznamu uživatelských skupin – lze využít dědičnosti. Každá úloha tedy obsahuje seznam uživatelských skupin, které k ní mají právo přistupovat.

Uživatelských skupin se využívá také ke sdílení úloh s ostatními lokalitami. Pokud chceme zpřístupnit nějakou úlohu vzdálené skupině, stačí nám připojit si tuto skupinu do našeho stromu uživatelských skupin. S touto připojenou skupinou lze pak pracovat obdobně jako se skupinou běžnou.

Není ovšem žádoucí, aby administrátoři vzdálených lokalit viděli celý seznam skupin lokality včetně jejich uživatelů. Proto je každé skupině přiřazen speciální příznak – rozsah platnosti. Ten může nabývat několika hodnot. Rozsah „veřejná“ znamená, že administrátoři vzdálených lokalit mohou s touto skupinou pracovat, jak bylo popsáno výše. Rozsah „privátní“ znamená, že vzdálení

administrátoři nemají o existenci této skupiny ponětí, pro ně neexistuje. Poslední možnou hodnotou rozsahu je „vzdálená“, kdy tato skupina je připojenou vzdálenou skupinou. Pro vzdálené administrátory je stejná jako skupina s rozsahem „privátní“.

Při specifikaci také vyplynula potřeba další úpravy. Doposud měli možnost upravovat zadání úlohy uživatelé s příznakem „správce úloh“. Toto řešení bylo dost jednoduché, bylo třeba ho upravit. Proto je nyní tvůrcem úlohy určen seznam osob, které mají právo úlohu upravovat. Tento seznam je podmnožinou uživatelů s příznakem „správce úloh“.

Virtlab se skládá z několika navzájem propojených celků nazvaných servery. Je třeba upravit jen jeden z nich a to server řídící. Ten je v současné době přepracováván do nové podoby. Nevyplatilo by se dělat úpravu pro stávající server, bylo tedy rozhodnuto, že se bude upravovat server nově vyvíjený.

2 Architektura Virtlabu

2.1 Popis Virtlabu

Historie distribuované virtuální laboratoře počítačových sítí sahá do roku 2005. Bylo potřeba najít řešení problému. Studenti měli omezenou možnost přístupu do síťové laboratoře, kde potřebovali řešit praktické úlohy zadávané v předmětu Počítačové sítě a jiných. Petr Grygárek tedy začal realizovat myšlenku *virtuální* laboratoře s pomocí svých diplomantů. Na dnešní podobě projektu se podílelo již několik desítek vývojářů, především z řad studentů.[1]

Každá lokalita je tvořena několika moduly, tzv. servery. Každý server spravuje jinou část systému. Nejdůležitější servery jsou:

Řídící server

Internetová aplikace, která umožňuje práci na Virtlabu – správu uživatelů a jejich skupin; tvorbu, prohlížení a rezervaci úloh a jiné činnosti prostřednictvím internetového prohlížeče. Řídící server spolupracuje se serverem rezervačním.

Rezervační server

Umožňuje rezervovat potřebné fyzické síťové prvky na určený čas, řídícímu serveru podává informace o dostupnosti síťových prvků.

Tunelovací server

Zajišťuje distribuovanost Virtlabu – přemosťuje přes internet provoz mezi síťovými prvky na jiných lokalitách. Uživateli se pak zdá, jako by prvky nebyly na vzdálené lokalitě, ale u něj v síti.

Konzolový server

Uživateli poskytuje způsob komunikace s konzolou síťových prvků. Na straně uživatele je Java applet, který komunikuje s konzolovým serverem, ten pak přeposílá požadavky přímo danému prvku.

Mým úkolem je úprava řídící aplikace, proto se budu nyní věnovat podrobněji jejímu popisu.

2.1.1 Současná podoba řídící aplikace

Aplikace je napsána v PHP, používá se databáze MySQL. Nebylo použito žádného frameworku, většina tříd nevyužívá objektově orientovaného programování[1]. Přístup k databázi MySQL je řešen přímo funkcemi jazyka PHP a jednoduchou knihovnou napsanou nad těmito funkcemi.

1 Výjimkou je například modul Automatizace hodnocení konfigurací od Zdeňka Filipce z roku 2009

2.1.2 Nová podoba řídící aplikace

Řídící aplikace je v současnosti přepisována do nové podoby[4], která využívá mnoho konceptů současného moderního programování. Aplikace je napsána v MVP frameworku Nette pod PHP 5.3. Pro přístup k databázi MySQL 5 se využívá PHP knihovna Dibi[2] a nad ní implementovaný ORM modul Ormion[3].

Nyní detailněji popíšu výše jmenované technologie.

PHP 5.3

Objektově orientovaný, skriptovací jazyk, překládaný za běhu na straně serveru. První stabilní 5.3 verze vyšla v červnu roku 2009.

Nette

Výkonný MVP framework pro PHP. Eliminuje bezpečnostní rizika, podporuje AJAX, SEO, DRY, KISS a znovupoužitelnost kódu. Poskytuje aplikaci zabezpečení, programátorům pak jedinečné ladící nástroje a množství pluginů. V České republice existuje kolem Nette rozsáhlá komunita, většina dokumentace je v češtině.[3]

MySQL 5

Moderní databázový systém s podporou transakcí, cizích klíčů, pohledů, uložených procedur i triggerů.[5]

Dibi

Abstraktní mini databázová knihovna pro PHP.

Ormion

Objektově-relační mapování pro Nette a Dibi. Ulehčuje práci s databází, mapuje tabulky databáze na PHP objekty.

2.2 Popis stávajícího řešení sdílení popisů experimentů

2.2.1 Architektura řešení

Stávající řešení popíšu ze dvou pohledů:

- funkčního – popis jeho možností z pohledu uživatele či administrátora

2 http://dibiphp.com/cs/
3 http://addons.nettephp.com/cs/ormion

- programátorského – jakým způsobem a jakými technikami je řešení implementováno

2.2.1.1 Funkční hledisko

Sdílení úloh bylo umožněno pouze pomocí rezervací. Lokální uživatel si musel vytvořit pro úlohu rezervaci a poté ji nasdílet uživateli jiné lokality. Ten se pak stal spolupracovníkem na této úloze.

2.2.1.2 Programátorské hledisko

Přenos informací mezi lokalitami probíhá pomocí protokolu SOAP. Soap klient volá na vzdáleném serveru funkce, které jsou zaregistrovány SOAP serverem. Seznam SOAP serverů a adres pro přístup k nim je uložen v konfiguračním souboru na relativní adrese */etc/virtlab/soap-servers.conf*. SOAP server je pak celý se všemi funkcemi implementován v jediném souboru – *soapserver.php*. Vzdálené funkce serveru jsou volány klientem přímo z PHP souborů. Funkci jsou předány parametry a vrácen je většinou HTML kód, který je pak přímo zobrazen.

2.2.2 Nedostatky řešení

Nejedná se v podstatě ani o sdílení popisů experimentů, ale o sdílení experimentů samotných. To znamená, že uživateli není umožněno si vzdálené úlohy prohlížet ani rezervovat, k tomu musí požádat uživatele vzdálené lokality o rezervaci a její nasdílení.

Řešení také neumožňuje upravovat přístupová práva k úlohám v rámci lokality – všichni uživatelé mají možnost prohlížet a rezervovat si všechny úlohy na dané lokalitě. Všichni správcové úloh mají možnost úlohu upravovat či mazat, toto nelze nijak omezit.

3 Specifikace požadavků a analýza řešení

Specifikace požadavků je výsledkem analýzy potřeb zadavatele, administrátorů a koncových uživatelů systému. Zadavatelem je vývojový tým Virtlabu v čele s Petrem Grygárkem, jehož požadavky byly zjišťovány a konzultovány na pravidelných schůzkách. Některých schůzek se účastnili i administrátoři a zároveň členové vývojového týmu Virtlabu (Kateřina Bambušková, Adam Janošek, Martin Milata, Zdeněk Filipec), kteří také přispěli svými požadavky k vytvoření specifikace.

Mým zadáním je upravit část řídící aplikace, proto budu specifikovat pouze tuto část.

3.1 Definice pojmů

Aby mohla být vypracována všem stranám srozumitelná specifikace, je třeba nejdříve definovat používané pojmy.

Lokalita

Virtlab není nainstalován pouze na VŠB-TUO. Jeho distribuce jsou i na jiných školách. Jedna tato instalace je nazývána lokalitou.

Úloha

V textu používaný výraz pro popis experimentu. Experiment je popsán několika parametry (název, popis, délka v minutách) a soubory (zadání úlohy, obrázek, před-konfigurace, ukázková konfigurace, popis topologie, obrázek topologie). Experiment si lze rezervovat na určitý čas – tím se pro tento čas vyhradí uživateli potřebné síťové prvky a uživatel může v této době na experimentu pracovat.

Lokální úloha

Úloha na lokalitě, na které je uživatel přihlášen.

Vzdálená úloha

Úloha na lokalitě, na které není uživatel přihlášen.

Roubování

Přidání ukazatele na vzdálenou veřejnou skupinu do vybrané lokální skupiny. Uživatelé této vzdálené skupiny pak mají stejná práva jako by byli ve skupině lokální.

Notifikace (zpráva, oznámení)

Je zpráva oznamující vzdálené lokalitě nějakou změnu na lokalitě stávající. Reakce vzdálené lokality

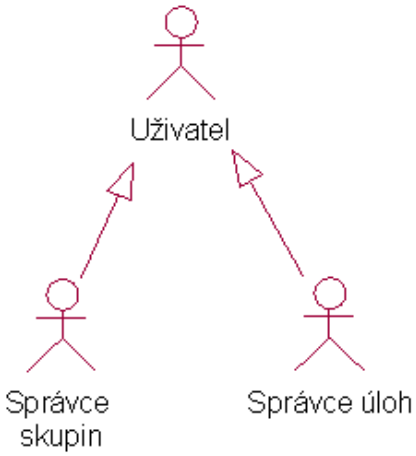
obvykle je, že pošle mail svým administrátorům, kde informuje o této změně.

3.2 Případy užití

K popisu specifikace se využívá diagramů případu užití (Use Case Diagrams). Ty jsou potom slovně vysvětleny a přiblíženy.

3.2.1 Aktéři

Nejprve popíšeme aktéry – uživatele systému. Ti zastávají určité role, mají svá práva a omezení. Často jsou role hierarchicky řazeny.



Uživatel

Uživatel je osoba, která má přístup do virtlabu, ale nemá žádná zvláštní práva.

Správce úloh

Správce úloh je osoba s právy uživatele, navíc oprávněná spravovat úlohy.

Správce skupin

Správce skupin je osoba s právy uživatele, navíc oprávněná spravovat skupiny uživatelů.

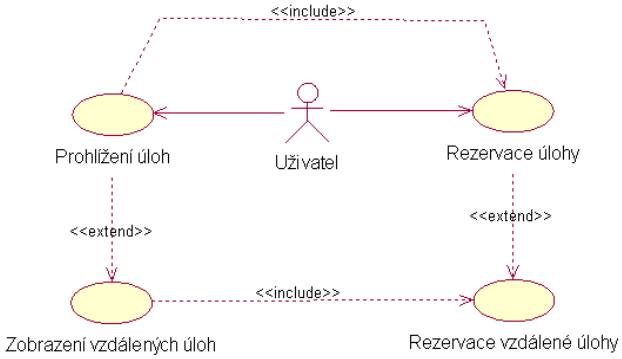
Obrázek 1: Diagram aktérů

V reálné aplikaci existuje administrátor, ten bude mít roli správce skupin. Také v aplikaci existuje správce úloh, který bude mít stejnojmennou funkci.

3.2.2 Případy užití

Každý aktér může provádět určité činnosti, ty shrnuje diagram případů užití a jejich popis.

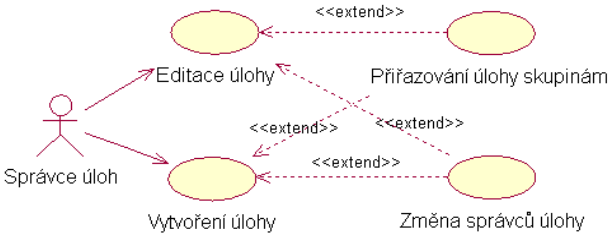
3.2.2.1 Diagramy případů užití z pohledu aktérů



Uživatel

Uživatel má možnost si zobrazit lokální i vzdálené úlohy, které jsou přístupné všem uživatelům lokality a také úlohy, které jsou mu přístupné díky členství ve skupinách. Pro vybranou úlohu si může rezervovat síťové prvky na určitý čas.

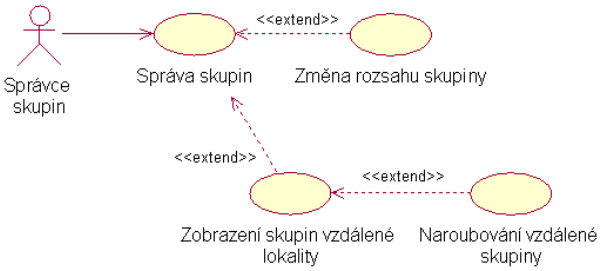
Obrázek 2: Use Case pro roli Uživatel



Obrázek 3: Use Case pro roli Správce úloh

Správce úloh

Spravuje úlohy: má právo vytvářet nové úlohy, měnit stávající úlohy a jejich správce, přiřazovat úlohy skupinám uživatelů.



Obrázek 4: Use Case pro roli Správce skupin

Správce skupin

Správce skupin má právo měnit strukturu stromu skupin. Vytváří či odebírá skupiny a uživatele z nich, roubuje vzdálené skupiny do stromu lokálních skupin.

3.2.2.2 Detailní popis případů užití

Prohlížení úloh

Každý uživatel systému má právo si prohlížet úlohy. Vidí ty úlohy, které jsou přiřazeny všem uživatelům dané lokality (ve stromu skupin jsou přiřazeny kořenu) a také ty, které jsou přiřazené skupinám jichž je členem.

Zobrazení vzdálených úloh

Uživatel si při prohlížení úloh může nechat zobrazit i vzdálené úlohy. Vidí ty vzdálené úlohy, které jsou přiřazeny skupinám jichž je členem.

Rezervace úlohy

Po výběru úlohy si uživatel vybere čas (od - do), na který by si chtěl úlohu rezervovat. Pokud jsou síťové prvky na tuto dobu dostupné, úloha mu bude rezervována.

Rezervace vzdálené úlohy

Probíhá stejně jako rezervace úlohy, jen si uživatel vybírá ze seznamu vzdálených úloh.

Změna rozsahu skupiny

Správce skupin může nastavit skupině rozsah platnosti na hodnoty:

- privátní
- veřejná

Rozsah s hodnotou *privátní* znamená, že ostatní lokality Virtlabu nevědí nic o této skupině.

Rozsah s hodnotou *veřejná* pak znamená, že ostatní lokality tuto skupinu vidí a mohou ji naroubovat do svého seznamu skupin.

Zobrazení skupiny vzdálené lokality

Při úpravě skupiny si může správce skupin nechat zobrazit seznam vzdálených lokalit a po výběru vzdálené lokality se mu zobrazí seznam veřejných skupin této lokality.

Naroubování vzdálené skupiny

Jakmile si správce skupin zobrazí skupiny vzdálené lokality, může naroubovat libovolné veřejné vzdálené skupiny na libovolné místo v lokálním stromu skupin.

Vytvoření úlohy

Správce úloh může vytvořit úlohu. Při vytváření může změnit seznam správců úlohy a také přiřadit úlohu libovolným skupinám.

Editace úlohy

Při editaci úlohy může správce dané úlohy upravovat seznam správců úlohy a také přiřadit upravovanou úlohu libovolným skupinám.

Přiřazování úlohy skupinám

Po výběru úlohy může dát správce úloh přístupová práva k ní libovolným skupinám (nebo také práva odebírat).

Změna správců úlohy

Po výběru úlohy může tvůrce úloh změnit její správce. Ti jsou vybíráni ze seznamu správců úloh. Buď vybere konkrétní správce nebo celý seznam správců.

3.3 Systémový návrh

Aby bylo možné vytvořit návrh implementace je nutné určit v jakém prostředí vyvíjený software poběží a zohlednit jeho omezení.

Řídící server Virtlabu běží na operačním systému typu Linux, pod serverem Apache. Řídící aplikace je vyvíjena v programovacím jazyku PHP verze 5.3 s MVP frameworkem Nette. Pro přístup k MySQL databázi se používá ORM Ormion napsaný nad databázovou knihovnou Dibi. Je možné využívat jazyk Javascript či technologii AJAX(J).

Uživatelé budou k aplikaci přistupovat prostřednictvím internetového prohlížeče, aplikace by tedy měla být optimalizována pro všechny používané druhy.

3.4 Analýza možností řešení

Ze specifikace požadavků vyplývá, že je potřeba výrazně pozměnit dva velké funkční celky:

- uživatele a jejich skupiny
- úlohy a možnost jejich sdílení

3.4.1 Uživatelé a jejich skupiny

Z konzultací a jejich následné analýzy vzešly dva různé návrhy řešení:

- oddělené globální a lokální skupiny
- roubovatelný strom

Popíšu zde oba návrhy. Více je ovšem propracován návrh roubovatelného stromu, protože byl nakonec vybrán k implementaci.

3.4.1.1 Oddělené globální a lokální skupiny

Lokální skupina uživatelů

Zůstane zachována. To znamená, že skupiny jsou řazeny do stromu. Kořen stromu obsahuje všechny uživatele lokality. Uživatel může být ve více skupinách.

Globální skupina

Zakladatel s právy administrátora může vytvořit globální skupinu. Při vytváření vybere lokality, ze kterých pak budou vybíráni uživatelé této skupiny. Uživatelé pak budou přidáni administrátory jednotlivých lokalit. Zakladatel si může vybrat, zda budou administrátoři emailem upozorněni o požadavku přidání uživatelů do globální skupiny.

Zakladatel může při vytvoření skupiny zvolit, zda přidání člena vyžaduje potvrzení zakladatelem. Pokud ano, může být o žádosti přidání člena (členů) informován emailem. Pokud ne,

může být emailem informován o přidání člena (členů) administrátorem do skupiny.

Zakladatel také může určit další správce globální skupiny. Ti budou mít obdobná práva jako on. Zakladatel může určit, kteří správcové budou moci přidávat další správce.

Zakladatel a správci mohou odebírat konkrétní uživatele či celou lokalitu ze skupiny. Mohou také skupinu uzavřít, což znamená, že již lokální administrátoři nebudou moci přidávat další uživatele.

Informace o globální skupině (seznam členů skupiny a seznam úloh, ke kterým mají členové přístup) jsou uloženy na všech lokalitách. Lokality jsou informovány o každé změně těchto informací konkrétní lokalitou.

Seznam globálních skupin nebude stromově uspořádaný (správa by byla příliš chaotická). Bude jednoúrovňový.

Skupina lokalit

Obdobná jako lokální skupina uživatelů. Strom lokalit, kořen jsou všechny lokality ze seznamu lokalit.

Skupina lokalit je uložena lokálně – každá lokalita má své vlastní skupiny (každý admin bude chtít své skupiny, jaké potřebuje).

Seznam lokalit

Seznam lokalit bude uložen lokálně. Při vytvoření lokality administrátoři přidají novou lokalitu do svého seznamu lokalit.

Sdílení úloh

Úloha může být přístupná pro:

- lokální skupiny uživatelů
- globální skupiny uživatelů
- lokality
- skupiny lokalit

Úlohy lze zpřístupnit i konkrétním uživatelům. Úlohu lze zpřístupnit vzdálené skupině uživatelů.

Úloha může být zpřístupněna určitým subjektem (skupinou, lokalitou...), ale také může být určitému subjektu zakázán přístup k této úloze.

3.4.1.2 Roubovatelný strom

- Je definováno, že kořen stromu uživatelských skupin obsahuje všechny uživatele lokality. Umožní to snadné naroubování celé lokality.

- Každá skupina by měla příznak *scope* určující zda je roubovatelná či ne.
  - Roubovatelnou skupinu (*public*) by viděly i ostatní lokality a správci by si ji mohli naroubovat do svého stromu
  - Neroubovatelná skupina (*private*) by byla jen pro účely konkrétní lokality
- Pokud by se roubovala skupina již obsahující nějaké roubované skupiny, tak by se tyto roubované skupiny nenaroubovaly (mohla by totiž vzniknout smyčka).

3.4.1.3 Srovnání obou možností

Oddělení globálních a lokálních skupin

Výhody:

- Konkrétní globální skupiny mohou být řízeny správci (i z různých lokalit). Ti pak můžou odebírat uživatele, zamknout globální skupinu či si vyžadovat potvrzení přidání každého uživatele.
- Globální a lokální skupiny budou striktně odděleny, možná o něco přehlednější a snažší k prezentaci.

Nevýhody:

- Lokální a globální skupina jsou podobné, ale v tomto případě budou všechny formuláře či funkce psány dvakrát. Myslím si, že by to šlo napsat i univerzálně.
- Neřeší problém nasdílení úloh pro vzdálené lokální skupiny. Šlo by vyřešit například přidáváním vzdálených lokálních skupin do globální skupiny. Potom by asi přibyl do lokálních skupin příznak *scope* jako u roubovatelného stromu.

Jeden roubovatelný strom

Výhody:

- jeden implementační celek
- pokrývá všechny možnosti sdílení

Nevýhody

- Stejně by se muselo nějak oddělit na prezentační vrstvě zobrazování globálních a lokálních skupin, vše by bylo zřejmě na jedné stránce - na méně stránkách než oddělené skupiny. Méně přehledné, ale více funkční a rychlejší na správu. Problém s oddělením lokálních a vzdálených skupin by se dal vyřešit různými barvami.

3.4.1.4 Srovnání z pohledu interakcí mezi lokalitami

Poznámka: ***Tučnou kurzívou je vyznačena interakce mezi lokalitami.***

Varianta: Oddělené lokální a globální skupiny

Pro globální skupinu:

Přidání skupiny

Uživatel dá vytvořit globální skupinu.

Uživatel vybere lokality, ze kterých budou mít uživatelé přístup.

Záznam o globální skupině se pošle všem vybraným lokalitám.

Administrátoři vzdálených lokálních skupin vyberou uživatele nové globální skupiny.

(*Varianta s lokálními skupinami uvnitř globální skupiny:* Administrátoři vyberou skupiny nové globální skupiny.)

Změněná globální skupina se přepošle všem vybraným lokalitám.

Zobrazení skupiny

Uživatel si vybere, kterou skupinu chce zobrazit.

Projde se globální skupina a zobrazí se její uživatelé.

(*Varianta s lokálními skupinami uvnitř globální skupiny*: pokud se v seznamu narazí na vzdálenou lokální skupinu, ***požádáme vzdálenou lokalitu o větev této skupiny***).

Zobrazí se všichni uživatelé.

Odstranění skupiny

Správce skupiny se rozhodne ji smazat.

Skupina se na lokální lokalitě odstraní a ***ostatním lokalitám se pošle žádost o její odstranění***.

Sdílení úloh

Uživatel vybere úlohu ke sdílení.

Ze stromu vybere větve skupin, které budou mít k úloze přístup.

Do access-listu úlohy se vybrané skupiny uloží.

Zobrazení uživateli dostupných úloh

Uživatel si nechá zobrazit dostupné úlohy.

Na lokální lokalitě se projde seznam lokálních i globálních skupin a zjistí se, kterých je uživatel členem.

Na všech lokalitách (cyklus):

> ***Lokalitě se pošle seznam skupin, ve kterých uživatel je.***
>
> Projdou se všechny access-listy všech úloh a zjistí se zda obsahují skupiny, ve kterých uživatel je.

Pokud obsahují, úloha se přidá do dostupných úloh.

Lokalitou je poslán seznam dostupných úloh.

Zobrazí se výsledný kompletní seznam dostupných úloh.

Varianta: Roubovatelný strom

Přidání skupiny

Uživatel vybere ve stromu skupinu, do které bude roubovat a zvolí, že chce přidat vzdálenou skupinu.

Zobrazí se seznam lokalit (uložen lokálně).

Uživatel vybere lokalitu.

Požádáme lokalitu o seznam roubovatelných skupin.

Zobrazí se seznam roubovatelných skupin.

Uživatel vybere skupiny k přiroubování.

Skupiny se uloží do stromu, přidá se koncovka @název_lokality

Zobrazení skupiny

Uživatel vybere skupinu, kterou chce zobrazit.

Projde se strom od vybrané skupiny dolů (větev).

Pokud narazíme na název_skupiny@název_lokality tak ***požádáme lokalitu o větev název_skupiny***.

Zobrazí se celá větev.

Odstranění skupiny

Vybraná skupina se prostě odstraní ze stromu.

Sdílení úloh

Uživatel vybere úlohu ke sdílení.

Ze stromu vybere větve skupin, které budou mít k úloze přístup.

Do access-listu úlohy se vybrané skupiny uloží.

Zobrazení uživatelovi dostupných úloh

Uživatel si nechá zobrazit dostupné úlohy.

Na každé lokalitě se provede:

Projde se celý strom skupin a zjistí se, v jakých skupinách uživatel je.

Projdou se všechny access-listy všech úloh a zjistí se zda obsahují skupiny, ve kterých uživatel je.

Pokud obsahují, úloha se přidá do dostupných úloh.

Lokalitou je poslán seznam dostupných úloh.

Zobrazí se výsledný kompletní seznam dostupných úloh.

Regulární výrazy

Byla zvažována možnost používat regulární výrazy při výběru skupin. Například pokud bychom chtěli vybrat všechny studenty, dali bychom zobrazit skupiny začínající na „stu_“.

Vzhledem k tomu, že bude používán strom skupin, kde všichni studenti mohou být v nadřazené skupině „studenti“ nemá toto praktický význam.

3.4.1.5 Vybraná varianta

Obě řešení byla shledána použitelnými, nicméně byl vybrán **roubovatelný strom**. A to především z důvodu jednotného přístupu k lokálním i globálním skupinám a výhodám z toho plynoucím. Sdílení úloh pomocí roubovatelného stromu se jeví jednodušším, přehlednějším i srozumitelnějším.

3.4.2 Úlohy a jejich sdílení

Úloha bude mít vlastníka.

Ten přiřazuje úloze:

- správce – Bude mít stejná práva jako vlastník – úprava úlohy, seznamu správců, seznamu oprávněných skupin.
- oprávněné skupiny – Skupiny, které mají k úloze přístup – mohou si ji prohlížet a případně rezervovat.

Momentální role „Správce úloh“ by se dala využít k tomu, že by to byl seznam potenciálních správců. Když správce úlohy vybírá další správce úlohy, tak by vybíral ze seznamu „Správců úloh“.

3.4.3 Komunikace mezi lokalitami - SOAP

SOAP je webová služba. Ty umožňují pomocí protokolu HTTP volat funkce na vzdáleném serveru. Formát výměny dat je obvykle XML.

Jednoduchý protokol pro práci s webovými službami se nazývá XML-RPC. Komplexnější je pak SOAP. K službě protokolu SOAP lze připojit WSDL dokument, který popisuje veškerou funkčnost webové služby.

PHP obsahuje zabudovanou extenzi PHP SOAP (třídy *SoapClient*, *SoapServer*, *SoapFault*, *SoapHeader*, *SoapParam*, *SoapVar*).[2, 6]

Princip serveru:

1. Vytvoření serveru (zadáme adresu serveru)
2. Napsání funkcí v PHP

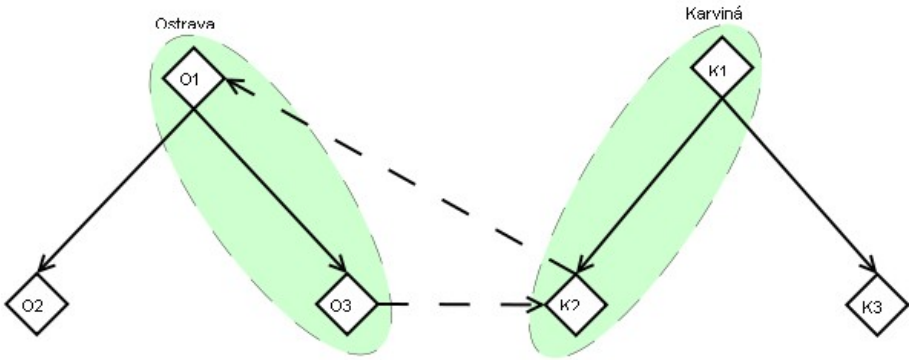
3. Zaregistrování funkcí k serveru

Princip klienta:

1. Zaregistrování se ke službě (zadáme adresu serveru)
2. Zavolání vzdálené funkce (s možností předání parametrů, funkce může i něco vracet)

3.5 Problémy roubovatelného stromu a jejich řešení

3.5.1 Cykly u více lokalit



Obrázek 5: Problém cyklu ve stromu skupin

Takto vypadá fyzicky nákres možnosti roubování skupin. Zeleně jsou označeny veřejné skupiny. Cyklus vzniká díky vzájemnému naroubování větví mezi dvěma lokalitami, v tomto případě vypadá cyklus takto: O1 → O3 → K2 → O1.

Problém se dá řešit tak, že budou do skupiny naroubovány jen skupiny dané vzdálené lokality. Například při roubování K2 na O3 se nebudou roubovat žádné skupiny z jiných lokalit než je Karviná. V tomto případě by se neroubovala skupina O1.

3.5.2 Notifikace

Je třeba vyřešit, jak a kdy budou administrátoři uvědoměni o změnách týkajících se naroubovaných skupin.

Notifikace při naroubování

Administrátoři lokality, z jejíhož stromu byla nějaká veřejná skupina naroubována, jsou uvědoměni o této akci. Stejně tak jsou uvědoměni při odstranění této skupiny ze stromu lokalit.

Notifikace při změně

Při úpravě veřejné skupiny, která je někam naroubována, je vzdáleným lokalitám tato akce oznámena. Lokality si pak projdou své stromy uživatelů a pokud mají tuto skupinu naroubovanou, pošlou upozornění o změně svým administrátorům. Za úpravu se považuje: přidání/odebrání uživatelů či podskupiny, odstranění této skupiny, změna rozsahu platnosti na *private*.

Při odstranění skupiny či změně scope na private si také vzdálená lokality odstraní tuto skupinu ze svého stromu lokalit.

3.6 Návrh grafického uživatelského rozhraní

Návrhem grafického rozhraní se ujasní požadavky na požadované funkce.

3.6.1 Seznam uživatelských skupin

Původní seznam skupin (obrázek 6) zobrazoval přehledně strom skupin. Při kliknutí na skupinu se zobrazil popis skupiny (obrázek 10). Upravená verze (obrázek 7) umožňuje navíc hromadnou změnu rozsahu platnosti či mazání skupin. Dále je v hranatých závorkách za názvem skupiny zobrazen počet jejích členů. Při najetí myší nad název skupiny se zobrazí její popis. Názvy skupin mají různé barvy pro různé rozsahy platností – šedě je privátní skupina, zeleně veřejná skupina a oranžově vzdálená skupina. V případě chyby u rozsahu platnosti se zobrazí název skupiny červeně. Tato situace by běžně neměla nastat.

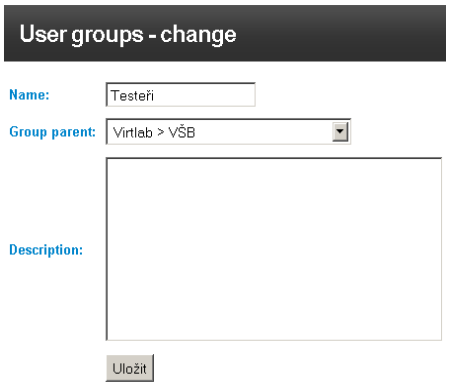
Obrázek 6: Původní seznam uživatelských skupin

Obrázek 7: Nový seznam uživatelských skupin

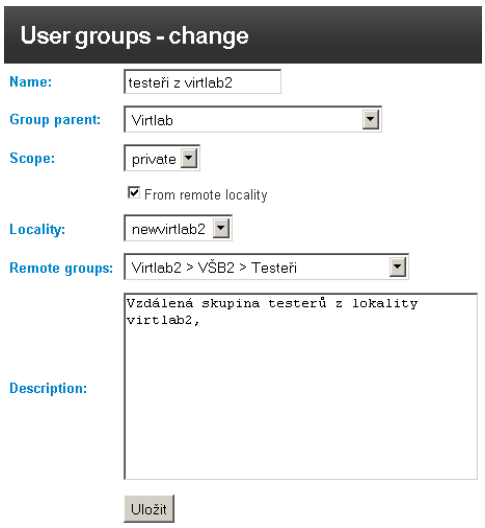
3.6.2 Úprava uživatelské skupiny

Při vytváření nové či úpravě stávající skupiny šel nastavit její název, popis a pozice ve stromu skupin (obrázek 8). Se zavedením rozsahu platnosti a vzdálených skupin je možné určit rozsah skupiny:

privátní nebo veřejná. Při zaškrtnutí tlačítka *Ze vzdálené lokality* se aktivuje seznam vzdálených lokalit a po výběru lokality se AJAXovým požadavkem načte seznam skupin dané vzdálené lokality. Úprava je zobrazena na obrázku 9.



Obrázek 8: Původní úprava uživatelské skupiny



Obrázek 9: Nové uživatelské skupiny

3.6.3 Popis uživatelské skupiny

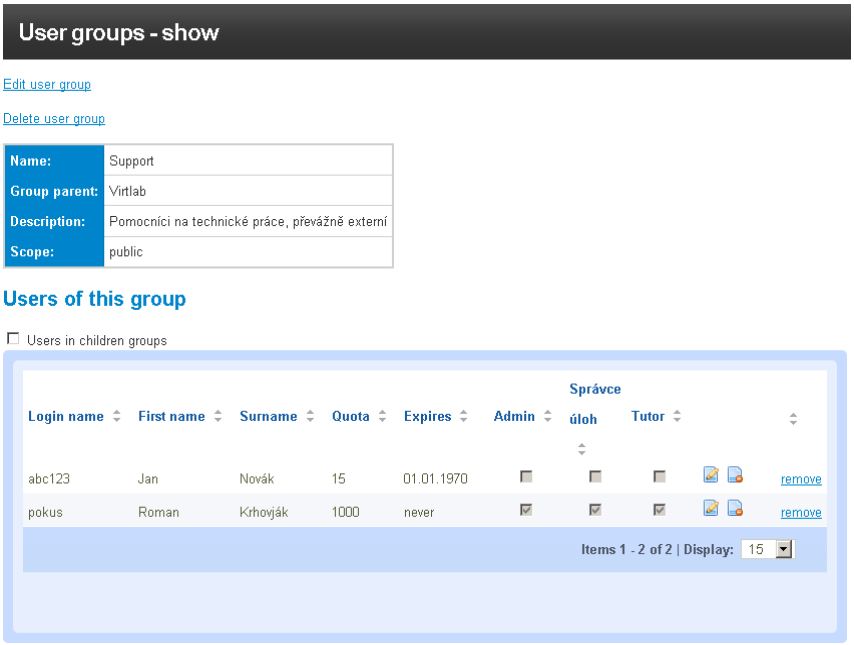
Původní popis uživatelské skupiny nebyl dostatečný – seznam uživatelů dané skupiny nepočítal s potomky dané skupiny, kteří mají také své uživatele. Neexistovala také žádná vzdálená skupina, proto je specifikováno jak vypadá její popis.

Popis lokální uživatelské skupiny

Popis skupiny zahrnoval její jméno, popis a seznam rodičovských skupin (Obrázek 10). Dále obsahoval prostý seznam členů. Nově (obrázek 11) je v popisu skupiny zobrazen i rozsah platnosti a upraven seznam členů. Je využita komponenta DataGrid[4], která již byla použita v seznamu uživatelů. Ta umožňuje řazení členů, jejich editaci a nově je implementována možnost odstranit členy z dané skupiny. Také přibylo nad seznam zaškrtávací tlačítko *I uživatelé z potomků*. To po zaškrtnutí odešle AJAXový požadavek, který aktualizuje seznam členů – přidá i členy ze skupin potomků.

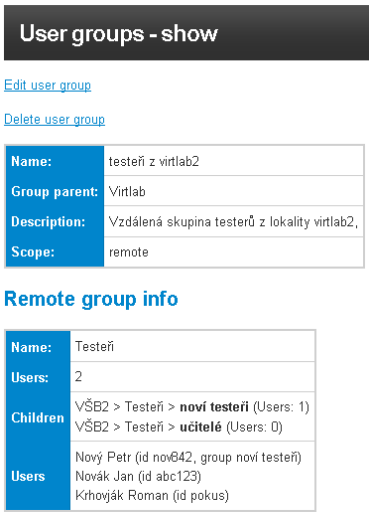
4 http://addons.nettephp.com/cs/datagrid

Obrázek 10: Původní popis lokální uživatelské skupiny



Obrázek 11: Nový popis lokální uživatelské skupiny

Popis vzdálené uživatelské skupiny



Obrázek 12: Popis vzdálené uživatelské skupiny

Vzdálená uživatelská skupina má základní popis stejný jako skupina lokální – název, seznam rodičů, popis a rozsah platnosti. Navíc následuje seznam informací o této skupině – Její název na vzdálené lokalitě, počet jejích uživatelů, seznam veřejných potomků (včetně počtu uživatelů v nich) a seznam všech uživatelů v této vzdálené skupině. Příklad popisu vzdálené skupiny je zobrazen na obrázku 12.

3.6.4 Úlohy

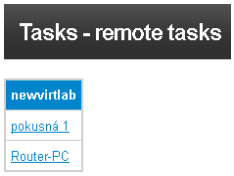
Stránka úlohy obsahovala seznam kategorií a tříd úloh s možností jejich výběru a také seznam úloh s možností jejich úpravy či mazání – tuto možnost měl jen správce úloh (obrázek 13). Nově (obrázek 14) jsou správci dané úlohy vybírání tvůrcem úlohy, to znamená, že členství ve skupině správci úloh nezaručuje možnost úlohu upravit či smazat. Také je přidán odkaz na zobrazení vzdálených úloh.

Obrázek 13: Původní seznam úloh

Obrázek 14: Nový seznam úloh

3.6.5 Seznam vzdálených úloh

Seznam vzdálených úloh obsahuje tabulky, kdy první řádek je název lokality a na dalších řádcích je seznam dostupných úloh (obrázek 15).



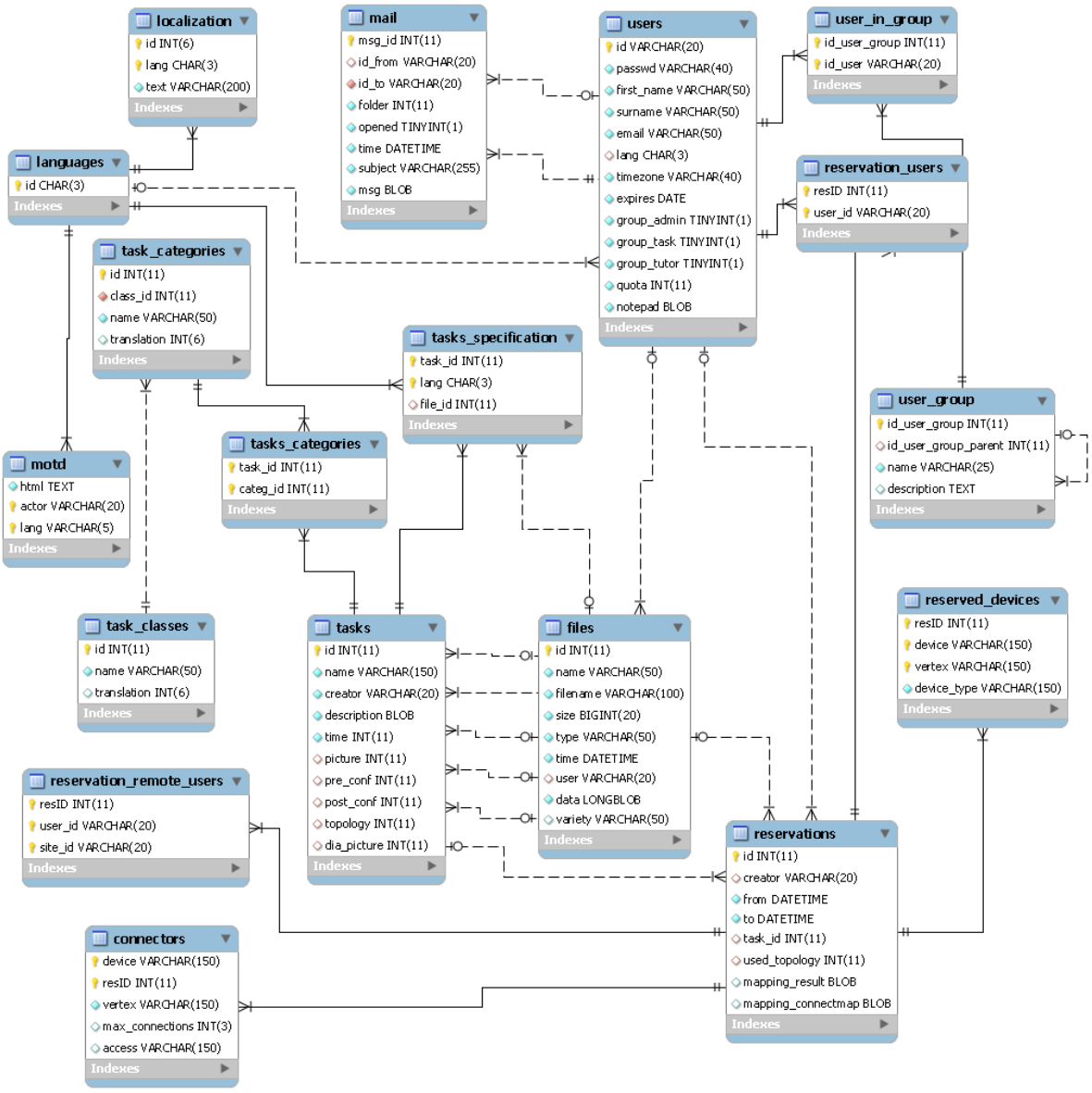
Obrázek 15: Seznam vzdálených uživatelských úloh

4 Návrh implementace

Návrh konkrétní implementace vychází ze specifikace požadavků a analýzy řešení. Popisuje seznam tříd a relací mezi nimi. Nejdříve ovšem popisuje změny v databázi – nové tabulky, úpravy tabulek, případně nové procedury, funkce, triggery.

4.1 Návrh databáze

Současná podoba databáze řídící aplikace Virtlabu má 18 tabulek. Tyto tabulky a jejich vztahy zobrazuje ERD na obrázku č. 16.



Obrázek 16: ERD před úpravou

4.1.1 Specifikace úpravy databáze

Databáze samozřejmě nezůstane nezměněna – k některým tabulkám přibudou sloupce, bude přidáno několik tabulek.[5]

4.1.1.1 Nové tabulky

task_admin

Správce úlohy: Přiřazuje k úloze seznam uživatelů, kteří mají oprávnění úlohu měnit či smazat (kromě tvůrce úlohy – sloupec *creator* v tabulce *tasks*). Noví správcové jsou vybíráni tvůrcem a současnými správci. Správcem konkrétní úlohy se může stát jen osoba s oprávněním dělat správce úloh (v tabulce *users* je hodnota *group_task* rovna jedné).

Jedná se tedy jen o spojovací tabulku mezi uživatelem (tabulka *users*) a úlohou (tabulka *tasks*)

shared_task

Sdílená úloha: Označuje, které skupiny uživatelů mají přístupová práva k dané úloze. Skupina, která má přístupová práva, si může úlohu prohlížet a rezervovat. Měnit přístupová práva skupin k úloze můžou tvůrce a správcové dané úlohy.

Je to tedy spojovací tabulka mezi úlohou (tabulka *tasks*) a uživatelskou skupinou (tabulka *user_group*).

notified_locality

Upozorňovaná lokalita: Pokud jsou ve skupině, která je někam naroubována, provedeny nějaké změny je specifikovaná lokalita o těchto změnách upozorněna. Za změny se považuje přidání či odebrání uživatele skupiny, přidání či odebrání podskupiny nebo smazání této skupiny. Záznam je do tabulky přidán, když je daná skupina nějakou lokalitou naroubována do jejího stromu skupin.

Tabulka má dva sloupce – jeden identifikuje skupinu a druhý lokalitu (propojeno na *soap_server.conf*), na kterou se zpráva odešle.

remote_group

Vzdálená skupina: Popisuje informace o vzdálené skupině, která byla naroubována do stromu skupin. Tato skupina je označena v tabulce *user_group* parametrem *scope* s hodnotou *remote*.

Jeden záznam v této tabulce identifikuje uživatelskou skupinu jako vzdálenou, tudíž obsahuje identifikátor této skupiny. Dále obsahuje název vzdálené lokality (napojeno na soubor *soap_server.conf*), ze které je skupina naroubována a identifikátor skupiny na vzdálené lokalitě.

temp_access

Dočasný přístup k souboru: Využívá se při přístupu k souboru ze vzdálené lokality. Soubory na serveru jsou uloženy pod číslem shodným s identifikátorem v tabulce *files*. Při snaze vzdálené lokality stáhnout soubor je uloženo toto číslo do tabulky *temp_access* a také se tam uloží vygenerované kontrolní číslo. Tyto údaje jsou odeslány vzdálené lokalitě a ta může v novém požadavku pomocí těchto údajů soubor stáhnout. Respektive uživatel si jej může stáhnout přímo ze

vzdálené lokality. Jedná se o bezpečnostní opatření, aby si nemohl kdokoliv soubory stahovat.

4.1.1.2 Úprava tabulek

user_group

Do této tabulky je přidán sloupec *scope* – rozsah platnosti. Jedná se o datový typ *set*, který může dosahovat přednastavených hodnot. Tyto hodnoty jsou:

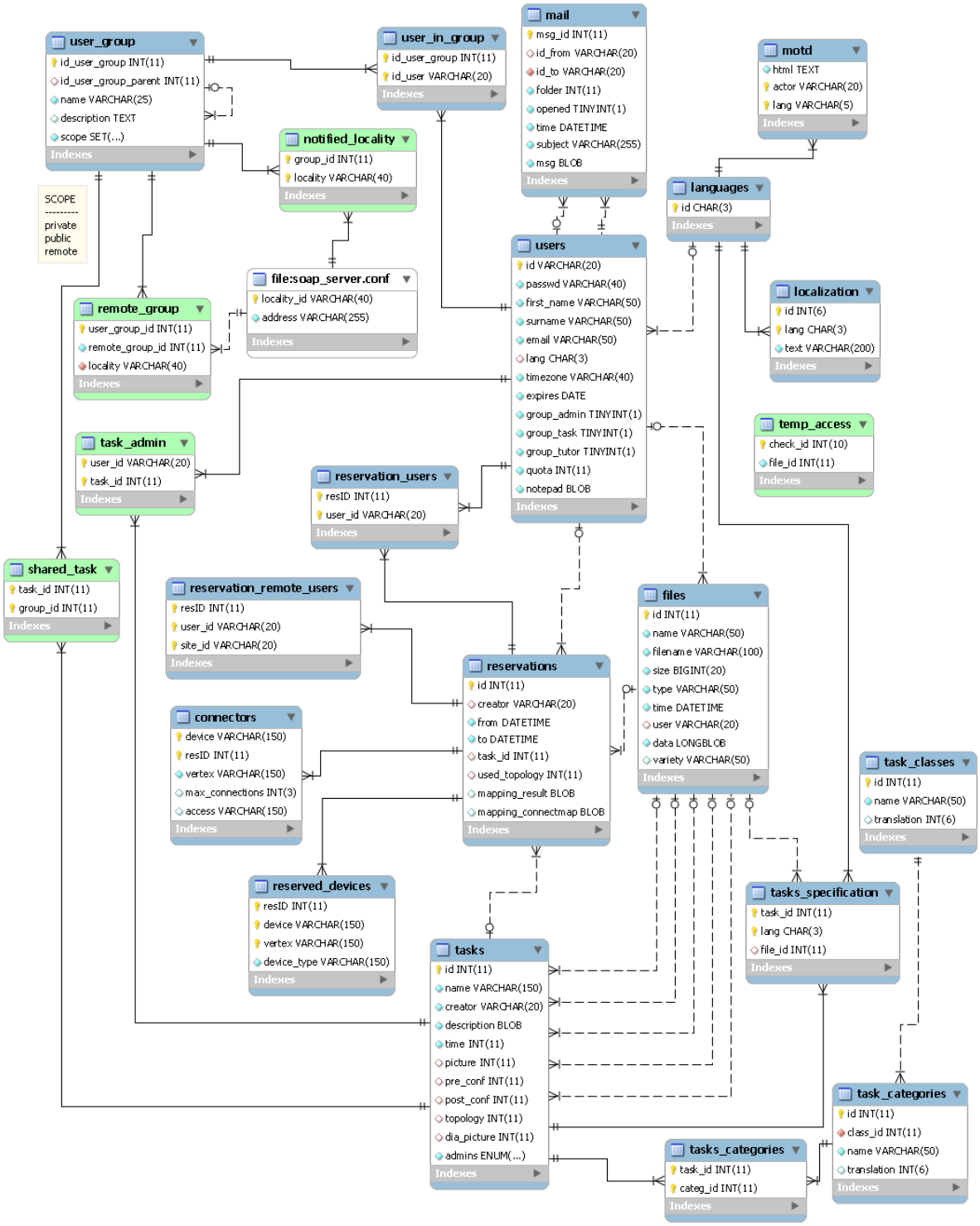
- *private* – privátní, jedná se o čistě lokální skupinu, která nemůže být roubována
- *remote* – vzdálená, jedná se o skupinu ze vzdálené lokality, nemůže být již nadále nikým roubována
- *public* – veřejná, ostatní lokality ji vidí a mohou si ji naroubovat do svého seznamu skupin

4.1.1.3 Konfigurační soubor pro SOAP servery

soap_server.conf : Tento soubor již existuje a je v něm uložen seznam vzdálených lokalit a jejich adres pro SOAP komunikaci (HTTPS adresy). Jedná se o soubor typu CSV, kdy oddělovačem sloupců jsou mezery. Ve výsledném ERD diagramu je zobrazen, aby bylo možné pochopit, jak budou navzájem lokality komunikovat.

4.1.2 Entity-relationship diagram

Výsledný ERD po úpravě databáze shrnuje obrázek č. 17. Modře jsou označeny původní tabulky, zeleně tabulky nově přidané a bíle konfigurační soubory.



Obrázek 17: ERD po úpravě

4.1.3 Procedury a funkce

Komponenta TreeView[5] přebírá jako parametr pro generování stromu pouze *DibiDataSource* objekt. Ten se typicky získává jedním SQL dotazem, proto bude vytvořena procedura, která vytvoří dočasnou tabulku s tímto požadovaným výsledkem.

4.2 Objektová analýza

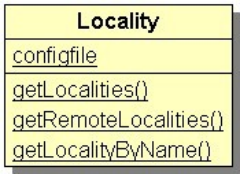
Zabývá se popisem nových i stávajících objektů, jejich nových funkcí a vztahů mezi objekty. Objekty jsou navrženy přímo pro Nette framework, využívající architekturu MVP.[3]

4.2.1 Model

Třídy modelu jsou mapovány na tabulky databáze a konfigurační soubory. Nachází se ve složce *app/models*. Všechny třídy modelu využívající ORM Ormion mají proměnnou *table*, která obsahuje název dotyčné tabulky.

Nové třídy tedy jsou:

Locality.php



Obrázek 18: Schéma třídy Locality

Obsahuje statické funkce pro práci s konfiguračním souborem pro SOAP komunikaci.

Funkce *getLocalities()* vrátí seznam lokalit jako asociativní pole, kdy klíč je adresa serveru a hodnota je jeho název.

Funkce *getRemoteLocalities()* vrátí obdobný seznam, ale vynechá lokální lokalitu. Ta je nakonfigurována v souboru *config.ini*.

Funkce *getLocalityByName($name)* přebírá jako parametr název, vrací adresu lokality.

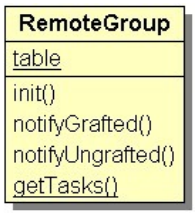
5 http://addons.nettephp.com/cs/treeview

NotifiedLocality.php

Obrázek 19: Schéma třídy NotifiedLocality

Je modelovou funkcí pro tabulku *notified_locality.*

RemoteGroup.php



Obrázek 20: Schéma třídy RemoteGroup

Je modelovou funkcí pro tabulku *remote_group*.

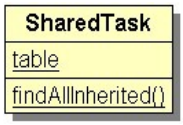
Funkce *getTasks($groupid, $locality)* přebírá identifikátor vzdálené skupiny a název lokality. Vyhledá a vrátí všechny úlohy, které jsou dané skupině přístupné.

V metodě *init()* je zajištěno, že po přidání záznamu se zavolá funkce *notifyGrafted()* a po odstranění záznamu se zavolá funkce *notifyUngrafted()*.

Ve funkci *notifyGrafted()* je poslána vzdálenému serveru notifikace o naroubování.

Funkce *notifiUngrafted()* informuje vzdálený server o odstranění naroubování.

SharedTask.php



Obrázek 21: Schéma třídy SharedTask

Je modelovou funkcí pro tabulku *shared_task*.

Funkce *findAllInherited($groupid)* vyhledá všechny úlohy přístupné dané skupině a to včetně těch co jsou přístupné rodičovským skupinám.

TaskAdmin.php

Obrázek 22: Schéma třídy TaskAdmin

Je modelovou funkcí pro tabulku *task_admin*.

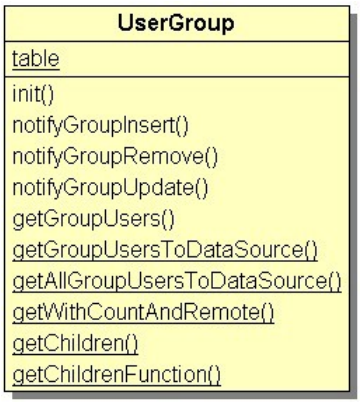
TempAccess.php

Obrázek 23: Schéma třídy TempAccess

Je modelovou funkcí pro tabulku *temp_access*.

Nové funkce v původních třídách:

UserGroup.php



Obrázek 24: Schéma třídy UserGroup

Nově přidaná metoda *init()* zajišťuje zavolání potřebných funkcí po přidání), smazání či úpravě záznamu v tabulce *user_group*.

Funkce *notifyGroupInsert()* je zavolána po vložení záznamu. Ověří se, zda jsou rodiče dané skupiny někam naroubováni. Pokud ano, pošle dotyčným lokalitám zprávu o přidání skupiny do naroubované části stromu.

Obdobně funguje funkce *notifyGroupRemove()*, která se zavolá po odstranění skupiny a pošle

případnou zprávu o odstranění skupiny z naroubované části stromu.

Funkce *notifyGroupUpdate()* informuje stejným typem zprávy jako *notifyGroupInsert()* vzdálenou lokalitu o přidání skupiny do naroubované části a to v případě, že se rozsah platnosti změní z privátního na veřejný.

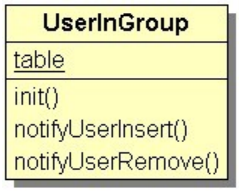
Jsou vytvořeny také nové funkce pro získávání specifických dat: *getGroupUsersToDataSource($groupid)* vrátí uživatele skupiny v objektu typu *DibiDataSource*. Tento typ je totiž vyžadován komponentou DataGrid.

Funkce *getAllGroupUsersToDataSource($groupid)* pak vrátí všechny uživatele v dané skupině a jejích potomcích (také v *DibiDataSource*).

Funkce *getWithCountAndRemote()* vrátí seznam všech skupin včetně informací o počtu uživatelů ve skupině a informací o vzdálené skupině (pokud jí skupina je).

Funkce *getChildren($groupid, $scope = '')* vrací seznam všech potomků dané skupiny. Pokud je zadán i parametr *scope*, vrátí jen potomky s tímto rozsahem platnosti.

UserInGroup.php



Obrázek 25: Schéma třídy UserInGroup

Funkce *init()* zajistí zavolání potřebných funkcí po vložení (zavolá se *notifyUserInsert()*) a po odstranění (*notifyUserRemove()*) záznamu.

Ve funkci *notifyUserInsert()* se ověří, zda je skupina naroubována nějakou lokalitou a pokud ano, pošle se jí informace o přidání uživatele do této skupiny.

Obdobně funguje funkce *notifyUserRemove()*, která upozorní lokalitu na odstranění uživatele ze skupiny.

Task.php

Obrázek 26: Schéma třídy Task

Přidána byla funkce *isUserTaskAdmin($useridentity)*, která vrátí informaci o tom, zda má přihlášený uživatel práva správce dané úlohy.

4.2.2 Moduly

Nový je modul pro SOAP komunikaci v podadresáři *soap*. Ten obsahuje soubory *MySoapClient.php*, *MySoapServer.php*, *FileDownload* a také soubor *.htaccess*.

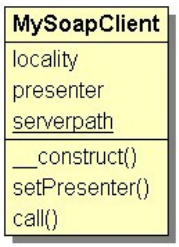
Také byly upraveny některé současné soubory v adresáři s moduly.

4.2.2.1 SOAP modul

.htaccess

Protože práva přístupu do souborů aplikace jsou ve výchozím nastavení z bezpečnostních důvodů zakázána, je třeba je pro tento adresář upravit. Všechny soubory v této složce jsou tedy přístupné. Je to z toho důvodu, že soubor serveru a soubor pro stažení souboru jsou volány vzdáleným serverem.

MySoapClient.php



Obrázek 27: Schéma třídy MySoapClient

MySoapClient je třída, která je programem volána ve chvíli, kdy je třeba komunikovat se vzdáleným serverem. Při vytváření instance tohoto objektu je zadána adresa lokality, na které se bude volat

SOAP server.

Metoda *setPresenter($presenter)* určí, která třída bude na serveru požadavek obsluhovat. Tyto speciální presentery jsou uloženy v aplikační složce v adresáři *presenters/soap*.

Konečně funkce *call($function, $parameters)* zavolá zadanou funkci na vzdáleném serveru a předá jí parametry. Vzdálená funkce vrací serializovaný výstup, ten je deserializován a vrácen.

MySoapServer.php

Obrázek 28: Schéma třídy MySoapServer

Je kód, který je volán vzdálenou lokalitou. Abychom vyhověli objektovému návrhu, zavolaný kód pouze vytvoří instanci třídy MySoapServer v tomto souboru.

Třída *MySoapServer* po své instanciaci inicializuje Nette framework, aby bylo možné využívat všechny jeho funkce a komponenty. Poté vytvoří SoapServer a zaregistuje do něj jedinou funkci – *call*.

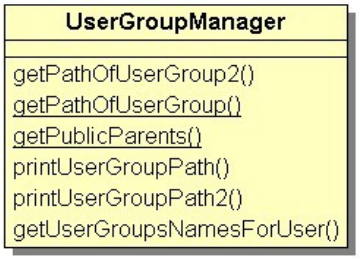
Funkce *call($presenter, $function, $parameters)* je volána klientem vzdálené lokality. V podstatě jen přeposílá požadavek zadanému presenteru a jeho funkci. Vrací pak serializovaný výsledek volané funkce.

FileDownload.php

Kód, který je volán vzdálenou lokalitou v momentě, kdy chce uživatel stáhnout soubor. Metodou GET je předán identifikátor souboru a kontrolní číslo. Skript ověří zda má uživatel přístup k souboru (zda je záznam v tabulce *temp_access* s předaným identifikátorem a kontrolním číslem). Pokud uživatel přístup má, je mu soubor odeslán a dočasný přístup odstraněn (smazán dotyčný záznam z tabulky *temp_access*).

4.2.2.2 Úprava stávajících modulů

UserGroupManager.php



Obrázek 29: Schéma třídy UserGroupManager

Funkce *getPathOfUserGroup($id_user_group)* byla přejmenována na *getPathOfUserGroup2($id_user_group)*, pro případ, že by ji bylo nutné někdy použít. Nově vzniklá funkce *getPathOfUserGroup($id_user_group)* vrací seznam skupin. Jedná se o cestu od dané skupiny ke kořeni stromu.

Funkce *getPublicParents($groupid)* funguje obdobně, s tím rozdílem, že vrací jen rodiče, kteří mají rozsah platnosti public, jsou tedy veřejné.

4.2.3 Presenter

V podadresáři *soap* se nachází třídy, které nejsou přímo presentery Nette, ale plní obdobnou funkci. Obsahují statické funkce, které jsou volány SoapClientem prostřednictvím SoapServeru. Vrací data požadovaná klientem. Dále byly upraveny presentery související s uživatelskými skupinami, úlohami a jejich rezervací.

4.2.3.1 Soap presenter

SoapFilePresenter.php

Obrázek 30: Schéma třídy SoapFilePresenter

Zpracovává požadavky související se soubory.

Funkce *getFileLink($fileId)* vrátí odkaz na zadaný soubor včetně nově vytvořeného kontrolního čísla.

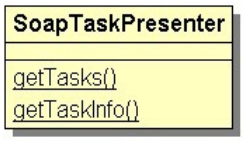
SoapNotificationPresenter.php

Obrázek 31: Schéma třídy SoapNotificationPresenter

Obsahuje sadu funkcí sloužících ke zpracování přijatých notifikací. K odeslání mailu administrátorům slouží *private* funkce *sendToAdmins($subject, $body)*.

Funkce nebudu podrobně rozebírat, více informací o principu notifikací je v kapitole 3.5.2.

SoapTaskPresenter.php



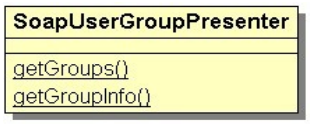
Obrázek 32: Schéma třídy SoapTaskPresenter

Třída obsahuje funkce poskytující informace o úlohách.

Funkce *getTasks($groups, $locality)* vrací seznam úloh, které jsou přístupné předaným vzdáleným skupinám ze specifikované vzdálené lokality.

Zavoláním funkce *getTaskInfo($id)* získáme informace o dané úloze – její popis a popis k ní přiložených souborů.

SoapUserGroupPresenter.php



Obrázek 33: Schéma třídy SoapUserGroupPresenter

Funkce této třídy poskytují informace o skupinách v lokalitě.

Funkce *getGroups()* vrátí seznam všech veřejných skupin.

Funkce *getGroupInfo($groupid)* vrátí informace o zadané skupině – kromě jejího popisu, také seznam potomků a seznam všech uživatelů ve skupině i jejích potomcích.

4.2.3.2 Úprava současných tříd

FilePresenter.php

Fukce *renderShowRemoteFile($id, $locality)* zobrazí soubor s daným id na určené lokalitě. Pomocí SOAPu získá odkaz (URL) na soubor a na ten poté přesměruje.

TaskPresenter.php

Funkce zobrazující template *showRemoteTasks* s názvem *renderShowRemoteTasks()* zjistí, ve kterých skupinách se uživatel nachází Poté projde lokality a získá z nich seznam úloh přístupných těmto skupinám. Ty jsou pak zobrazeny v template.

Další novou renderovací funkcí je *renderShowRemoteTask($taskid, $locality)*, která z určené lokality zjistí informace o dané úloze a zobrazí je.

Dále jsou upraveny funkce pro generování (*createComponentTaskForm()*) a zpracování (*taskFormSubmitted(AppForm $form)*) formuláře skupiny – je přidána možnost vybírat správce skupiny a také možnost vybrat skupiny, které mají k úloze přístup.

UserGroupPresenter.php

Komponenta TreeView byla přeprogramována, proto jsou i funkce s ní související uzpůsobeny této změně. Tato třída byla celá do velké míry upravena tak, aby šlo manipulovat se vzdálenými uživatelskými skupinami či rozsahem platnosti skupiny. Zaměřím se jen na nově vzniklé funkce.

V zobrazení detailu skupiny je možné uživatele ze seznamu odebírat. Série metod *renderRemoveUserFromGroup($group, $id)*, *createComponentRemoveUserFromGroupForm()* a *removeUserFormSubmitted(AppForm $form)* slouží právě k tomuto účelu.

Pro efektivnější zobrazení seznamu uživatelů vybrané skupiny vytváří funkce

createComponentUsersInGroupGrid($name) seznam v komponentě DataGrid. Ten umožní řazení, editaci uživatelů či odebrání uživatele ze skupiny.

Při zaškrtnutí či odškrtnutí tlačítka *I uživatelé z potomků* je odeslán AJAXový požadavek - zavolá se funkce *handleChildrenGroupsShow()*.

Metoda *createComponentTreeForm()* vytváří formulář, který umožňuje u označených skupin měnit rozsah platnosti či je mazat.

Pro zpracování formuláře v *TreeForm* se používají metody *setPublic(SubmitButton $button)*, *setPrivate(SubmitButton $button)* a *deleteGroups(SubmitButton $button)* v závislosti na tom, jakým tlačítkem byl formulář odeslán.

4.2.4 Templates

Templates jsou šablony příslušející k jednotlivým presenterům. Jedná se většinou o HTML kód obohacený o cykly a podmínky, sloužící k požadované prezentaci dat uživateli.

UserGroup

V tomto adresáři se šablonami nově přibyla šablona *removeUserFromGroup*. Ta zobrazí potvrzující formulář, zda si přeje uživatel odstranit vybraného uživatele ze skupiny.

Ostatní šablony byly upraveny tak, aby odpovídali navrhnuté funkčnosti a grafickému návrhu. Přibyl také soubor s javascriptovými funkcemi, které umožňují načtení a zobrazení seznamu vzdálených lokalit a vzdálených skupin (AJAXový požadavek).

Task

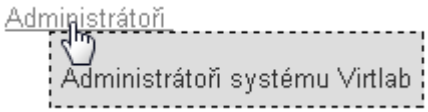
Do šablon pro práci s úlohou přibyly funkce *showRemoteTasks* a *showRemoteTask*. První zmíněná zobrazí všechny vzdálené úlohy dostupné uživateli. Druhá pak zobrazí detaily o vybrané vzdálené úloze a umožní její rezervaci.

4.2.5 Ostatní

libs/TreeView

Původní komponenta byla rozšířena tak, aby splňovala nové požadavky na zobrazení stromu uživatelských skupin.

js/tooltip.js



Obrázek 34: Ukázka JQuery Tooltip

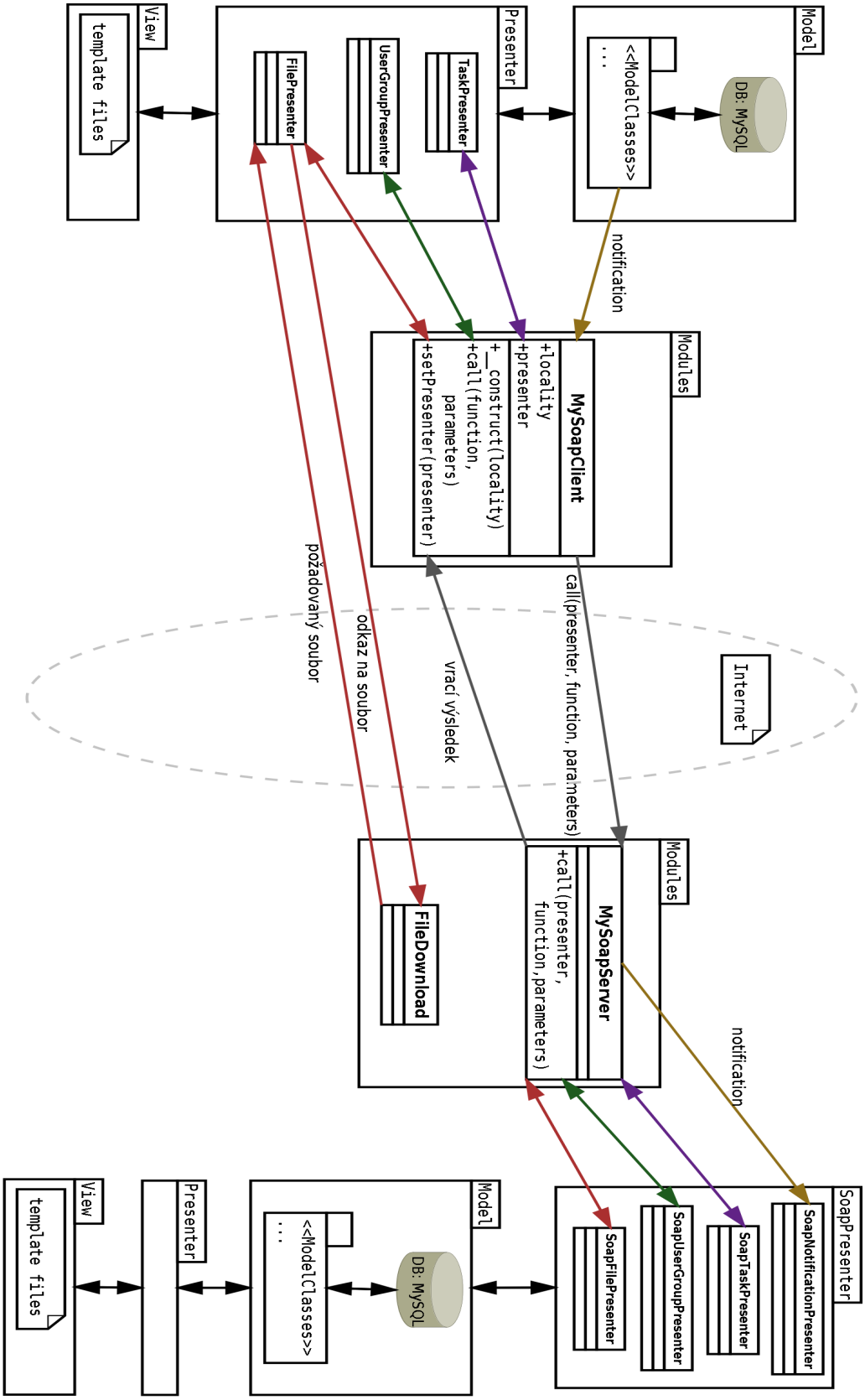
Jedná se o JQuery knihovnu, která u zadaného prvku po najetí myší zobrazí specifikovaný text. Text je uložen v atributu title daného HTML tagu. Knihovnu jsem upravil tak, aby dlouhé testy odřádkovala.

config.ini

Tento konfigurační soubor se nachází v aplikačním adresáři. Byla do něj přidána proměnná *variable.locality_soap*, která udává absolutní adresu této lokality. Její podoba je stejná jako v konfiguračním souboru SOAP serverů (*soap-servers.conf*).

4.2.6 Diagram objektů s ohledem na SOAP komunikaci

Obrázek 35 přehledně zobrazuje některé objekty úpravy a znázorňuje, jakým způsobem spolu lokality a jejich objekty komunikují. Nejedná se o kompletní diagram objektů, pouze o schéma návrhu se znázorněním principu interakce mezi lokalitami.



Obrázek 35: Diagram objektů s ohledem na SOAP komunikaci

5 Implementace řešení

5.1 Databáze

V databázi se nachází tabulky s formátem úložiště dat InnoDB. Jeho hlavní výhodu (pro použití u Virtlabu) oproti jiným MySQL formátům vidím v možnosti specifikování cizích klíčů. Rozdílů je oproti největší konkurenci MyISAM více (např. transakce). Nebudu je rozepisovat, lze je najít v dokumentaci[6]. Já jsem tedy využil také úložiště InnoDB, určil cizí klíče a specifikoval integritní pravidla. Pro kódování řetězců znaků se používá formát UTF-8.

Skript pro úpravu databáze je v příloze B.

Za zmínku stojí nutnost přenastavit proměnnou databáze *max_sp_recursion_depth* na hodnotu 50. Tato proměnná udává, jaká je maximální hloubka, do které půjdou rekurzivní funkce v procedurách. Tato rekurze je využita v proceduře, která získá strom uživatelských skupin. Nepředpokládám, že by strom uživatelských skupin měl větší hloubku než 50. Kdyby toto nastalo, lze ji jednoduše přenastavit.

5.2 Popis implementace SOAP

Pro SOAP se využívá extenze **php_soap**, kterou je nutné mít na serveru aktivovanou. Aktivace se ve Windows provede odkomentováním řádky *;extension=php_soap.dll* (čili odstraněním středníku) v souboru *php.ini*. Extenze má různá nastavení, ta ovšem není třeba měnit, postačí výchozí. Na operačním systému typu Linux se instalace liší dle distribuce, často je ovšem nutná rekompilace PHP serveru.[2,6]

Stěžejní soubory pro SOAP jsou *MySoapClient.php* a *MySoapServer.php*. Jejich zdrojový kód včetně komentářů je v příloze C. *MySoapClient* je volán z presenterů aplikace. *MySoapServer* je pak volán *MySoapClient*em. Klient serveru říká, v kterém presenteru, kterou funkci zavolat. Poté, co ji server zavolá, vrací klientovi požadovaný výsledek.

Ve zdrojovém kódu může být zarážející, proč je vracená hodnota kódována třemi různými funkcemi (*json_encode*, *base64_encode* a *serialize*). Metoda *serialize* je nejvhodnější pro serializaci dat pro přenos přes SOAP. Má ovšem problém s UTF-8 a také pokud serializuje nějaký objekt, tak „zapomene“, jakého byl typu a označí jej jako *std_class*. Funkce *json_encode* naopak umí pracovat pouze s kódováním UTF-8, ale při pokusu o přenos přes SOAP selže. Zakódování pomocí *base64* ošetří přenos dat, která nejsou „8-bit clean“. Zkoušel jsem kteroukoliv z funkcí samostatně, či jakoukoliv kombinaci těchto dvou kódování, ale vždy se naskytla situace, kdy přenos selhal nebo ke klientovi došel neúplný výsledek. Tyto problémy budou pravděpodobně způsobeny tím, že PHP 5 nepodporuje nativně kódování UTF-8 (mělo by se zlepšit v PHP 6).

6 http://dev.mysql.com/doc/refman/5.1/en/storage-engines.html

5.3 Další informace o implementaci

Ve zdrojovém kódu jsou všechny objekty a jejich metody okomentované. Navíc třídy, funkce i proměnné byly pojmenovány tak, aby jejich názvy dávaly co největší smysl. U složitějších či delších kódu jsou komentáře i uvnitř funkcí. Pro programátora by tedy neměl být problém se v implementaci vyznat.

6 Testování a nasazení

6.1 Instalace modulu

6.1.1 Instalace do současného řešení

V současnosti běží nová verze řídící aplikace Virtlabu pouze na testovacím virtuálním serveru *praha.dvirtlab.net/new/*. Její autor ji tam v tomto roce nasadil a stihla být otestována. Mým úkolem je do této distribuce přidat mou úpravu.

Instalace by měla být poměrně jednoduchá – spustit SQL skript pro tvorbu a úpravu tabulek, nahrát nové a změněné soubory, upravit konfigurační soubory. Skript pro úpravu tabulek se nachází v adresáři database, jedná se o soubor *sql2.sql*. Zdrojové soubory lze přenahrát všechny (kromě konfiguračních), neboť jsem je aktualizoval dle poslední verze řídícího serveru. Informace o nastavení konfiguračních souborů jsou v kapitole 4.2.5 - pod nadpisem *config.ini* a v kapitole 4.1.1.3.

6.1.2 Nová instalace distribuce

Nová instalace proběhne obdobně jako instalace do současného řešení – je třeba vytvořit databázi (v tomto případě se spouští nejdříve skript *sql1.sql* a poté *sql2.sql*), nahrát soubory a nastavit konfigurační soubory. Více informací je také v diplomové práci tvůrce nové řídící aplikace.[4]

6.2 Lokální testování

V této kapitole stručně popíšu, na jakých hardwarových a softwarových prostředcích jsem řídící aplikaci upravoval. Také přidám různé tipy, jak takovouto aplikaci efektivně vyvíjet.

O hardware jen stručně – jedná se o notebook Dell XPS M1330, procesor T7250@2GHz – dvoujádrový, 2 GB RAM, grafická karta GeForce 8400M GS. Vzhledem k tomu, že při vývoji byly obvykle zapnuty i nějaké ladící a logovací nástroje, nebyla odezva počítače ani webového prohlížeče zrovna nejlepší. Problém nebyl způsoben nedostatkem paměti RAM, vždy zbývala nějaká volná.

Aplikace byla vyvíjena pod operačním systémem Windows XP, později pod Windows 7. Pro Windows existuje propracovaná aplikace pro vývoj v PHP s názvem WampServer[7]. Jedná se o kompilace serveru Apache, databáze MySQL a PHP serveru. Také umožňuje některá nastavení těchto serverů upravit v grafickém uživatelském rozhraní. Já jsem používal verzi Wamp 2.0i, která se dodává s Apache 2.2.11, MySQL 5.1.36 a PHP 5.3.0.

Jako vývojové prostředí jsem zvolil nekomerční řešení Eclipse PDT[8] ve verzi 2.1. Pro debugování v Eclipse jsem využil rozšíření XDebug[9]. Myslím, že se jedná o jedno z nejlepších nekomerčních

7 http://www.wampserver.com/en/
8 http://www.eclipse.org/pdt/
9 http://xdebug.org/

řešení, je dost robustní a nabízí programátorům mnoho možností, jak si usnadnit vývoj. Problém jsem měl jen s ne příliš velkou rychlostí aplikace a občas i se stabilitou.

Pro ladění javascriptu a AJAXu je vhodným nástrojem plugin pro prohlížeč FireFox s názvem FireBug[10]. Dle mého názoru je to skvělý nástroj, který nejen že umožňuje debug javascriptu, ale i ladění CSS stylů a mnoho dalšího.

Také jsem využil logování SQL dotazů do souborů – a to jak všech tak i takzvaných slow (těch co trvají příliš dlouho).Nastavit logování lze v souboru my.ini. Logování všech souborů zapneme pomocí direktivy *log*, například *log=c:/mysql_all.log*. Slow-log zapneme pomocí *log-slow-queries*, například takto: *log-slow-queries=c:/mysql_long.log*.

Abych měl přehled co a kdy jsem upravil, využil jsem subversion, konkrétně nástroj TortoiseSVN[11]. Hodilo se to zejména když jsem měl na localhostu nainstalované dvě distribuce Virtlabu a bylo je třeba synchronizovat.

Při instalaci dvou distribucí Virtlabu na jeden stroj jsem narazil na jeden problém a tím byla databáze. Procedury jsou v MySQL ukládány pro celý server nikoliv pro konkrétní databázi na něm. Problém jsem vyřešil tak, že jsem přiinstaloval další MySQL server, který jsem rozjel na jiném portu.

6.3 Testování na produkčním serveru

Na produkční server se úprava dostala krátce před odevzdáním diplomové práce. Aplikace bude testována, dokud nebudou vylazeny všechny chyby. Poté může být nasazena v ostrém provozu.

10 http://getfirebug.com/
11 http://tortoisesvn.tigris.org/

7 Závěr

Úprava řídící aplikace prospěje jak uživatelům, tak administrátorům. Uživatelům poskytne možnost rezervovat si úlohy i ze vzdálených lokalit. Pro administrátory je přínos ještě větší. Mají možnost upravovat přístupová práva uživatelů k úlohám. Také mohou povolit přístup k úlohám vzdáleným uživatelům a to pomocí uživatelských skupin. Tvůrce úlohy může určit, kteří správci skupin mají možnost úlohu upravovat a mazat.

Mým úkolem bylo navázat na novou podobu řídící aplikace, implementovanou v průběhu tohoto roku. Musel jsem tedy úzce spolupracovat s autorem.[4] Měl jsem dostatek času na analýzu řešení a návrh aplikace. Ty jsem řešil s pomocí členů vývojového týmu virtlabu na pravidelných konzultacích. Méně času jsem měl na implementaci a testování, neboť autor nové verze řídící aplikace odevzdává svou práci ve stejnou dobu jako já, neměl jsem tedy od počátku zdrojové kódy k dispozici. I přesto se mi podařilo včas program implementovat a lokálně otestovat. Nyní je nasazen na produkčním serveru a začíná jeho testování v prostředí internetu.

S nasazením nové podoby řídícího serveru se předběžně počítá do zimního semestru roku 2010/2011. Letos na něm i na ostatních serverech Virtlabu pracovalo několik diplomantů, bude třeba tedy jejich úpravy sloučit, důkladně otestovat a nasadit. K tomu poslouží letní prázdniny, kdy studenti nemají potřebu Virtlab využívat.

Nová verze řídící aplikace byla implementována v programovacím jazyce PHP s využitím frameworku Nette. Architektura aplikace je třívrstvá – využívá moderní architektonický model MVP. Jako úložiště dat byla zvolena databáze MySQL 5. Pro přístup k této databází se využilo mini databázové vrstvy Dibi, nad kterou byl použit ORM modul Ormion. Mé znalosti byly před započetím práce omezeny na jazyk PHP a databázi MySQL, nabyl jsem tedy mnoho nových znalostí. Myslím si, že mi tato práce byla velkým přínosem a že nově získané znalosti využiji i ve svém budoucím povolání.

Literatura

[1] *VirtlabWiki* [online]. Vývojový tým Virtlabu. Dostupné z WWW: <http://www.cs.vsb.cz/vl-wiki>

[2] *PHP Manual: SOAP* [online]. The PHP Group, c2010. Dostupné z WWW: <http://www.php.net/manual/en/book.soap.php>

[3] *Nette Framework: Dokumentace* [online]. Nette Foundation, c2010. Dostupné z WWW: <http://nettephp.com/cs/dokumentace>

[4] GRAMES, Martin.*Reimplementace řídící aplikace systému Virtlab s použitím moderních webových technologií*. Ostrava, 2010. Diplomová práce na fakultě Elektrotechniky a informatiky Vysoké školy báňské - Technické univerzity Ostrava na Katedře informatiky. Vedoucí diplomové práce Petr Grygárek.

[5] KOFLER, Michael. *Mistrovství v MySQL 5*. Vydání první. Computer Press, a.s., 2007. 808 s. ISBN: 978-80-251-1502-2.

[6] CHOW, Shu-Wai. *Programujeme Mashup aplikace pro Web 2.0 v PHP*. Vydání první. Computer Press, a.s., 2008. 280 s. ISBN: 978-80-251-2057-6.

Seznam obrázků

Seznam příloh

Příloha A – Obsah CD

Obsah přiloženého CD je popsán následovně: ***Název složky*** – popis obsahu.

virtlab – zdrojové kódy aplikace.

diploma – text diplomové práce v různých formátech.

Příloha B – Úprava databáze

```
-- new tables and alters

SET NAMES utf8;
SET foreign_key_checks = 0;
SET sql_mode = 'NO_AUTO_VALUE_ON_ZERO';

ALTER TABLE `user_group` ADD `scope` SET( 'private', 'public', 'remote' ) NOT NULL DEFAULT
'private';

CREATE TABLE `task_admin` (
`user_id` VARCHAR( 20 ) NOT NULL ,
`task_id` INT NOT NULL ,
PRIMARY KEY ( `user_id` , `task_id` ),
FOREIGN KEY (`user_id`) REFERENCES `users` (`id`),
FOREIGN KEY (`task_id`) REFERENCES `tasks` (`id`)
) ENGINE = InnoDB ;

CREATE TABLE `notified_locality` (
  `group_id` INT NOT NULL ,
  `locality` VARCHAR(40) NOT NULL ,
  PRIMARY KEY (`group_id`, `locality`),
  FOREIGN KEY (`group_id`) REFERENCES `user_group` (`id_user_group`)
) ENGINE = InnoDB;

CREATE TABLE `shared_task` (
  `task_id` INT NOT NULL ,
  `group_id` INT NOT NULL ,
  PRIMARY KEY (`task_id`, `group_id`),
  FOREIGN KEY (`task_id`) REFERENCES `tasks` (`id`),
  FOREIGN KEY (`group_id`) REFERENCES `user_group` (`id_user_group`)
) ENGINE = InnoDB;

CREATE TABLE `remote_group` (
  `user_group_id` INT NOT NULL ,
  `locality` VARCHAR(40) NOT NULL ,
  `remote_group_id` INT NOT NULL ,
  PRIMARY KEY (`user_group_id`),
  FOREIGN KEY (`user_group_id`) REFERENCES `user_group` (`id_user_group`)
) ENGINE = InnoDB;

-- procedures

set @@max_sp_recursion_depth=32;

DELIMITER $$

CREATE PROCEDURE `get_sub_groups`(IN node_id INT)
BEGIN
    DECLARE hotovo INT DEFAULT 0;
```

```
   DECLARE podkat CURSOR FOR
      SELECT id_user_group FROM user_group WHERE id_user_group_parent=node_id;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET hotovo=1;
   OPEN podkat;
   podkatcyklus: LOOP
      FETCH podkat INTO node_id;
      IF hotovo=1 THEN LEAVE podkatcyklus; END IF;
      INSERT INTO __users_in_cat
         SELECT U.id, U.first_name, U.surname, U.email, U.lang, U.timezone, U.expires,
U.group_admin, U.group_task, U.group_tutor, U.quota, G.id_user_group
            FROM users U, user_in_group G
            WHERE G.id_user_group = node_id AND U.id=G.id_user;
      CALL get_sub_groups(node_id);
   END LOOP podkatcyklus;
   CLOSE podkat;
END$$

CREATE PROCEDURE `return_cat_users`(IN startid INT)
BEGIN
   DROP TABLE IF EXISTS __users_in_cat;
   CREATE TEMPORARY TABLE __users_in_cat
      (id VARCHAR(20), first_name VARCHAR(50), surname VARCHAR(50), email
VARCHAR(50), lang CHAR(3),
      timezone VARCHAR(40), expires DATE, group_admin TINYINT(1),group_task
TINYINT(1),group_tutor TINYINT(1),
      quota INT, id_user_group INT)
      ENGINE = HEAP;

   INSERT INTO __users_in_cat
      SELECT U.id, U.first_name, U.surname, U.email, U.lang, U.timezone, U.expires,
U.group_admin, U.group_task, U.group_tutor, U.quota, G.id_user_group
         FROM users U, user_in_group G
         WHERE G.id_user_group = startid AND U.id=G.id_user;
   CALL get_sub_groups(startid);
END$$

DELIMITER ;

ALTER TABLE `tasks` ADD `admins` ENUM( 'selected', 'all' ) NOT NULL DEFAULT 'all';


ALTER TABLE `task_admin` DROP FOREIGN KEY `task_admin_ibfk_1` ;
ALTER TABLE `task_admin` ADD FOREIGN KEY ( `user_id` ) REFERENCES `users` (`id`) ON
DELETE NO ACTION ON UPDATE NO ACTION ;

ALTER TABLE `task_admin` DROP FOREIGN KEY `task_admin_ibfk_2` ;
ALTER TABLE `task_admin` ADD FOREIGN KEY (`task_id`) REFERENCES `tasks`(`id`) ON
DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `shared_task` DROP FOREIGN KEY `shared_task_ibfk_1` ;
ALTER TABLE `shared_task` ADD FOREIGN KEY ( `task_id` ) REFERENCES `tasks` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE ;
```

```
ALTER TABLE `shared_task` DROP FOREIGN KEY `shared_task_ibfk_2` ;
ALTER TABLE `shared_task` ADD FOREIGN KEY ( `group_id` ) REFERENCES `user_group`
(`id_user_group`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `remote_group` DROP FOREIGN KEY `remote_group_ibfk_1` ;
ALTER TABLE `remote_group` ADD FOREIGN KEY ( `user_group_id` ) REFERENCES
`user_group` (`id_user_group`) ON DELETE CASCADE ON UPDATE CASCADE ;

CREATE TABLE `temp_access` (
`check_id` INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`file_id` INT NOT NULL
) ENGINE = InnoDB COMMENT = 'temporary access to files for users from remote localities';
```

Příloha C - Zdrojové kódy vybraných tříd

MySoapServer.php

```php
<?php
/*
 * initializes soap server
 * called by remote MySoapClient class
 *
 * @author Roman Krhovjak
 */

new MySoapServer;

/*
 * Calls specified function of specified soap presenter
 *
 * @param string $presenter
 * @param string $function
 * @param array $parameters
 * @return mixed
 */
function call($presenter, $function, $parameters) {
	//serializuji - aby se zachovala vsechna data
	// json_encode - aby se zakodovali utf-8 znaky
	$pars = array();
	if( !is_array($parameters) ) $pars['par1'] = $parameters;
	else $pars=$parameters;
	$res = call_user_func_array(array($presenter, $function), $pars);
	return json_encode(base64_encode(serialize($res)));
}

/*
 * Soap server class
 */
class MySoapServer {
	/*
	 * Initializes nette, dibi, ormion and its functionality
	 * Initializes soap server and adds function 'call' to server
	 */
	public function __construct() {
		define('WWW_DIR', dirname(__FILE__) . '/../../..');
		define('APP_DIR', WWW_DIR . '/app');
		define('LIBS_DIR', WWW_DIR . '/libs');

		require LIBS_DIR . '/Nette/loader.php';

		$loader  = new RobotLoader();

		$loader->addDirectory(APP_DIR . '/models');
		$loader->addDirectory(APP_DIR . '/modules');
		$loader->addDirectory(APP_DIR . '/presenters');
		$loader->addDirectory(LIBS_DIR);
		$loader->register();
```

```
            Environment::loadConfig();
            dibi::connect(Environment::getConfig("database"));

            $url = "http://" . $_SERVER['HTTP_HOST'] .
'/app/modules/MySoapServer.php';
            $server = new SoapServer(null, array('uri' => $url));
            $server->addFunction("call");
            $server->handle();
      }
}
?>
```

MySoapClient.php

```
<?php
/*
 * Soap client class
 *
 * @author Roman Krhovjak
 */
class MySoapClient {
      private $locality;
      private $presenter;
      public static $serverpath = "app/modules/soap/MySoapServer.php";

      /*
       * Creates instace of this class with specified locality
       *
       * @param string $locality
       */
      public function __construct($locality) {
            $this->locality = $locality;
      }

      /*
       * Defines soap presenter which will be used in remote locality
       *
       * @param string $presenter
       */
      public function setPresenter($presenter) {
            $this->presenter = $presenter;
      }

      /*
       * Calls specified function in remote locality and returns result
       *
       * @param $function
       * @param $parameters
       * @return mixed
       */
      public function call($function, $parameters) {
            $location = $this->locality . self::$serverpath;
            // pripojit k lokalite
            $client = new SoapClient(null, array('location' => $location,
'uri' => $this->locality));

            // zavolat soap funkci call na serveru, deserializovat a vratit
vysledek
```

```
            try {
                  return unserialize(base64_decode(json_decode($client-
>call($this->presenter, $function, $parameters))));
            } catch (SoapFault $e) {
                  error_log($e);
            }

            return false;
      }
}

?>
```