



BAKALÁŘSKÁ PRÁCE

**Automatické administrační
rozhraní nad relační databází**

**Automatic administrative interfaces
for relational databases**

Martin Vondráček

Unicorn College © 2011

Unicorn College, V Kapslovně 2767/2, Praha 3, 130 00

Název práce v ČJ: Automatické administrační
rozhraní nad relační
databází

Název práce v AJ: Automatic administrative
interfaces for relational
databases

Autor: Martin Vondráček

Akademický rok: 2010/2011

Kontakt: E-mail: vondr81@gmail.com
Tel.: (+420) 736 533 498

1. ZADÁNÍ

**2010.10.07 - 18:07 - Vondráček Martin - Zadání BP - Automatická administrační
rozhraní nad relační databází (ZBP/20101007_011)**

Artefakt	Požadavky na schválení zadání BP
Zadavatel	Student (Vondráček Martin)

ZADÁNÍ ZÁVĚREČNÉ BAKALÁŘSKÉ PRÁCE (ZBP)

Název ZBP v češtině	Automatická administrační rozhraní nad relační databází
Název ZBP v angličtině	Automatic administrative interfaces for relational databases
Studijní obor	Informační technologie
Akademický rok	2010/2011
Vedoucí závěrečné práce	Tomáš Holas
Termín odevzdání Zadání ZBP	07.10.2010
Termín odevzdání práce ZBP	13.05.2011

Cíl závěrečné bakalářské práce

Cílem je vytvořit přehled stávajících administračních rozhraní nad relačními databázemi pro nejrozšířenější jazyky pro webové aplikace (php, python, ruby). Vysvětlit obecné principy fungování takových rozhraní, jejich výhody a nevýhody. Na základě stanovených hodnotících kritérií provést testy a porovnat jejich výsledky. Určit adekvátnost použití jednotlivých rozhraní pro rozdílné typy projektů. Nakonec vytvořit vlastní mini-framework s rozhraním pro mobilní zařízení s využitím javascriptových knihoven jQuery a jQTouch.

Základní literatura

The Django Book - Adrian Holovaty and Jacob Kaplan-Moss
Zend Framework in Action - Allen Rob
Agile Web Development with Rails .3rd Edition .2009 - Sam Ruby , Dave Thomas , David Heinemeier Hansson
Začínáme programovat v Ruby on Rails - Steven Holzner

Dokumentace k jednotlivým projektům:
<http://rubyonrails.org/documentation>
<http://merbivore.com/documentation.html>
<http://www.sinatrarb.com/documentation>

<http://www.zend.com/en/resources/zend-documentation/>
<http://nette.org/cs/dokumentace>
<http://kohanaframework.org/documentation>

<http://docs.djangoproject.com/en/1.2/>

2. ABSTRAKT

Cílem práce je přiblížit problematiku tvorby administračního rozhraní nad relační databází a ukázat, zda je možné smysluplné použití aplikace tohoto typu na mobilních přístrojích. Výsledkem je přehled přístupů k tvorbě administračních rozhraní a detailní rozbor a srovnání těch nepoužívanějších, zejména z hlediska uživatelského rozhraní. Součástí práce je i aplikace přizpůsobená pro použití na současných chytrých telefonech, na které lze ověřit použitelnost konceptu mobilní administrace nad relační databází a ukázat možnosti využití moderních nástrojů pro vývoj softwaru.

Klíčová slova: Relační databáze, administrační rozhraní, mobilní zařízení, Ruby on Rails, jQuery, jQTouch

3. ABSTRACT

The aim of the thesis is to bring a closer look on the issue of creating administration interface for relational database and demonstrate possible meaningful use of this type of application for mobile devices. The result is an overview of attitudes to creating administration interfaces and a detailed analysis and comparison of the most used ones especially from the point of view of the user interface. Also a part of the thesis is an application adjusted for smart phones which can be used to verify the usability of the mobile administration for relational database concept and to demonstrate possibilities for utilizing modern tools for software development.

Keywords: Relational databases, administration interface, mobile devices, Ruby on Rails, jQuery, jQTouch

4. PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Automatické administrační rozhraní nad relační databází jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou v práci citovány a jsou též uvedeny v seznamu literatury a použitých zdrojů.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb.

V Praze dne

.....

Martin Vondráček

5. PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Tomáši Holasovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

6. OBSAH

1. Zadání.....	3
2. Abstrakt.....	4
3. Abstract.....	5
4. Prohlášení.....	6
5. Poděkování.....	7
6. Obsah.....	8
7. Úvod.....	10
8. Typy Administračních rozhraní.....	11
8.1 Textové rozhraní.....	11
8.2 Grafické rozhraní.....	12
8.3 Webové rozhraní.....	13
9. Existující implementace administračních rozhraní.....	15
9.1 phpMyAdmin.....	15
9.2 Adminer.....	18
9.3 PhpPgMyAdmin.....	20
10. Implementace se zaměřením na mobilní zařízení.....	22
10.1 Případy užití.....	22
10.1.1 UC1 – Zobrazení dat v databázi.....	22
10.1.2 UC2 – Vložení dat do databáze.....	23
10.1.3 UC3 – Vyhledávání dat v databázi.....	24
10.1.4 UC4 – Úprava dat v databázi.....	25
10.1.5 UC5 – Mazání dat z databáze.....	26
10.2 Kritéria výsledné aplikace.....	26
10.3 Použité technologie.....	27
10.3.1 Ruby.....	27
10.3.2 Ruby on Rails.....	28
10.3.3 MySQL.....	29
10.3.4 jQuery.....	30
10.3.5 JQTouch.....	31
10.4 Architektura/Design aplikace.....	35
10.5 Návrh uživatelského rozhraní.....	36
10.5.1 Index.....	36
10.5.2 show_databases.....	37
10.5.3 show_tables.....	37
10.5.4 table_detail.....	37
10.5.5 table_info.....	38
10.5.6 show_data.....	39
10.5.7 insert_data.....	39
10.5.8 edit_data.....	40
10.5.9 Search.....	40
10.5.10 Problémy spojené s návrhem a implementací uživatelského rozhraní.....	41
10.6 Popis tříd a funkcí.....	41
10.6.1 TableInfo.....	41
10.6.2 ColumnInfo.....	42
10.6.3 MobminDao.....	43
10.6.4 MobminController.....	44
10.6.5 Specifická řešení.....	45
10.6.6 Vyhodnocení dle stanovených kritérií.....	45
11. Závěr.....	47
12. Conclusion.....	48
13. Seznam použité literatury.....	49
14. Seznam použitých symbolů a zkratk.....	50

15. Seznam obrázků.....	51
16. Seznam tabulek.....	52
17. Seznam příloh.....	53
Příloha 1 – Cd s aplikací.....	54

7. ÚVOD

Účelem této práce je ukázat, jak lze přizpůsobit tradiční webové aplikace, v tomto případě administrační rozhraní pro relační databáze, novým trendům v informačních technologiích. Novými trendy lze chápat zejména masivní rozvoj mobilních zařízení jako chytrých telefonů a tabletů, pro které není vždy možné využívat stávající aplikace způsobem, na který je uživatel zvyklý. To je způsobeno technickými parametry těchto zařízení, zejména menšími obrazovkami a zcela odlišným způsobem ovládání.

Cílem je vytvořit webovou aplikaci takovou, která umožní spravovat databázi a přistupovat k ní způsobem obdobným, jako při použití klasického tenkého klienta nebo klientské aplikace, ale bude zároveň snadno ovladatelná právě z mobilního zařízení a bude tak přístupná opravdu odkudkoliv. Práce by měla zároveň odpovědět na otázku, zda je vůbec možné a žádoucí aplikaci pro správu databáze používat na mobilních zařízeních. Výsledek bude tedy zaměřen hlavně na uživatelské rozhraní, přehlednost a snadnou ovladatelnost a neklade si za cíl přinést zásadní změny v oblasti správy relačních databází.

Důvodem pro volbu Administračního rozhraní pro relační databáze je masivní využití tohoto typu aplikace především tvůrci webu a správci obsahu nebo serverů, ale zároveň nedostatek aplikací nebo určitá jednotvárnost při řešení této problematiky.

Výsledná aplikace je napsána za použití programovacího jazyka Ruby verze 1.9.2, aplikačního rámce Ruby on Rails verze 3.0.5, javascriptové knihovny jQuery verze 1.5.1 s rozšířením jQTouch, které se zaměřuje na vývoj mobilních aplikací, zejména pak na chytré telefony iPhone. Při výběru technologií použitých pro vývoj aplikace byly rozhodujícími faktory rychlost a jednoduchost vývoje při zachování kvality výsledné aplikace.

V první části práce budou představeny různé přístupy ke správě databází, popis a srovnání některých hotových a v praxi využívaných řešení této problematiky. V další části budou popsány nejen technologie použité při vývoji, ale i další technologie používané při vývoji webových aplikací pro mobilní zařízení, dále návrh a jednotlivé části řešené aplikace a nakonec bude vyhodnocen úspěch nebo neúspěch aplikace podle stanovených kritérií.

8. TYPY ADMINISTRAČNÍCH ROZHRAŇÍ

Problematika databází a databázových systémů je problematika velice obsáhlá a pojem administrace nebo správa databáze nabývá různých významů, podle toho z jakého úhlu pohledu se na něj pohlíží. Obecně lze administraci databáze chápat buď jako správu a údržbu database management systému (DBMS) nebo-li česky systému řízení báze dat nebo ji lze chápat jako správu dat uložených pomocí tohoto systému. DBMS můžeme zjednodušeně brát jako systém, který zajišťuje přístup k datům uloženým v databázi nebo jako rozhraní, které nám umožní s těmito daty pracovat a přistupovat k nim z aplikačních programů nebo pomocí administračního rozhraní. Správa databáze potom znamená správu těchto systémů, tedy jejich instalaci, konfiguraci, zálohování nebo obnovu dat, logování, monitoring, optimalizaci běhu a zajišťování bezpečnosti. Mezi nejvýznamnější DBMS patří Oracle Database¹ (aktuální verze je 11g), Microsoft SQL Server², MySQL³ nebo DB2⁴ od firmy IBM. V této práci bude administrace databáze chápána jako správa dat pomocí DBMS, tzn., že databází budou chápána data uložená v tabulkách. Automatické administrační rozhraní tak bude tvořit další vrstvu nad DBMS a umožňovat práci s daty a ovládání DBMS.

8.1 Textové rozhraní

Existuje několik možností, jak přistupovat ke správě databáze. Prvním z přístupů je ovládání pomocí příkazové řádky. Z hlediska uživatelského rozhraní se jedná o práci v čistě textovém rozhraní a to jak na vstupu tak na výstupu. Pomocí zadávání příkazů do příkazové řádky se volají funkce DBMS, který vypisuje výsledky volání na obrazovku. Tímto způsobem lze naplno využít všech možností, které nám DBMS nabízí. Na první pohled jsou ale vidět nevýhody tohoto přístupu. Uživatel musí alespoň přiměřeně znát a ovládat možnosti, které mu DBMS poskytuje. Musí znát alespoň základní příkazy a správnou syntaxi pro ty, které chce využívat. Vzhledem k rozsáhlým možnostem, které současné databázové systémy nabízejí to však znamená věnovat značný čas jejich studiu a to je velkou překážkou pokud se uživatel nechce zaměřovat pouze na správu databáze a stačí mu využívat pouze základní funkce. To je případ zejména tvůrců menších webových aplikací, nebo webových prezentací, které uchovávají v databázi relativně malé množství dat a databáze není kritickým místem fungování těchto aplikací. Takoví uživatelé nevyužívají pokročilé funkce databází jako například ladící funkce nebo o takových funkcích ani nevědí.

1 <http://www.oracle.com/cz/products/database/index.html>

2 <http://www.microsoft.com/sqlserver/en/us/default.aspx>

3 <http://www.mysql.com/>

4 <http://www-01.ibm.com/software/data/db2/>

Další překážkou v užívání jsou rozdíly mezi implementacemi jednotlivých databázových systémů. To co funguje v MySQL nemusí nutně fungovat v databázi Oracle nebo MS SQL. Toto se týká zejména pokročilých funkcí databází, protože základní příkazy by měli odpovídat standardům SQL. Záměrně je napsáno "by měli", protože ne vždy se tvůrci databází těchto standardů drží[1]. Tento přístup má však i své výhody. Pokud uživatel má dostatečné vědomosti o DBMS, který využívá, nepotřebuje ke správě využívat už žádný jiný nástroj, stačí pouze přístup k systému nebo databázovému stroji (například přes ssh) a dostatečná oprávnění pro potřebné operace. Práce s příkazovou řádkou zároveň ulehčuje správu více databázových systémů, protože uživateli stačí skript, případně sada skriptů, které pak lze spouštět pro každou z databází. Shrneme-li tedy tento přístup, ovládání DBMS přes příkazovou řádku znamená vysokou produktivitu práce při minimální uživatelské přívětivosti a při vyšších nárocích na znalosti databázových systémů jak obecně, tak konkrétních implementací.

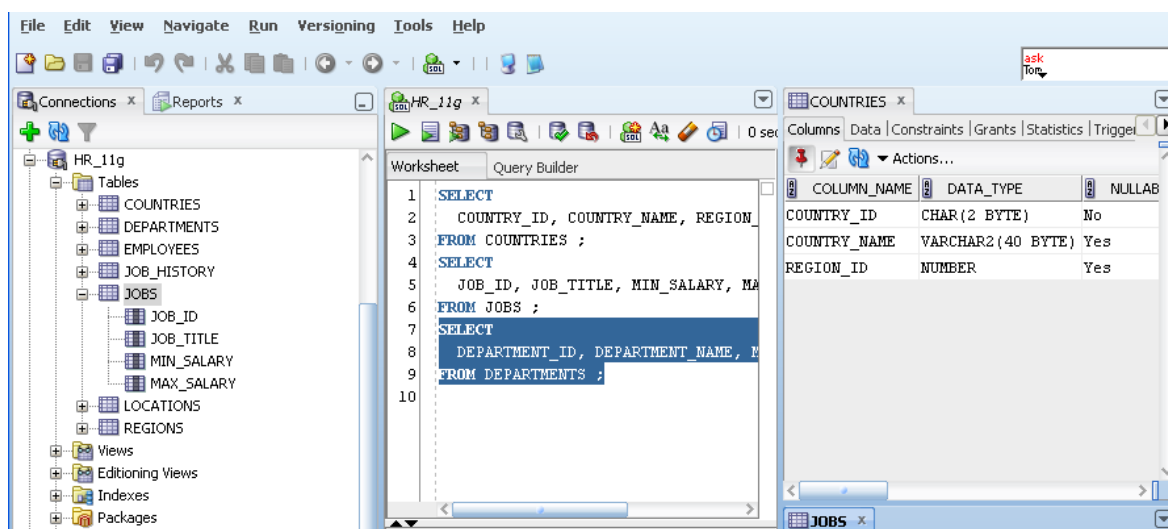
```
mysql> describe rdb.Category;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id         | bigint(20) | NO   | PRI | NULL    |       |
| name       | varchar(50) | NO   | UNI | NULL    |       |
| description | text       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Obrázek 1: Textové rozhraní databáze MySQL

8.2 Grafické rozhraní

Dalším přístupem ke správě databází je využití desktopové aplikace. Jedná se vlastně o tlustého klienta s propracovaným grafickým uživatelským rozhráním. Tento přístup již klade na uživatele nižší nároky týkající se znalostí DBMS, resp. jednotlivých funkcí a příkazů, kterými se vykonávají. Toto břemeno na sebe berou tvůrci aplikace, kteří připravují uživatelské rozhraní nabízející přístup k funkcím pro správu databáze. Toto rozhraní již bývá více uživatelsky přívětivé a i méně zkušený uživatel dokáže ovládat DBMS i bez předchozích znalostí. Uživatelské rozhraní také pokrývá kompletní možnosti, které databázové systémy nabízejí a často mají i funkce, kterých nelze v textovém rozhraní dosáhnout, například grafy zobrazující zátěž databáze v čase apod. Uvedme nevýhody, které tento přístup přináší. Především uživatel si musí na svůj počítač takovou aplikaci nainstalovat, což může přinášet problémy a na uživatele jsou kladeny určité požadavky. Očekává se, že bude mít hardware dostatečně výkonný, aby byl schopen aplikaci spustit nebo že bude disponovat patřičným operačním systémem potřebným pro běh aplikace. V neposlední řadě je zde

otázka bezpečnosti. Klientský počítač je nutné dostatečně zabezpečit před škodlivými programy, které by mohly získat přístupové údaje do databáze, případně se z klientského počítače dostat na databázový server, kde by mohly napáchat nevratné škody na datech. Pomocí konkrétní aplikace se také dají ovládat pouze vybrané databázové systémy. Tedy například s nástrojem SQL Developer od společnosti Oracle nelze ovládat databáze od společnosti Microsoft, případně nástroje pro správu databáze MySQL nelze využít pro správu PostgreSQL a naopak. Pokud uživatel využívá několika různých druhů DBMS, musí mít nainstalováno několik aplikací. To samozřejmě neplatí vždy (výjimkou budiž například program Navicat⁵ ve verzi Premium, podporující MySQL, MS SQL Server, SQLite, Oracle a PostgreSQL). V kategorii tlustých klientů lze najít celou škálu aplikací od těch nejjednodušších umožňujících základní operace pro vkládání, mazání a úpravu dat jako např. SQLite administrator⁶ pro databázi SQLite až po aplikace užívané ve firemním prostředí, které kromě funkcí týkajících se vlastního databázového systému nabízejí i další pokročilé nástroje například pro návrh databáze a propojení s dalšími podnikovými systémy. Takovou aplikací je např. Oracle Enterprise Manager 11g for Database⁷.



Obrázek 2: Grafické rozhraní programu SQL Developer společnosti Oracle

8.3 Webové rozhraní

Posledním z přístupů jsou automatická administrační rozhraní. Jedná se o webové aplikace, které po spuštění nad databází vygenerují uživatelské rozhraní umožňující správu databáze podobně jako při využití tlustého klienta. Jedná se tedy o řešení, které přenáší klientskou aplikaci

5 http://www.navicat.com/en/products/navicat_premium/premium_overview.html

6 <http://sqliteadmin.orbmu2k.de/>

7 <http://www.oracle.com/us/products/enterprise-manager/database-management/index.html>

do prostředí internetového prohlížeče a spojuje výhody plynoucí z používání grafického rozhraní s výhodami plynoucími z podstaty webových aplikací. Uživatel není omezen v přístupu ke správě zařízení nebo operačním systémem, který využívá. K přístupu do databáze stačí webový prohlížeč a přístup k internetovému připojení.

Tvůrce aplikace má pak v závislosti na použitém programovacím jazyku možnost využít různých konektorů pro jednotlivé implementace relačních databází a naprogramovat aplikaci tak, že ji lze využít pro správu různých databázových systémů. Toto řešení ovšem klade na tvůrce značné nároky zejména při návrhu aplikace a v praxi se tvůrci až na výjimky zaměřují na jednu konkrétní implementaci DBMS.

Je nutné ovšem připomenout i nevýhody, které s sebou přístup do databáze přes webový prohlížeč přináší. Možnosti grafického uživatelského rozhraní jsou limitovány schopnostmi jazyka HTML, které jsou proti uživatelským rozhraním desktopových aplikací značně omezené. Uživatel by měl také využívat prohlížeč, který bude výslednou aplikaci správně zobrazovat a podporovat použité technologie a pro tvůrce to znamená testování pro různé prohlížeče včetně jejich různých verzí. Nevýhodou je také, že výběr funkcí databáze, které se objevují v aplikaci, je pouze rozhodnutím tvůrce aplikace a je velice složité obsáhnout ve webovém rozhraní kompletně všechny funkce, které databázové systémy nabízejí a pokud uživatel potřebuje některou z pokročilých funkcí, nezbyvá mu, než se uchýlit k práci s desktopovou aplikací nebo příkazovou řádkou.

V následující části textu budou představeny některé rozšířené webové aplikace používané pro správu databází.

9. EXISTUJÍCÍ IMPLEMENTACE ADMINISTRAČNÍCH ROZHRAŇÍ

9.1 phpMyAdmin

Nejznámější implementací automatického administračního rozhraní je bezesporu phpMyAdmin⁸. Tento softwarový nástroj je určený pro správu databáze MySQL a je šířený pod licencí GNU/GPL v2 a jak název napovídá je napsaný v programovacím jazyce PHP, což je také jeden z důvodů jeho velkého rozšíření. PhpMyAdmin vznikl v roce 1998. Jeho autorem je Tobias Ratschiller, který projekt v roce 2001 opustil. PhpMyAdmin si mezitím vybudoval širokou základnu uživatelů i přispěvatelů, a proto ještě v témže roce vznikl „The phpMyAdmin project“, který měl za úkol koordinovat další vývoj a stále tak činí.

V současnosti se aplikace nachází ve verzi 3.4.3.1. Ke svému běhu potřebuje PHP verze 5.2.0 nebo novější s podporou session, rozšířením Standard PHP Library (SPL) a podporou JSON.

Podporovanou databází je MySQL verze 5.0 a novější.

PhpMyAdmin má v současnosti širokou škálu funkcí. Kromě vytváření, mazání, prohlížení a úpravy databází, tabulek, sloupců, dat a indexů a dalších operací umožňuje[2]

- údržbu serverů, databází a tabulek, s návrhy na konfiguraci serveru
- provádět, upravovat a ukládat libovolné SQL-příkazy, a to i dávkově
- vytvářet a číst dump tabulek
- export dat do různých formátů: CSV, XML, PDF, ISO / IEC 26300 - OpenDocument Text a tabulky, Word, Excel a LATEX formáty
- import dat a MySQL konstrukcí z aplikace Microsoft Excel, tabulek OpenDocument, XML, CSV a SQL souborů
- správu více serverů
- správu uživatelů a oprávnění
- export návrhu databáze do formátu PDF
- synchronizaci dvou databází umístěných na stejném nebo na dvou různých serverech

8 http://www.phpmyadmin.net/home_page/

PhpMyAdmin se je v současnosti dostupný v 62 jazycích[3] (tento údaj je ale nutno brát s rezervou vzhledem k tomu, že přibližně polovina jazykových mutací není přeložena ani z poloviny). Výhodou je také rozsáhlá dokumentace včetně knihy⁹ od jednoho z autorů, které uživatelům ulehčují používání phpMyAdminu a správu databáze. K dispozici je i česká verze knihy¹⁰, která je ovšem zaměřena na verzi 2.6.0.

K nevýhodám patří značná velikost aplikace a také nutnost instalace aplikace, a to buď manuální¹¹ s nutností zásahu do konfigurace PHP a nebo s pomocí instalačního skriptu¹².

Uživatelské rozhraní v posledních verzích prošlo změnami, týkajícími se hlavně přehlednosti a zvýšení uživatelského pohodlí. Obrazovka je rozdělena na dvě hlavní části, navigační rám umístěný v levé části a hlavní okno. V horní části navigačního rámu se nachází odkazy na obecné funkčnosti jako odhlášení z aplikace, zobrazení okna pro zadávání SQL příkazů, dokumentaci phpMyAdmin, MySQL dokumentaci a znovunačtení rámu. Pod těmito odkazy je menu pro snadnou navigaci mezi dostupnými databázemi a tabulkami. Hlavní okno aplikace je v úvodní obrazovce rozděleno do celků podle oblastí, kterých se týkají, na obecné nastavení, nastavení uživatelského rozhraní, informace o databázovém a webovém serveru a samotné aplikaci. Nově je zobrazeno výraznější a přehlednější ohraničení těchto částí, které ovšem může v uživateli evokovat možnost si tyto informační boxy rozmístit libovolně na obrazovce, což ovšem není možné.

V horní části hlavního okna je umístěna tzv. drobečková navigace (Breadcrumb navigation) pro snadný pohyb mezi jednotlivými databázemi a tabulkami. Pod touto navigací jsou odkazy relativní k aktuální úrovni zanoření do databáze. Pokud je uživatel připojen k serveru, má možnost zobrazení statusu databáze a detailních informací o datových přenosech, statistikách SQL dotazů a přehledu databázových proměnných, jejich aktuálních hodnot, popisu každé proměnné a odkazu do dokumentace MySQL. Tyto proměnné jsou rozděleny do skupin, které jsou odkazovány v úvodu obrazovky. Uživatel tak může rychle kontrolovat nastavení databázového serveru. Další informace, které jsou k dispozici je přehled uživatelů a nastavení jejich práv, export a import databází, přehled podporovaných úložišť, nastavení replikace serveru, synchronizaci s jiným serverem a nastavení aplikace.

Nastavení aplikace umožňuje značnou modifikaci vzhledu aplikace, a to jak navigačního rámu, tak hlavního okna aplikace. Uživatel si tak může nastavit, zda chce zobrazit jednotlivé databáze a tabulky jako seznam nebo je chce uspořádat do stromové struktury, maximální počet zobrazených tabulek nebo velikost zanoření stromové struktury, oddělovače, zobrazování popisu tabulky místo názvu nebo zvýrazňování řádků pod kurzorem myši. U nastavení hlavního okna potěší zejména možnost upravovat editační rozhraní. Uživatel může nastavit například zda chce

9 Delisle Marc, Mastering phpMyAdmin 3.3.x for Effective MySQL Management

10 Delisle Marc, Pokorný Jan, RNDr., phpMyAdmin – efektivní správa MySQL, Zoner Press 2004

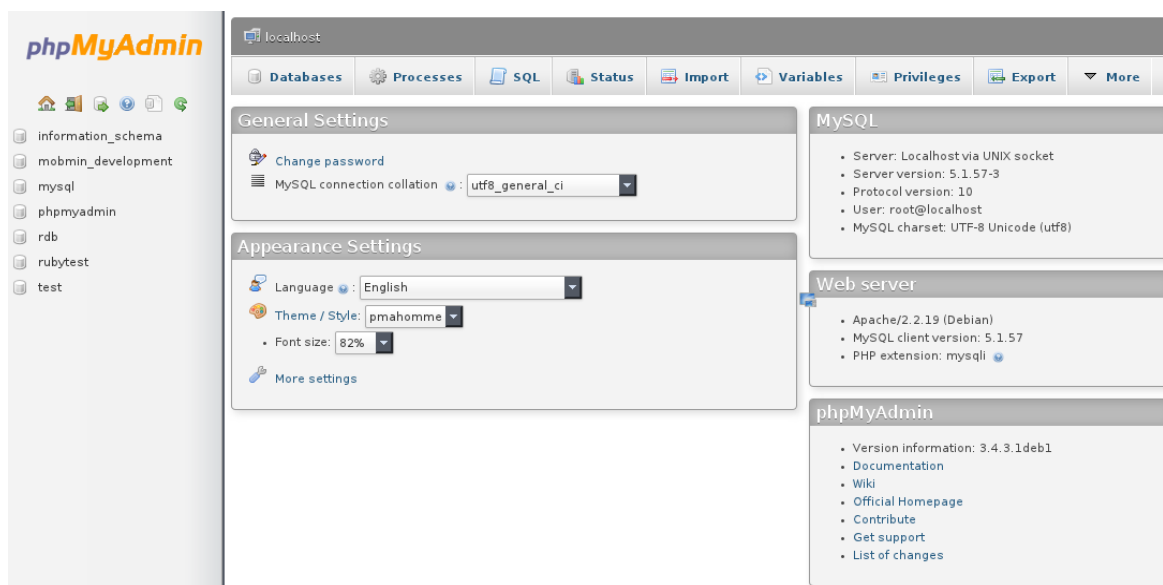
11 http://www.phpmyadmin.net/documentation/#quick_install

12 http://www.phpmyadmin.net/documentation/#setup_script

pro textové sloupce s omezenou velikostí použít HTML element *input* nebo *textarea* a zároveň může u elementu *textarea* nastavit požadovanou velikost. Nastavení aplikace lze exportovat a importovat a je tedy možné přenášet oblíbené nastavení.

Pokud se uživatel nachází na detailu tabulky, má k dispozici odkazy na její procházení, vkládání nových dat, import a export dat, operace s tabulkou a tracking (sledování). Vkládání a editace dat probíhá pomocí zobrazení samostatné stránky s formulářem. V posledních verzích přibyla i možnost tzv. inline editace. Znamená to, že uživatel zobrazí data v tabulce a po výběru inline editace může měnit data přímo, bez přechodu na další stránku. Tento způsob úpravy dat je rychlejší a pohodlnější. Pokud chce uživatel využít funkci sledování tabulky, vybere atributy, které chce sledovat a vytvoří počáteční verzi. Pokud později změní některý ze sledovaných atributů, uloží se nová verze tabulky a uživatel může pouhým kliknutím na odkaz přecházet mezi jednotlivými verzemi.

Ačkoliv phpMyAdmin nabízí širokou možnost úpravy uživatelského rozhraní a již v základu nabízí několik grafických témat, jeho použití na mobilních zařízeních je nepohodlné. I na velkých monitorech jsou některé obrazovky nepřehledné (zejména přehled dat v tabulce a odkazy na operace s daty) a kvůli nutnosti na malém displayi neustále obrazovku posouvat do stran se práce s aplikací stává pomalou a těžkopádnou.



Obrázek 3: Úvodní obrazovka aplikace phpMyAdmin

9.2 Adminer

Odlišný přístup k tvorbě administračního rozhraní nad relační databází zvolil Ing. Jakub Vrána¹³ při tvorbě administračního rozhraní nazvaného Adminer. Prvotním impulsem pro vznik této aplikace byla autorova potřeba spravovat databáze na serverech, které nepovolují vzdálená připojení a neochota kvůli několika úpravám v datech nahrávat na server poměrně velkou aplikaci phpMyAdmin. Myšlenka pro vznik Admineru (původní název byl phpMinAdmin) tedy byla "Chtělo by to aplikaci se všemi základními správcovskými funkcemi, která by se vešla do jednoho nepříliš velkého souboru."^[4] Vznikla tak minimalistická aplikace v jednom souboru, aktuálně o velikosti 344 kB pro vícejazyčnou, resp. 206 kB pro českou verzi (autor upustil od původního požadavku vejít se do velikosti souboru 100 kB).

Adminer je napsán v PHP a přes svou minimální velikost disponuje širokým rozsahem funkcí:^[5]

- Připojení k databázovému serveru pod zadaným uživatelským jménem a heslem
- Výběr databáze, vytvoření nové databáze
- Seznam sloupců, indexů, cizích klíčů a triggerů tabulky
- Změna názvu, úložiště, porovnávání, auto_increment a komentáře k tabulce
- Změna názvu, typu, porovnávání, komentáře a výchozí hodnoty sloupců
- Přidání a mazání tabulek a sloupců
- Vytvoření, změna, smazání a vyhledávání podle indexů včetně fulltextových
- Vytvoření, změna, smazání a propojení seznamů podle cizích klíčů
- Vytvoření, změna, smazání a získání dat z pohledů
- Vytvoření, změna, smazání a volání uložených procedur a funkcí
- Vytvoření, změna a smazání triggerů
- Výpis dat s možností vyhledávání, třídění a omezení počtu vypisovaných záznamů
- Vložení, úprava a smazání záznamu
- Podpora všech datových typů, práce s BLOB pomocí nahrávání souborů
- Provedení libovolného SQL příkazu zadaného přímo nebo nahraného ze souboru
- Export struktury tabulek, dat, pohledů, uložených procedur a databází do SQL nebo CSV
- Alter export pro přenesení změn do produkční databáze

¹³ <http://www.vrana.cz/vrana.html>

- Schéma struktury databáze s vazbami podle cizích klíčů
- Seznam procesů s možností jejich ukončení
- Přehled uživatelů a práv s možností jejich nastavení
- Přehled proměnných s odkazy do dokumentace
- Správa událostí a rozdělených tabulek (MySQL 5.1)
- Schéma, sekvence, uživatelské typy (PostgreSQL)

Sám autor Admineru se nebrání srovnání s phpMyAdmin a poukazuje na vlastnosti, které Adminer nabízí navíc. Oproti samotné databázi MySQL u phpMyAdmin je to podpora databázových systémů MySQL, SQLite, PostgreSQL, MS SQL a Oracle. Požadovaná verze PHP je 4.3 a vyšší, verze MySQL je 4.1 a vyšší. Adminer tak umožňuje nasazení i na serverech, kde z jakýchkoliv důvodů běží starší verze PHP a MySQL. V případě phpMyAdmin musí uživatel nasadit starší verzi s možnými chybami. Instalace Admineru probíhá tím nejjednodušším možným způsobem - soubor `adminer.php` se zkopíruje do vybraného adresáře na serveru.

Přirozeně se naskytuje otázka, jakým způsobem autor dosáhl minimální velikosti výsledné aplikace. Odpovědí je kompilace a minifikace zdrojového kódu. Zdrojový kód aplikace ve vývojové verzi se nachází v klasické adresářové struktuře v několika souborech a adresářích logicky rozdělených na databázové ovladače, jazyky, statické soubory (javascript, obrázky, css soubory) a jádro aplikace tak, aby byl kód přehledný, udržitelný a dále rozšiřovatelný. Následně se spustí soubor `compile.php`, který všechny soubory spojí do jednoho, upraví je, například zkrácením uživatelských proměnných a funkcí a jejich nahrazením kratšími identifikátory pomocí funkce `token_get_all` a odstraní komentáře a prázdné znaky. Hlavním důvodem malé velikosti Admineru ale stále zůstává úsporně napsaný kód programu.

Uživatelské rozhraní aplikace Adminer je v porovnání s ostatními aplikacemi velmi střídmé. Horní a pravá část stránky slouží k rychlé navigaci. V horní části je hierarchická navigace ve směru Databázový systém → adresa serveru → jméno databáze → jméno tabulky. V levé části se nachází odkazy na vložení SQL příkazu, export databáze/tabulky, odhlášení z aplikace a pod těmito statickými odkazy je HTML element `select` pro výběr databází. Po výběru databáze se pod tímto elementem zobrazí odkazy na jednotlivé tabulky ve formátu `select jméno_tabulky`. Po výběru `select` se zobrazí data z tabulky, po výběru `jména_tabulky` se zobrazí informace o tabulce, jednotlivých sloupcích, indexech, cizích klíčích a triggerech. Tento systém dovoluje uživateli během dvou kliknutí přejít na jakoukoliv tabulku v dostupných databázích.

V hlavní části stránky se v závislosti na tom, kde se uživatel nachází, zobrazují dostupné informace a odkazy na možné akce relevantní k zobrazované úrovni. Na hlavní stránce je to tedy přehled databází s údaji o způsobu porovnávání řetězců, počtem tabulek v jednotlivých databázích

a s nabídkou vytvoření nové databáze nebo úpravy stávající. S přechodem na tabulku se zobrazí informace o jednotlivých sloupcích a nabídka k tvorbě nebo úpravě tabulky, obdobně pak při přechodu na jednotlivé sloupce.

Adminer nenabízí žádnou možnost měnit styl aplikace přes vlastní rozhraní, a přestože na klasickém stolním počítači je uživatelské rozhraní velmi dobře použitelné, při použití na mobilním zařízení se aplikace chová jako běžná stránka a její použití není uživatelsky přívětivé. Řešením pro tuto situaci je vytvoření nového souboru CSS stylů přizpůsobených pro malé displaye. Adminer možnost přidávání nových stylů velmi dobře podporuje a nabízí jich několik k dispozici. Instalace nových stylů probíhá v duchu celé aplikace prostým přejmenováním nového souboru na adminer.css a nakopírováním do stejného adresáře, ve kterém se Adminer nachází.

MySQL » localhost » Databáze: rdb

Adminer 3.3.1 3.3.2

Databáze: rdb

SQL příkaz Export Odhlásit

rdb

Vytvořit novou tabulku

vypsat Author
vypsat Category
vypsat Product
vypsat Product_Category
vypsat Product_Shop
vypsat Shop
vypsat Type

Pozměnit databázi Schéma databáze Oprávnění

Tabulky a pohledy

Vyhledat data v tabulkách: Vyhledat

Tabulka	Úložiště	Porovnávání	Velikost dat	Velikost indexů	Volné místo	Auto Increment	Řádků	Komentář
Author	MyISAM	utf8_general_ci	144	2 048	0		3	
Category	MyISAM	utf8_general_ci	244	3 072	0		3	
Product	MyISAM	utf8_general_ci	964	5 120	0		12	
Product_Category	MyISAM	utf8_general_ci	350	5 120	0		14	
Product_Shop	MyISAM	utf8_general_ci	783	4 096	0		27	
Shop	MyISAM	utf8_general_ci	340	2 048	0		3	
Type	MyISAM	utf8_general_ci	224	3 072	0		3	
7 celkem	MyISAM	utf8_general_ci	3 049	24 576	0			

Analyzovat Optimalizovat Zkontrolovat Opravit Vypřázdňit Odstranit

Přesunout do jiné databáze: rdb Přesunout Zkopírovat

Vytvořit tabulku Vytvořit pohled

Procedury a funkce

Vytvořit proceduru Vytvořit funkci

Události

Vytvořit událost

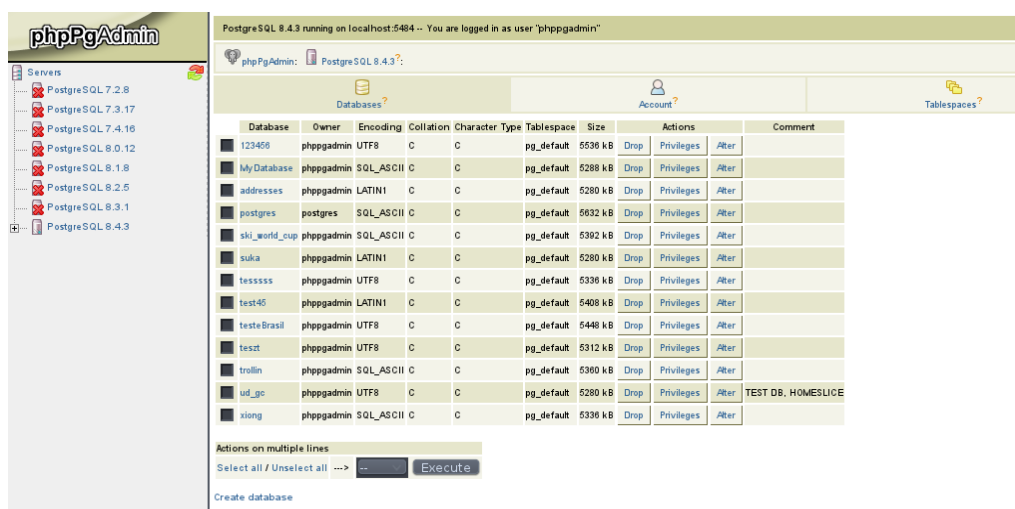
Obrázek 4: Uživatelské rozhraní aplikace Adminer v české verzi

9.3 PhpPgMyAdmin

Pro úplnost krátce zmíním ještě aplikaci phpPgAdmin. Vznikla původně jako port aplikace phpMyAdmin, ale s podporou pro databázový server PostgreSQL. V roce 2002 byla kompletně přepsána, a přestože uživatelům nabízí takřka shodné funkce, momentálně s phpMyAdmin nesdílí žádný kód. Podporované verze PostgreSQL jsou 7.4.x až 9.0.x.

Uživatelské rozhraní se od phpMyAdmin v zásadě neliší. Stránka je rozdělena na dva rámce. V levém sloupci je přehled možných serverů a databází, ke kterým má uživatel přístup a u každé

databáze informace o schématech, jazycích a „castech“. Tento přehled je řešen pomocí stromové struktury. V hlavním rámci v horní části jsou odkazy na podřízené uzly v závislosti na aktivním uzlu stromu a v hlavní části jsou o něm zobrazeny informace. Uživatelské rozhraní phpPgAdmin není přehledné. Na vině je zvolená velikost písma, která znesnadňuje čtení odkazů. Při hlubším zanoření do stromové struktury se také musí upravovat šířka levého rámce tak, aby byly údaje čitelné. Při zobrazení na menším displayi se phpPgAdmin stává prakticky nepoužitelným.



Obrázek 5: Uživatelské rozhraní programu phpPgAdmin

Dalšími programy, které nabízejí stejnou nebo podobnou funkčnost jako výše uvedené jsou MyWebSQL¹⁴ nebo SQL Buddy¹⁵, který již není aktivně vyvíjen. V obou případech se opět jedná o kombinaci PHP a MySQL.

¹⁴ <http://mywebsql.net/>

¹⁵ <http://www.sqlbuddy.com/>

10. IMPLEMENTACE SE ZAMĚŘENÍM NA MOBILNÍ ZAŘÍZENÍ

Funkčností, které by mělo webové administrační rozhraní splňovat je velké množství a každý uživatel má o tom, co by měla aplikace umět zcela jiné představy. Implementace všech funkcností administračního rozhraní značně přesahuje rozsah bakalářské práce, a proto bude tato práce omezena pouze na základní funkčnosti, které musí bezpodmínečně umět každá aplikace, která se stará o správu databází, a to jsou: procházení databází, tabulek a samotných dat uložených v tabulkách, jejich úprava, mazání a vkládání nových záznamů. Uživatel by měl také mít možnost vyhledávat podle zadaných kritérií. Přehled těchto funkcností zachycují následující případy užití nebo-li „use cases“.

10.1 Případy užití

10.1.1 UC1 – Zobrazení dat v databázi

Název	Zobrazení dat v databázi
Popis	Uživatel po přihlášení vybere požadovanou databázi, zvolí tabulku, vybere možnost „Show data“ a zobrazí data uložená v databázi.
Aktéři	Uživatel
Základní tok	<ol style="list-style-type: none"> 1. Systém zobrazí seznam dostupných databází 2. Uživatel vybere databázi 3. Systém zobrazí seznam tabulek v databázi 4. Uživatel vybere tabulku 5. Systém zobrazí nabídku akcí, které může uživatel provést 6. Uživatel vybere možnost zobrazit data 7. Systém zobrazí seznam údajů uložených v tabulce
Alternativní toky	<ol style="list-style-type: none"> 1. Uživatel nemá práva na prohlížení databází <ol style="list-style-type: none"> 1.1. Systém zobrazí informaci, že uživatel nemá potřebná práva. 3. Uživatel nemá práva na prohlížení tabulek <ol style="list-style-type: none"> 3.1. Systém zobrazí informaci, že uživatel nemá potřebná práva. 7. Uživatel nemá práva na zobrazování záznamů v tabulce <ol style="list-style-type: none"> 7.1. Systém zobrazí informaci, že uživatel nemá potřebná práva.
Podmínky spuštění	Uživatel je úspěšně přihlášen
Stav po ukončení	Aplikace zobrazuje obrazovku s daty z databáze Aplikace je připravena k dalšímu použití

Tabulka 1: Use case 1 - zobrazení dat v databázi

10.1.2 UC2 – Vložení dat do databáze

Název	Vložení dat do databáze
Popis	Uživatel po přihlášení vybere požadovanou databázi a zvolí tabulku, do které chce vkládat data. Po výběru možnosti „Insert data“ se zobrazí formulář pro zadání údajů do databáze, který uživatel po vyplnění odešle.
Aktéři	Uživatel
Základní tok	<ol style="list-style-type: none"> 1. Systém zobrazí seznam dostupných databází 2. Uživatel vybere databázi 3. Systém zobrazí seznam tabulek ve vybrané databázi 4. Uživatel vybere tabulku 5. Systém zobrazí nabídku akcí, které může uživatel provést 6. Uživatel vybere možnost vložit data 7. Systém zobrazí formulář pro vložení údajů 8. Uživatel vyplní formulář 9. Uživatel odešle formulář 10. Systém uloží data do databáze 11. Systém zobrazí hlášení o úspěšném uložení dat
Alternativní toky	<ol style="list-style-type: none"> 1. Uživatel nemá práva na prohlížení databází <ol style="list-style-type: none"> 1.1. Systém zobrazí informaci, že uživatel nemá potřebná práva. 3. Uživatel nemá práva na prohlížení tabulek <ol style="list-style-type: none"> 3.1. Systém zobrazí informaci, že uživatel nemá potřebná práva. 7. Uživatel nemá práva na vkládání dat do databáze <ol style="list-style-type: none"> 7.1. Systém zobrazí chybovou hlášku 10. Při vkládání dat došlo k chybě <ol style="list-style-type: none"> 10.1. Systém zobrazí chybovou hlášku a zobrazí formulář pro vložení údajů
Podmínky spuštění	Uživatel je úspěšně přihlášen
Stav po ukončení	Aplikace zobrazuje obrazovku s daty z databáze Aplikace je připravena k dalšímu použití

Tabulka 2: Use case 2 - vložení dat do databáze

10.1.3 UC3 – Vyhledávání dat v databázi

Název	Vyhledávání dat v databázi
Popis	Uživatel po přihlášení vybere požadovanou databázi a zvolí tabulku, ve které chce vyhledávat data. Po výběru možnosti „Search“ se zobrazí jednoduchý formulář, umožňující zadat hledaný výraz a vybrat sloupec, ve kterém se bude vyhledávat.
Aktéři	Uživatel
Základní tok	<ol style="list-style-type: none"> 1. Systém zobrazí seznam dostupných databází 2. Uživatel vybere databázi 3. Systém zobrazí seznam tabulek ve vybrané databázi 4. Uživatel vybere tabulku 5. Systém zobrazí nabídku akcí, které může uživatel provést 6. Uživatel vybere možnost Vyhledat data 7. Systém zobrazí formulář pro vložení hledaného řetězce a výběr sloupce ve kterém chce uživatel vyhledávat 8. Uživatel vyplní formulář 9. Uživatel odešle formulář 10. Systém zobrazí výsledek hledání
Alternativní toky	<ol style="list-style-type: none"> 1. Uživatel nemá práva na prohlížení databází <ol style="list-style-type: none"> 1.1. Systém zobrazí informaci, že uživatel nemá potřebná práva 3. Uživatel nemá práva na prohlížení tabulek <ol style="list-style-type: none"> 3.1. Systém zobrazí informaci, že uživatel nemá potřebná práva 10. Uživatel nemá práva na prohlížení dat v tabulce <ol style="list-style-type: none"> 10.1. Systém zobrazí informaci, že uživatel nemá potřebná práva
Podmínky spuštění	Uživatel je úspěšně přihlášen
Stav po ukončení	Aplikace zobrazuje obrazovku s vybranými daty z databáze Aplikace je připravena k dalšímu použití

Tabulka 3: Use case 3 - Vyhledávání dat v databázi

10.1.4 UC4 – Úprava dat v databázi

Název	Úprava dat v databázi
Popis	Uživatel po přihlášení vybere požadovanou databázi a tabulku, ve které chce upravovat data. Pomocí postupu popsaného v UC1 nebo UC3 vybere požadovaná data a po výběru možnosti „Edit“ se zobrazí formulář, pro úpravu požadovaných dat.
Aktéři	Uživatel
Základní tok	<ol style="list-style-type: none"> 1. Uživatel zobrazí data z tabulky databáze (UC1 nebo UC3) 2. Systém k zobrazeným datům zobrazí nabídku akcí 3. Uživatel vybere možnost Upravit data 4. Systém zobrazí formulář pro úpravu dat s vyplněnými údaji z databáze 5. Uživatel změní potřebná data 6. Uživatel odešle formulář 7. Systém aktualizuje odeslaná data
Alternativní toky	<ol style="list-style-type: none"> 7. Uživatel nemá práva na úpravu dat <ol style="list-style-type: none"> 7.1. Systém zobrazí informaci, že uživatel nemá potřebná práva 7. Při ukládání aktualizovaných dat do databáze došlo k chybě <ol style="list-style-type: none"> 7.1. Systém zobrazí informaci o chybě 7.2. Systém zobrazí formulář pro úpravu dat s vyplněnými údaji
Podmínky spuštění	Uživatel je přihlášen, úspěšně dokončený UC1 nebo UC3
Stav po ukončení	Aplikace zobrazuje obrazovku s daty z databáze Aplikace je připravena k dalšímu použití

Tabulka 4: Use case 4 - Úprava dat v databázi

10.1.5 UC5 – Mazání dat z databáze

Název	Mazání dat z databáze
Popis	Uživatel po přihlášení vybere požadovanou databázi a tabulku, ze které chce mazat údaje. Postupem, který popisuje UC1 nebo UC3 vybere požadovaná data a po výběru možnosti „Delete“ a následném potvrzení, že chce údaje skutečně smazat odstraní data z databáze.
Aktéři	Uživatel
Základní tok	<ol style="list-style-type: none"> 1. Uživatel zobrazí data z tabulky databáze (UC1 nebo UC3) 2. Systém k zobrazeným datům zobrazí nabídku akcí 3. Uživatel vybere možnost Smazat data 4. Systém zobrazí ověřující dotaz na smazání dat 5. Uživatel potvrdí dotaz 6. Systém smaže data z databáze
Alternativní toky	<ol style="list-style-type: none"> 5. Uživatel nepotvrdí dotaz <ol style="list-style-type: none"> 5.1. Systém nevykoná žádnou akci, zobrazí obrazovku s daty 6. Uživatel nemá práva na mazání dat <ol style="list-style-type: none"> 6.1. Systém zobrazí informaci, že uživatel nemá potřebná práva
Podmínky spuštění	Uživatel je přihlášen, úspěšně dokončený UC1 nebo UC3
Stav po ukončení	Aplikace zobrazuje obrazovku s daty z databáze Aplikace je připravena k dalšímu použití

Tabulka 5: Use case 5 - Mazání dat z databáze

10.2 Kritéria výsledné aplikace

Hlavním kritériem určujícím úspěch nebo neúspěch řešené aplikace je použitelnost a přizpůsobení uživatelského rozhraní pro dotykové ovládání. Všechny informace musí být přehledně zobrazeny tak, aby je uživatel nemusel přibližovat a následně posouvat obrazovku. Pokud to bude nutné, mělo by být možné se posouvat vždy jen v jednom směru (vertikálně nebo horizontálně). Jednotlivé logické části od sebe musí být odděleny pro zvýšení celkové přehlednosti. Ovládací prvky musí být dostatečně velké kvůli pohodlnému dotykovému ovládání a rozmístění navigačních tlačítek by mělo mít jednotný koncept. Odlišeny musí být aktivní a neaktivní prvky. Samozřejmostí je unifikovaný vzhled všech částí aplikace.

Vzhledem k tomu, že mobilní zařízení nemají vždy dostupné kvalitní internetové připojení, musí být množství přenášených dat omezeno na minimum. Zvýší se tak zároveň odezva celé aplikace.

Dalším kritériem úspěšnosti je rychlost vývoje. Pro splnění tohoto kritéria je důležitý výběr použitých technologií.

10.3 Použité technologie

Pro implementaci automatického administračního rozhraní byl zvolen programovací jazyk Ruby, verze 1.9.2 a aplikační rámec Ruby on Rails verze 3.0.5. Aplikace poběží nad databází MySQL verze 5.1 a vyšší. Pro integraci uživatelského rozhraní do prostředí operačního systém iOS (chytrý telefon iPhone a tablet iPad) jsou použity: javascriptová knihovna jQuery a zásuvný modul jQueryTouch.

V následujícím textu budou popsány jednotlivé technologie.

10.3.1 Ruby

Ruby je interpretovaný, dynamicky typovaný, objektově orientovaný programovací jazyk, vydaný pod otevřenou licencí „Ruby license“¹⁶, kompatibilní s licencí GPL¹⁷. Autorem je japonský programátor Yukihiro Matsumoto. První verze jazyka byla vydána v roce 1995. Motivací bylo vytvořit jazyk výkonnější než Perl¹⁸ a více objektově orientovaný než Python¹⁹[6]. Výsledkem je programovací jazyk s jednoduchou, snadno čitelnou a naučitelnou syntaxí, ve kterém je všechno objekt. Například definice třídy Person by mohla vypadat následovně:

```
class Person
  attr_accessor :first_name, :surname

  def initialize(first_name, surname)
    @first_name = first_name
    @surname = surname
  end

  def introduce
    puts "Hello, my name is #{@first_name} #{@surname}"
  end
end

new_person = Person.new("Jan", "Novak")
new_person.introduce
new_person.first_name = "Pavel"
puts new_person.first_name + " " + new_person.surname
```

Výstupem bude:

```
Hello my name is Jan Novak
Pavel Novak
```

¹⁶ <http://www.ruby-lang.org/en/LICENSE.txt>

¹⁷ <http://www.gnu.org/licenses/old-licenses/gpl-1.0.txt>

¹⁸ <http://www.perl.org>

¹⁹ <http://www.python.org>

Ukázkou zjednodušujícího přístupu Ruby je první řádek v definici třídy, kde nám příkaz `attr_accessor` umožní přímo přistupovat k instančním proměnným třídy `Person` aniž bychom museli definovat metody k jejich přiřazení nebo čtení (jejich použití je vidět na posledních dvou řádcích). Jazyk Ruby jsem zvolil vzhledem k jeho vlastnostem při vývoji aplikací. Dalšími důvody pro tuto volbu jsou absence nástroje ke správě databází napsaného v Ruby, která by vývojářům umožnila přístup přes webový prohlížeč bez nutnosti mít na serveru nainstalovanou podporu pro další jazyk a framework Ruby on Rails.

10.3.2 Ruby on Rails

Ruby on Rails (RoR) je framework pro tvorbu webových aplikací využívající návrhový vzor Model-View-Controller. Zaměřen je hlavně na produktivitu při vývoji. Mezi jeho hlavní principy patří DRY (Don't repeat yourself), Convention over configuration a REST (Representational State Transfer).

Hlavní myšlenkou DRY[7] přístupu je redukovat duplicitu informací v systému. Zvláště v případech vícevrstevných architektur je důležité, aby jednotlivé části aplikace měli své pevně dané umístění v jednotlivých vrstvách a nevyskytovaly se na několika místech. Dodržení DRY principu umožňuje lepší správu velkých aplikací, kdy změna jedné komponenty by neměla ovlivnit zbytek aplikace. Při dodržování DRY principu vývojářům pomáhají i generátory kódu, v případě RoR kromě jiných například „Scaffold“ generátor, který podle zadaných parametrů vygeneruje základní kostru aplikace, které se vývojář může (ale nemusí) držet.

Myšlenkou principu Convention over configuration, konvence před konfigurací, je zbytečně nezatěžovat vývojáře konfigurací. Tvůrce aplikace je nucen konfigurovat jednotlivé části pouze pokud se potřebuje odchýlit od obecných konvencí aplikace. Například třída „Train“ bude udržovat informace v tabulce „trains“, pouze pokud se tvůrce aplikace rozhodne, jinak je nucen název tabulky specifikovat, např. „locomotives“. Výhodou je, že při vývoji se vývojář může soustředit na specifická řešení dané aplikace a nemusí se věnovat (zejména v úvodních částech vývoje) opakujícím se konfiguračním krokům, ale zároveň neztrácí potřebnou flexibilitu. Tento přístup můžeme dnes najít v mnoha dalších aplikačních frameworkcích jako Spring, Kohana, Maven, Grails.

REST je architektonický styl pro distribuované systémy[8], který umožňuje přistupovat k datům (obecně ke zdrojům), které jsou identifikovány pomocí url (obecně identifikátorem zdrojů) přes standardní metody http. Mějme například entitu `person`. Pro získání této entity použijeme `GET /person/23`, kde 23 bude jednoznačná informace pro určení požadované instance entity `person`, například id. Pro vytvoření nové entity `person` použijeme metodu `POST`, pro smazání metodu `DELETE` a pro změnu metodu `PUT`. Výhodou použití Ruby on Rails je, že vývojář nemusí řešit problémy spojené s implementací protokolu HTTP v prohlížečích, například že neznají metodu `DELETE`[9].

10.3.3 MySQL

MySQL je celosvětově nejpoblárnější open-source relační databáze[10], momentálně spravovaná firmou Oracle. Podporuje replikaci, uložené procedury, trigery, pohledy. Disponuje konektory pro rozšířené programovací jazyky a je dostupná na víc než 20 různých platformách a operačních systémech, včetně Unixu, Linuxu a Windows. Nabízí několik formátů datových úložišť: InnoDB, MyISAM, NDB (MySQL cluster), paměťové úložiště, csv a další. Nejpoužívanější datová úložiště jsou MyISAM a InnoDB.

MyISAM²⁰ (do verze 5.5.5 bylo nastaveno jako výchozí) ukládá každou tabulku do tří souborů, každý se jménem tabulky a příponou .frm (informace o formátu tabulky), .MYD (data) a .MYI (indexy). Data uložená pomocí MyISAM jsou přenositelná mezi různými platformami a systémy (kompatibilita na binární úrovni), velikost datového úložiště je maximálně 256TB, na systémech, které to podporují umožňuje ukládat soubory do délky souboru 63 bitů. Tabulka může obsahovat $(2^{32})^2$ řádků, 64 indexů s maximálním počtem 16 sloupců na index. Důležitým faktem je, že datové úložiště MyISAM nepodporuje transakční zpracování dat.

Datové úložiště InnoDB (výchozí od verze 5.5.5) na rozdíl od MyISAM podporuje ACID transakční zpracování. ACID je zkratkou pro Atomicity – atomicitu, Consistency – konzistenci, Isolation – izolovanost a Durability – trvalost.

Atomicita vyjadřuje, že činnost, kterou databáze během transakce vykonává, se provádí jako celek a buď se úspěšně provede, nebo se v případě chyby v průběhu neprovede vůbec.

Vlastnost konzistence určuje, že databáze bude před začátkem a po skončení transakce v konzistentním stavu a do databáze budou zapsána pouze validní data.

Izolovanost zaručuje, že změny v datech, které probíhají během transakce nejsou viditelné pro okolí.

Trvalost znamená, že změny, které jsou provedeny v datech během úspěšné transakce, jsou trvale uloženy do databáze.

Velikost úložiště InnoDB je omezena 64TB. Tabulky a indexy jsou na rozdíl od MyISAM ukládány v tablespaces v několika souborech nebo diskových oddílech a mohou tak dosahovat velkých datových objemů i na souborových systémech, kde je velikost souboru omezena na 2GB.

Vlastnosti jednotlivých datových úložišť jsou při zpracování administračních rozhraní důležité, protože například při absenci transakčního zpracování operací na straně databáze musí tvůrce aplikace zabezpečit konzistenci a integritu dat.

20 <http://dev.mysql.com/doc/refman/5.5/en/myisam-storage-engine.html>

10.3.4 jQuery

jQuery je javascriptová knihovna ulehčující vývoj webových aplikací s podporou javascriptu. Její hlavní výhody jsou: snadné procházení DOM struktury HTML dokumentu, zpracování událostí a práce s ajaxem, manipulace s CSS, animace, utility a přidávání nových funkcionalit pomocí pluginů.

Jak jQuery funguje? Do HTML stránky se do hlavičky přidá pomocí elementu *link* odkaz na soubor s jQuery knihovnou (komprimovaná verze má cca 30KB). Po načtení stránky a zavolání inicializace jQuery načte DOM elementy pomocí enginu Sizzle²¹ do pole a přiřadí jim základní funkce. Pokud tvůrce aplikace potřebuje přidat nějakému elementu další funkce, jQuery nalezne tento element/elementy v poli a funkci jim přiřadí. Pro výběr DOM elementů se používají selektory stejně jako v CSS.

```
$(document).ready(function() {  
    $("p").css('background-color', 'red');  
});
```

Na předchozím příkladu lze ukázat jednoduchost použití jQuery a zároveň způsob, jakým se provádí obsluha událostí a práce s elementy a css. Znak `$` slouží jako alias pro jmenný prostor jQuery a konstrukce `$(document).ready()` je tedy volání funkce `ready()` na objektu `document`. Tato funkce zpracovává událost načtení celého stromu DOM elementů a jako argument přijímá funkci, která se má při této události provést. V tomto případě se vyberou všechny elementy `p` a pomocí funkce `.css(css_atribut, hodnota_atributu)` se nastaví pozadí na červenou barvu.

Kromě samotné knihovny jQuery je tvůrcům webových aplikací k dispozici i jQuery UI²². Tato část nabízí k použití již hotové prvky uživatelského rozhraní, na které jsou uživatelé zvyklí z desktopových aplikací, ale jejichž převedení do prostředí webových prohlížečů není snadné. Jde zejména o výběr data pomocí kalendáře, dialogová okna, zobrazení záložek (tabů) uvnitř těla stránky, posuvníky nebo zobrazení průběhu události. Tvůrci aplikací mají k dispozici i nástroje pro práci s jednotlivými HTML elementy jako změna velikosti, posun elementů pomocí chycení a tažení myši (drag and drop), výběr nebo řazení položek seznamu. U všech těchto částí lze navíc jednoduše měnit barevná schémata a rychle je tak přizpůsobit designu vytvářené aplikace.

jQuery disponuje rozsáhlou dokumentací a příklady použití pro každou část, kterou nabízí. Stále se také vyvíjí nové funkce a opravují chyby. K dispozici je i repositář pluginů²³ od různých autorů,

21 <http://sizzlejs.com/>

22 <http://jqueryui.com/>

23 <http://plugins.jquery.com/>

kteří řeší všechny oblasti webu od funkcí ajaxu, validace a odesílání formulářů až po přehrávání audiovizuálního obsahu.

Podporované prohlížeče jsou MS Internet Explorer 6.0+, Mozilla Firefox 2.0+, Apple Safari 3.0+, Opera 9.0+ a Google Chrome.

10.3.5 JQTouch

Poslední technologií použitou při vytváření administračního rozhraní je jQTouch. Jedná se o plugin pro jQuery zaměřený na vývoj webových aplikací pro mobilní zařízení iPhone, iPad, iPod Touch a mobilní zařízení s operačním systémem android s prohlížečem založeným na jádře WebKit²⁴. Pomocí tohoto pluginu lze jednoduše dosáhnout vzhledu vytvářené aplikace tak, aby vypadala a chovala se jako nativní iPhone/iPad/iTouch program.

Použití je obdobné jako u jQuery. V hlavičce HTML dokumentu se jako atribut elementu *link* přidá soubor s jQuery, další s jQTouch pluginem a CSS styly potřebné pro správný vzhled. Nakonec se jQTouch inicializuje pomocí funkce `new $.jqTouch()`, které se mohou předat inicializační parametry jako použité grafické téma nebo selektory pro různé akce, pokud se tvůrce potřebuje odchýlit od běžného nastavení.

JQTouch funguje tak, že má připravenou sadu funkcí, vzhledů a animací navázaných na různé HTML elementy, nebo atributy *class* a *id* a tvůrci aplikace pak stačí tyto elementy nebo atributy přidávat, naplnit daty a o vzhled se nemusí dále starat.

Ačkoliv mohou tvůrci používat jakékoliv elementy a atributy (v rámci standardů), v duchu přístupu „konvence před konfigurací“ mají bez nutnosti cokoli nastavovat k dispozici následující prvky:

²⁴ <http://www.webkit.org/>

Element	Popisek
div.toolbar	Lišta v horní části stránky. Vhodná pro umístění elementu <h1>, pro navigaci po aplikaci (tlačítko back) a pro další doplňující tlačítka (tlačítko „about“ atd.).
div.toolbar > a.back	Tlačítko pro zpětnou navigaci na stránkách. Přejít je zajišťován javascriptem.
div.info	Lišta ve spodní části stránky. Vhodná pro umístění dodatečných informací (datum, nápověda).
ul	Menu pro navigaci, nebo zobrazování informací jako tabulky. V závislosti na atributu class se nastavuje téma vzhledu. Základní možnosti jsou „rounded“, „plastic“, „edgetoedge“.
ul.individual	Třída „individual“ nastyluje neřazený seznam jako tlačítko s poloviční šířkou stránky.
li.arrow li.forward	Třída „arrow“ nebo „forward“ zobrazuje jednotlivé položky netříděného seznamu jako klikatelné (přidává na konec různé druhy šipek).
h2	Odděluje jednotlivé části stránky.
Form > a.submit	Odkaz je nastylován jako tlačítko a odesílá formulář.

Tabulka 6: Přehled nastýlovaných HTML elementů při použití jQTouch

Použití různých animací pro přechod mezi jednotlivými stránkami je stejně jednoduché. K odkazům stačí přiřadit patřičnou třídu. Přehled možných animací a způsob použití je uveden v následující tabulce:

Název animace	Použití	Popis
Slide	a	Základní animace, posun o obrazovku vpravo
Slide up	a.slideup	Překrytí aktuální obrazovky zespoda
Dissolve	a.dissolve	Rozplynutí aktuální obrazovky
Fade	a.fade	Prolínání
Flip	a.flip	Otočení aktuální obrazovky o 180°
Pop	a.pop	Nová obrazovka se zvětší z prostřední části aktuální obrazovky
Swap	a.swap	Výměna obrazovek
Cube	a.cube	Pootočení kostky

Tabulka 7: Přehled animací jQTouch

Používání pluginu jQTouch vyžaduje od tvůrců aplikací, aby dodržovali určitou strukturu HTML dokumentu. Při práci se používá pouze jeden HTML dokument a jednotlivé podstránky jsou obaleny elementem *div* s atributem *id* a jsou již zahrnuty v tomto dokumentu. Každá podstránka

pak má vlastní prvky *h1*, *div.toolbar* atd. Při prvním požadavku na server se tak načtou všechny podstránky a o přesuny mezi jednotlivými částmi se již stará jQTouch bez nutnosti nahrávat další data ze serveru. Otázkou je, jak prohlížeč pozná, na které stránce má začít. Odpovědí je atribut *class* s hodnotou *current*. Element s touto třídou smí být v dokumentu pouze jeden. Při tvorbě se umístí u elementu *div*, který představuje výchozí stránku a při přechodu mezi stránkami se jQTouch stará o to, aby byla tato třída správně přesouvána mezi jednotlivými podstránkami.

Pro názornost uvedu příklad toho, jak by vypadal zdrojový kód těla HTML dokumentu s hlavní stránkou a dvěma podstránkami.

```
<body>

  <div id="page1">      <!-- zacatek stranky 1 -->
    <div class="toolbar">
      <a class="back" href="#">back</a>
      <h1>Subpage 1</h1>
    </div>
    Welcome to subpage 1
    <div class="info">
      You are currently on subpage nr. 1
    </div>
  </div>

  <div id="page2">      <!-- zacatek stranky 2 -->
    <div class="toolbar">
      <a class="back" href="#">back</a>
      <h1>Subpage 2</h1>
    </div>
    Welcome to subpage 2
    <div class="info">
      You are currently on subpage nr. 2
    </div>
  </div>

  <div id="home" class="current"> <!-- uvodni stranka -->
    <div class="toolbar">
      <h1>Home page</h1>
    </div>
    <ul class="rounded">
      <li class="arrow">
        <a class="fade" href="#page1">Subpage nr. 1</a>
        <a class="fade" href="#page2">Subpage nr. 2</a>
      </li>
    </ul>
    <div class="info">
      You are currently on homepage
    </div>
  </div>

</body>
```

Pokud by ovšem použití jQTouch vyžadovalo, aby byly všechny informace načteny najednou, omezilo by se využití pouze na statické prezentace a pro aplikace s dynamicky generovaným obsahem by byl plugin nepoužitelný. JQTouch samozřejmě nabízí řešení. Pokud je potřeba odkazovat na stránku, která není součástí úvodního dokumentu, lze to udělat obvyklým způsobem. Pokud totiž odkaz vede na adresu vně dokumentu (samozřejmě na stejném serveru), jQTouch tuto stránku načte na pozadí pomocí ajaxu. Výsledek volání by pak měl obsahovat pouze vnitřní část HTML dokumentu mezi elementy *div* (stejně jako podstránky v původním dokumentu). Stejným způsobem je řešeno i odesílání a zpracovávání formulářů. Tento přístup umožňuje vytváření webových aplikací způsobem stejným jako doposud a zároveň minimalizuje objem dat přenášovaných mezi klientem a serverem.

Přechod mezi stránkami pomocí ajaxového volání s sebou ovšem nese i problémy. Jiným způsobem se musí řešit například zpracování událostí pomocí jQuery. Pokud v úvodním HTML dokumentu zavoláme funkci, která na nějakou událost, např. kliknutí, navěsí specifickou funkci, tato funkce se provede pouze při kliknutí na elementy, které byly součástí původního dokumentu a na výsledcích ajaxového volání již nebude fungovat. Je to dáno způsobem, jakým jQuery řeší navěšení funkcí na elementy. Prohlížeč nejprve načte původní dokument. V momentě, kdy je načten, se jako výsledek funkce `$(document).ready()` zavolá navěšení funkcí na elementy. To probíhá tak, že jQuery zavolá vnitřní funkci `find()`, která načte celý strom DOM elementů, nalezne požadované a k těm přiřadí novou uživatelskou funkci. Pokud ale následně přidáme ke stromu elementů další část jako výsledek ajaxového volání, přiřazení požadované funkce již neproběhne. Tento problém lze vyřešit pomocí callbacku, který po ukončení ajaxového požadavku znovu zavolá navěšení požadované funkce na nové elementy, nebo lze použít funkci `live()`²⁵.

K dispozici jsou dvě základní grafická témata, která je možné upravit a přizpůsobit třeba požadovaným firemním barvám, nebo lze vytvořit téma zcela nové. Plugin také nabízí možnost přidávat vlastní rozšíření stejným způsobem, jakým se přidávají do jQuery.

JQTouch je poměrně skoupý na dokumentaci²⁶.

Alternativou k použití při vývoji pro mobilní zařízení je knihovna jQuery mobile, která řeší stejnou problematiku podobným způsobem. Na rozdíl od jQTouch, který se zaměřuje na výrobky s operačním systémem rodiny iOS, se jQuery mobile snaží být multiplatformním řešením a kromě iOS podporuje i operační systémy Android, Windows Phone, Symbian, MeeGo, bada, BlackBerry a palm webOs. Bohužel ze snahy o širokou podporu napříč operačními systémy plynou problémy a na různých systémech je podpora ze strany jQuery mobile na různé úrovni. Odladit výslednou aplikaci tak, aby na různých přístrojích fungovala stejně, je značně náročné na čas při vývoji

25 <http://api.jquery.com/live/>

26 <http://code.google.com/p/jqtouch/>

a testování (nehledě na nutnost pořízení různého hardwaru). I proto byl zvolen plugin jQTouch, který se zaměřuje pouze jedním směrem.

10.4 Architektura/Design aplikace

Implementace řešení automatického administračního rozhraní vychází z použití frameworku Ruby on Rails a dodržuje architektonický vzor Model-View-Controller (MVC). Tento návrhový vzor rozděluje aplikaci do tří vrstev, datového modelu, uživatelského rozhraní a řídicí logiky podle funkce, kterou mají jednotlivé části vykonávat. Rozdělení aplikace do jednotlivých vrstev umožňuje její lepší správu a provádění změn v jedné z vrstev s minimálním dopadem na další.

Vrstva Model slouží pro definici objektů/entit, s kterými aplikace pracuje. V této vrstvě je také implementována aplikační logika a zajišťuje komunikaci s perzistentním datovým úložištěm (databází), pokud je to potřeba. Tato vrstva je nezávislá na dalších dvou vrstvách.

Vrstva View slouží pro komunikaci aplikace a uživatele. Zobrazuje výsledek uživatelských akcí, sbírá data od uživatele a provádí validace dat na straně klienta před odesláním na server.

Vrstva Controller přijímá uživatelské požadavky a data z View a na základě těchto požadavků a dat inicializuje akce ve vrstvě Model a vytváří odpovědi.

Průběh obsloužení uživatelského požadavku ve webové aplikaci s konceptem MVC probíhá následovně:

1. Uživatel provede na webové stránce nějakou akci (odešle formulář, stiskne tlačítko)
2. Controller dostane o této akci oznámení
3. Na základě požadované akce vyvolá Controller příslušné volání do třídy Model, případně předá data z View.
4. Model vykoná požadovanou akci / zpracuje předaná data
5. Controller zavolá zobrazení patřičného View
6. Uživateli se zobrazí stránka s aktualizovanými údaji.

Vytvářené administrační rozhraní dodržuje rozložení do vrstev MVC. V následujících kapitolách budou jednotlivé přiblíženy.

10.5 Návrh uživatelského rozhraní

Vzhledem k tomu, že uživatelské rozhraní a jeho přizpůsobení pro mobilní zařízení je hlavním přínosem vytvářené aplikace, začnu popis právě vrstvou View a uživatelským rozhraním.

Specifickým problémem při vytváření aplikace pro malé displaye je nedostatek místa pro zobrazení informací tak, aby byly snadno čitelné a současné použití aktivních prvků tak, aby jejich použití bylo co nejjednodušší. To, co se na klasickém displayi vejde na jednu obrazovku je potřeba rozdělit na více částí a uživatel se mezi nimi musí přesouvat. Je proto důležité, aby tato nutnost přecházení mezi stránkami byla eliminována maximální ergonomií celé aplikace.

Při implementaci řešení jsem pro navigaci použil hierarchickou stromovou strukturu s úrovněmi zanoření ve směru databáze → tabulka → nabídka akcí pro tabulku → záznam v tabulce. Tedy uživatel vidí přehled dostupných databází, po výběru databáze vidí dostupné tabulky a po výběru tabulky může pracovat s daty. Pro maximální uživatelské pohodlí je každá úroveň zanoření řešená jako samostatná obrazovka. Uživatel tak vidí jednoduchý přehled informací a dostupných akcí, které může snadno dotykem vybrat. Pohodlí při dotykovém ovládání na malém display a snadná čitelnost informací byly tedy hlavní určující prvky při řešení jednotlivých obrazovek.

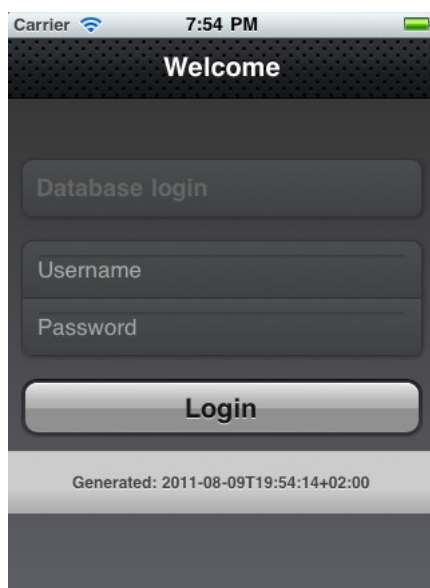
V aplikaci jsou použity dvě šablony pro tvorbu výsledného HTML kódu. První šablona (`application.html.erb`) je použita pro úvodní stránku. Obsahuje všechny elementy nutné pro validní HTML kód, zejména pak hlavičku, ve které se načítají javascriptové knihovny jQuery a jQTouch, soubor s jejich inicializací a použité kaskádové styly. Druhá šablona (`inner.html.erb`) pak obsahuje pouze část HTML dokumentu uzavřenou mezi elementy `div` a částí, které jsou stejné na všech stránkách tedy `div.toolbar`, `div.info`, `a.back`, `a.logout` a `ul.errors` pro výpis chybových hlášení. Toto řešení je použité kvůli ajaxovému volání jQTouch, které při GET nebo POST požadavcích pouze připojí obsah dokumentu ke stávajícímu a není nutné znovu nahrávat javascript a css. V některých prohlížečích, pro které není jQTouch uzpůsoben to také způsobuje podivné neočekávané chování v podobě opakujících se požadavků na server.

10.5.1 Index

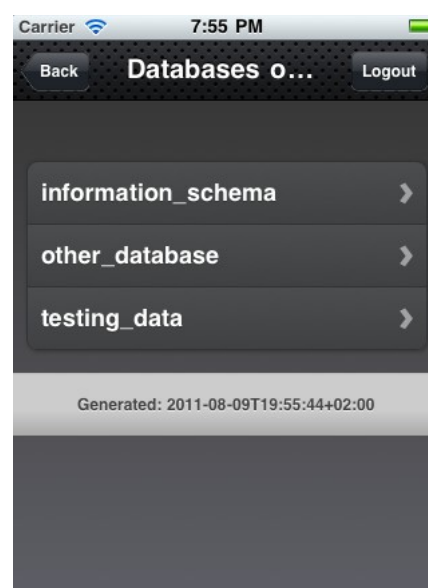
Úvodní obrazovka aplikace obsahuje formulář pro zadání uživatelského jména a hesla pro přístup do databázového serveru. V případě neúspěšného přihlášení se zobrazuje chybová hláška, kterou vypsál server. V případě úspěšného přihlášení je uživatel přesměrován na další obrazovku.

10.5.2 show_databases

Na této obrazovce je výpis databází dostupných pro přihlášeného uživatele. Výpis je řešen formou seznamu. U každé položky seznamu je šipkou naznačena možnost výběru. Po výběru dotykem je uživatel přesměrován na další obrazovku.



Obrázek 6: Přihlašovací obrazovka



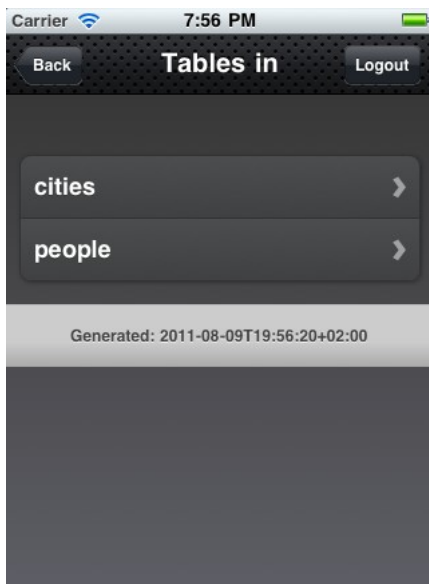
Obrázek 7: Přehled dostupných
databází

10.5.3 show_tables

Na této obrazovce je opět formou seznamu zobrazen výpis dostupných tabulek ve vybrané databázi. Po výběru je uživatel přesměrován na další obrazovku.

10.5.4 table_detail

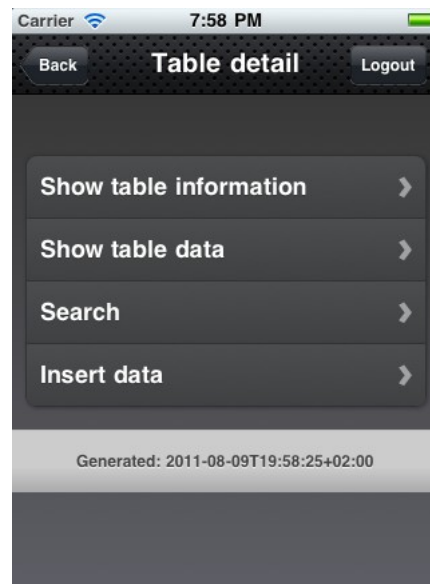
Nabídka akcí spojených s vybranou tabulkou. Uživatel má možnost výběru „Show table information“, „Show table data“ a „Insert data“.



Obrázek 8: Přehled tabulek v databázi

10.5.5 table_info

Zobrazení detailních informací o tabulce. Informace jsou rozděleny do seznamů oddělených popisem. Prvním seznamem jsou obecné informace. Počet řádků a sloupců v tabulce, použité úložiště a druh porovnávání řetězců. Další seznamy jsou potom informace o jednotlivých sloupcích. U každého sloupce je zobrazen název, datový typ, defaultní hodnota, pokud přidány, zobrazují se u sloupců komentáře, hodnota autoincrement a indexy. Pokud je sloupec primárním klíčem, je zvýrazněn.



Obrázek 9: Nabídka akcí po výběru databáze



Obrázek 10: Informace o tabulce



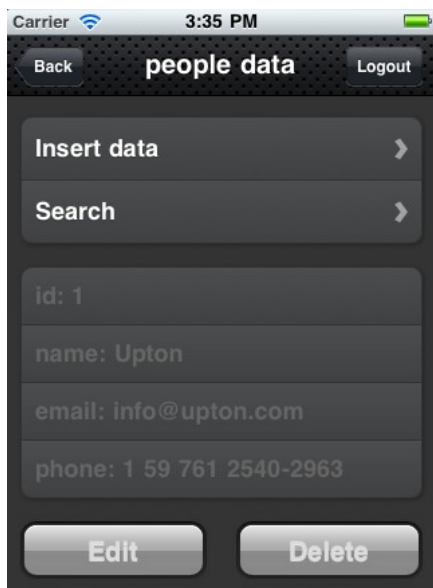
Obrázek 11: Informace o sloupcích

10.5.6 show_data

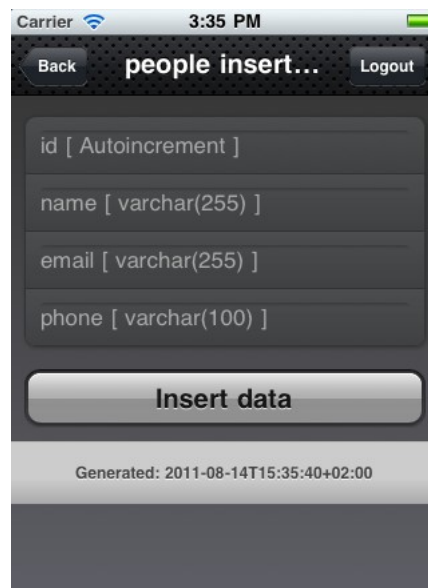
Na této obrazovce jsou dostupná data uložená v databázi. Každý záznam je reprezentován odděleným seznamem, kde položky seznamu odpovídají sloupcům tabulky. Každý sloupec je zobrazen ve formátu jméno_sloupce: hodnota. Pro každou řádku jsou k dispozici tlačítka „Edit“ a „Delete“. Na začátku seznamu dat jsou odkazy na vložení a vyhledávání záznamu.

10.5.7 insert_data

Formulář pro vložení záznamu do databáze. Každý sloupec je opět zobrazen formou položky seznamu. Pro lepší uživatelskou orientaci je v atributu *placeholder* u každé položky vyplněn název sloupce a datový typ. Po výběru sloupce tyto atributy zmizí a uživatel může rovnou vkládat data. Toto řešení je výhodné, jelikož šetří místem na obrazovce a uživatel nemusí formulář posouvat, nebo ho musí posouvat méně než v případě zobrazení informací ve zvláštní položce. Po úspěšném odeslání formuláře na server je uživatel přesměrován na pohled show_data. Pokud při ukládání záznamu dojde k chybě, je formulář zobrazen znovu, v úvodu stránky je výpis chybového hlášení.



Obrázek 12: Přehled záznamů v databázi



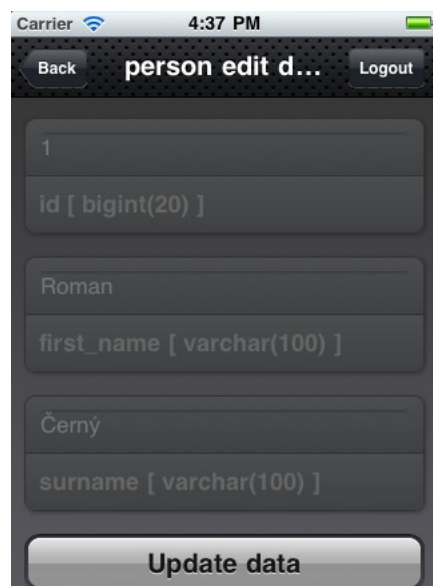
Obrázek 13: Formulář pro vložení záznamu

10.5.8 edit_data

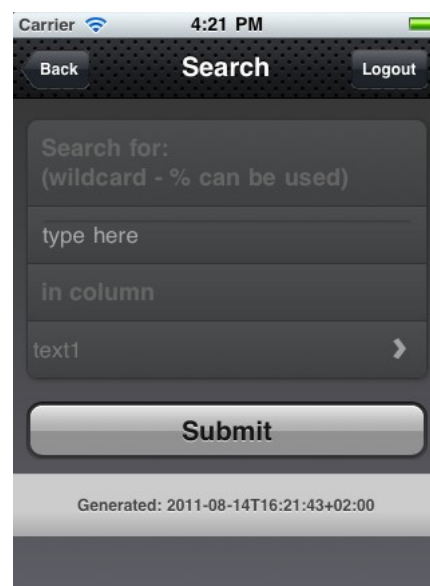
Formulář pro změnu záznamu v tabulce. Při tvorbě aplikací bývá zvykem, že formuláře pro vytváření a editaci objektů bývají stejné, jen v případě editace jsou jednotlivá pole vyplněná. V tomto případě je zvoleno rozdílné řešení. Důvodem je, že při editaci údajů se nedá použít atribut *placeholder* pro zobrazení informací o datovém typu sloupce. Tento údaj je ovšem pro uživatele důležitý a pokud uživatel zná požadovaný datový typ sloupce, ulehčí mu to práci a může tak předejít opakovanému vkládání dat. Nevýhodou je nutnost zobrazení dodatečných informací jako další položky a tedy roztažení celého formuláře.

10.5.9 Search

Základní vyhledávání v tabulce. Uživatel vyplní vyhledávaný výraz a vybere sloupec, ve kterém chce vyhledávat. Po odeslání požadavku se zobrazí seznam výsledků odpovídajících zadaným kritériím.



Obrázek 14: Formulář pro úpravu záznamu



Obrázek 15: Vyhledávání v tabulce

10.5.10 Problémy spojené s návrhem a implementací uživatelského rozhraní

Použití knihovny jQTouch usnadňuje vytváření uživatelského rozhraní s nativním vzhledem, ale jsou s ním spojené i problémy, které musí tvůrce aplikace řešit. V případě administračního rozhraní bylo potřeba vyřešit hlavně zpětnou navigaci pomocí tlačítka back umístěného v informačním řádku na každé stránce. Při konvenčním použití je na tlačítko navázána javascriptová funkce, která zajistí přechod na předchozí obrazovku s použitím stejné animace jako při vstupu. Takové řešení ale způsobuje, že pokud uživatel odešle nějaký formulář s daty a je přesměrován na jinou obrazovku, po stisknutí tlačítka back se zobrazí opět již jednou odeslaný formulář a data tak mohou být odeslána znovu, nebo tak může vidět záznam, který se již v databázi nenachází. Javascriptová historie si také pamatuje všechny navštívené stránky (stejně jako to dělají prohlížeče), což ovšem znamená, že než se uživatel dostane například na obrazovku s výběrem databází, musí projít zpětně přes všechny obrazovky, které navštívil. V řešené aplikaci jsem tlačítko back řešil rozdílným přístupem a to tak, že již nefunguje jako v prohlížeči, ale zprostředkovává pohyb po jednotlivých úrovních aplikace. Pokud se tedy uživatel nachází na přehledu záznamů v databázi a stiskne tlačítko back, přesune se na operace spojené s tabulkou bez ohledu na to, jestli předtím vkládal nebo mazal data. Z přehledu tabulek se dostane na přehled databází atd.

Dalším problémem spojeným spíše než s vlastním uživatelským rozhraním tak se samotnou vrstvou View bylo předávání uživatelských dat z formulářů. Nejjednodušším přístupem se zdálo použít název sloupce jako atribut *name* u položky formuláře. Odeslané parametry by tak byly ve formátu *název_sloupce => hodnota*. Databáze MySQL ale dovoluje ukládat názvy tabulek a sloupců v kódování UTF-8, které ale není povoleno v atributu *name*, s čímž počítá Ruby on Rails a pokud se použije takový název, je převeden do kódování ASCII a pokus o další práci končí chybovou hláškou. Tento problém jsem vyřešil pomocí skrytého pole s názvem sloupce. Z formuláře se tedy odesílají parametry ve formátu *key => název_sloupce, value => hodnota* a vzájemné spárování těchto údajů jsem řešeno přidáním indexu do atributu *name*.

10.6 Popis tříd a funkcí

Ve vrstvě Model jsou definovány třídy pro práci s tabulkami a daty v tabulkách pro přístup do databáze.

10.6.1 TableInfo

Slouží pro uchovávání informací o tabulce. Má následující atributy:

- **database:** String, jméno databáze, do které tabulka patří

- **name:** String, jméno tabulky
- **row_count:** Integer, počet řádků v tabulce
- **columns:** pole objektů typu ColumnInfo, informace o sloupcích tabulky
- **rows:** pole objektů, jednotlivé řádky tabulky
- **engine:** String, název úložiště
- **collation:** String, název porovnávání řetězců

10.6.2 ColumnInfo

Uchovává informace o sloupci v tabulce. Má následující atributy:

- **name:** String, jméno sloupce
- **type:** String, název datového typu
- **collation:** String, název porovnávání řetězců
- **nullable:** Boolean, možnost ukládat hodnoty NULL
- **key:** String, Textové vyjádření přítomného klíče, může nabývat hodnot „Primary key“, „Unique constraint“, „Multiple ocurences“ nebo může zůstat prázdný. Tento atribut se nastavuje při inicializaci instance aby se nemusela hodnota klíče ověřovat ve View.
- **def_value:** defaultní hodnota sloupce, datový typ hodnoty je určen typem sloupce,
- **auto_increment:** Boolean, automatické zvyšování hodnoty
- **pk:** Boolean, určuje zda je sloupec primárním klíčem
- **unique:** Boolean, určuje zda je na sloupci nastaveno omezení unikátní hodnoty
- **comment:** String, komentář ke sloupci

Ve třídě ColumnInfo jsou definovány tři metody, které se volají při inicializaci objektu a slouží pro nastavení atributů objektu.

- **nullable=(new_value):** nastavuje atribut nullable
- **key=(new_key):** nastavuje atributy key, pk a unique
- **auto_increment=(new_ai):** nastavuje atribut auto_increment

10.6.3 MobminDao

Třída MobminDao slouží pro přímou komunikaci s databází a pro naplnění instancí objektů TableInfo a ColumnInfo daty. Myšlenka použití samostatného objektu pro přístup k datům vychází z návrhového vzoru Data Access Object (DAO). Tento objekt, jedná se vlastně o jakousi mezivrstvu, by měl sloužit jako rozhraní, jehož implementace bude zajišťovat komunikaci s databází práci s daty. Výhodou takového přístupu je, že změny v této implementaci neovlivní ostatní části aplikace. Pokud bude potřeba například změnit konektor pro přístup do databáze, nebo přidat podporu pro další databázové servery, stačí tyto změny provést pouze ve třídě MobminDao. Tato třída se volá z controlleru a obsahuje následující atributy:

- **user**: String, uživatelské jméno pro přístup do databáze, atribut je přístupný pouze pro zápis
- **password**: String, heslo pro přístup do databáze, atribut je přístupný pouze pro zápis

a tyto metody:

- **initialize(user, password)**: konstruktor, nastavuje atributy user a password
- **connection**: pomocí ActiveRecord se připojí do databáze se zadaným uživatelským jménem a příjmením a vrátí toto připojení. Toto je místo, kde se nastavuje adaptér pro požadovanou databázi, umístění serveru a další atributy připojení.
- **clear_connection**: ukončí připojení do databáze.
- **show_databases**: připojí se k databázovému serveru a vrátí pole řetězců se jmény databází.
- **show_tables(database)**: připojí se k databázovému serveru a vrátí pole řetězců se jmény tabulek z databáze předané jako parametr funkce.
- **table_info(database_name, table_name)**: připojí se k databázovému serveru a naplní objekt TableInfo daty. U nově vzniklé instance TableInfo nechává prázdný atribut rows, nenahrává tedy všechna data (řádky) z požadované tabulky. Vrací instanci objektu TableInfo.
- **show_data(database_name, table_name, where, what)**: vrací instanci objektu TableInfo naplněnou daty. To dělá tak, že nejprve zavolá metodu table_info a poté z databázového serveru naplní atribut rows daty. Pokud mají parametry where a what hodnotu nil, nahrají se všechny záznamy z tabulky, jinak jsou výsledky omezeny hodnotami těchto atributů.
- **insert_data(data, database_name, table_name)**: ukládá nový řádek tabulky. Atribut data je pole ve formátu klíč => [jméno_sloupce, hodnota_k_uložení]

- **delete_row(row, database_name, table_name)**: maže řádek z databáze
- **update_row(data, database_name, table_name)**: upravuje řádek v databázi

Při práci s databází MySQL je třeba mít na paměti, že ne všechna úložiště podporují transakce a pokud má databáze zůstat v konzistentním stavu, je potřeba chyby ošetřit na straně aplikace. To je důležité zejména u editace řádků, protože pokud nastane problém při vkládání, resp. mazání řádků, databáze vypíše chybovou hlášku a operaci neprovede. Pokud ale nastane problém při úpravě řádku, může se stát, že data neodpovídají stavu před začátkem operace ani požadovanému stavu po ukončení operace. V aplikaci je to řešeno tak, že se upravovaná data uchovávají a pokud během operace UPDATE dojde k chybě, uloží se do databáze zpět původní data a chybová hláška se předá uživateli.

Pro aplikace je v řídicí vrstvě definován jediný controller. Jednotlivé metody dle konvencí RoR odpovídají url v adrese požadavku. Uvedu tedy jejich přehled a pouze stručný popis.

10.6.4 MobminController

- **index**: zobrazuje pohled s úvodní stránkou aplikace a přihlašovacím formulářem
- **show_databases**: zobrazuje pohled s přehledem databází
- **show_tables**: zobrazuje pohled s přehledem tabulek v databázi
- **table_info**: zobrazuje pohled s informacemi o tabulce
- **show_data**: zobrazuje pohled se všemi řádky v tabulce
- **insert_data**: zobrazuje pohled s formulářem pro vložení záznamu do databáze
- **save_data**: volá instanci MobminDao a předává data k uložení do databáze a poté přesměruje na pohled show_data
- **delete_row**: volá instanci MobminDao a předává data ke smazání. Poté přesměruje na pohled show_data
- **edit_data**: zobrazuje pohled s formulářem pro úpravu záznamu v databázi
- **update_data**: volá instanci MobminDao a předává data k úpravě. Poté přesměruje na pohled show_data
- **search**: zobrazuje pohled s formulářem pro vyhledávání
- **search_result**: předává instanci MobminDao parametry vyhledávání a zobrazuje nalezená data. Pro zobrazení používá pohled show_data

- **logout:** odhlásí uživatele z databáze

10.6.5 Specifická řešení

V konfiguraci aplikace je během inicializace RoR zakázáno nahrání komponenty ActiveRecord, přestože v aplikaci je tato komponenta používána. Framework RoR (potažmo jeho tvůrci) předpokládá, že vytvářená aplikace poběží nad databází, ve které budou uložena data relevantní k aplikaci (např. tabulky pro třídy ve vrstvě model) a očekává proto, že při inicializaci načte konfigurační soubor s přístupovými údaji k databázovému serveru a bude pracovat pouze s jednou databází, pod jedním uživatelským účtem. Tento přístup je ovšem u aplikace, kde předem není jasné nad jakou databází poběží nepoužitelný. Komponenta ActiveRecord je tedy použita až při inicializaci třídy MobminDao, která předává údaje o uživateli, který se chce k databázovému serveru připojit.

Aplikace také žádným způsobem neřeší uživatelská práva k přístupu do databáze. Pouze předává přihlašovací údaje a veškerá přístupová práva jsou řešena na úrovni databáze.

10.6.6 Vyhodnocení dle stanovených kritérií

Kritérii stanovenými pro vyhodnocení úspěšnosti nebo neúspěšnosti byli použitelnost, přizpůsobení uživatelského rozhraní pro dotykové ovládání, přehlednost, objem přenesených dat a rychlost vývoje.

Pro splnění podmínky použitelnosti byly stanoveny alespoň základní funkcionality: čtení, vytváření, úprava a mazání záznamů v databázích. Všechny tyto operace s daty lze pomocí výsledné aplikace provádět. Navíc lze zjistit informace o jednotlivých tabulkách a sloupcích. Obdobným způsobem by šlo řešit i správu tabulek a databází. Podmínka použitelnosti je tedy splněna.

Všechny ovládací prvky aplikace jsou dostatečně velké a odlišené od čistě informačních částí. Tlačítka pro navigaci a odhlášení jsou na všech obrazovkách stejně rozmístěná, uživatel tedy vždy jasně ví kde je nalezne. Podmínka přizpůsobení dotykovému ovládání je také splněna.

Pro zobrazení všech informací bylo zvoleno minimalistické uživatelské rozhraní, které se snaží být maximálně přehledné a srozumitelné. Pro údaje na všech obrazovkách byl zvolen stejný typ zobrazování pomocí oddělených seznamů, tak aby se uživatel vždy orientoval. Problémem mohou být obrazovky kde je zobrazeno příliš mnoho záznamů. Řešením by mohlo být použití stránkování. Celkově lze ale považovat podmínku přehlednosti za splněnou.

Použité technologie, využívající návrhového vzoru MVC a ajax pro načítání nových informací splňují požadavek na minimalizaci objemu dat a rychlost vývoje aplikace.

Všechny podmínky stanovené pro vyhodnocení výsledné aplikace jako úspěšné byly splněny a lze konstatovat, že je možné vytvořit administrační rozhraní přizpůsobené pro mobilní zařízení, které bude plnohodnotnou alternativou ke stávajícím řešením.

11. ZÁVĚR

Administrační rozhraní nad relační databází jsou široce využívané nástroje pro správu databází a pro přístup k datům. Mají své výhody i nevýhody. Výhodami jsou grafické uživatelské rozhraní a snadný přístup odkudkoliv. Nevýhodami jsou omezené možnosti jazyka HTML při vytváření ovládacích prvků a pomalejší odezva než u desktopových aplikací.

Mohutný rozmach mobilních technologií, zejména v oblasti hardware, umožňuje uživatelům využívat i aplikace a služby spojené hlavně se stolními počítači a notebooky i na mobilních zařízeních, kde to dříve nebylo možné. Přehled běžně používaných administračních rozhraní dokazuje, že se jejich tvůrci zatím přizpůsobením pro tyto zařízení nezabývají. Řešená aplikace přitom ukazuje, že lze využívat tento druh aplikace i na mobilních přístrojích.

Hlavní změnou při vývoji je především zcela rozdílný přístup při tvorbě uživatelského rozhraní. Tvůrci mobilních aplikací musí velice pečlivě přemýšlet nad tím, které informace budou zobrazovány a jakým způsobem. Jak přizpůsobit velikost a rozložení ovládacích prvků, tak aby byly snadno přístupné a jak provázat jednotlivé části aplikace a ulehčit uživateli navigaci.

Další oblastí kde je potřeba přizpůsobit vývoj aplikace potřebám mobilní komunikace je objem přenášených dat mezi serverem a klientem. Minimalizace množství dat lze dosáhnout třeba zvýšeným využitím ajaxových volání a přenášením pouze nutných dat místo celých stránek.

Vhodným výběrem používaných technologií lze pak dosáhnout požadovaných výsledků v přiměřeném čase.

12. CONCLUSION

Administrative interfaces for relational databases are widely used tools for database administration and for data access. They have their advantages as well as disadvantages. The advantages are their graphic user interface and easy access to data. The disadvantages are limited abilities of the HTML language at creating control elements and slower response than with desktop applications.

The big boom of mobile technologies, especially in HW area, allows their users to use applications and services connected mainly with desktop computers or laptops also with mobile devices which was not possible in the past. The outline of commonly used administrative interfaces proves that their authors have not been working on adjustments for these devices. The application solved shows that this kind of application can also be used for mobile devices.

The main change at the development is especially the completely new attitude towards the development of user interface. The authors of mobile applications must take into consideration which pieces of information will be displayed and which way. How to adjust the size and placement of control elements so that they would be easily accessed and how to connect particular parts of the application in order to make the navigation easier for the user.

Another area where it is necessary to adjust the development of the application to the needs of mobile communication is the amount of data transferred between the client and the server. To minimize the amount of data, increased use of ajax calling can be used as well as transferring only essential data instead of full pages.

The required results within adequate time can be reached by appropriate choice of used technologies.

13. SEZNAM POUŽITÉ LITERATURY

V seznamu literatury uveďte všechny prameny a literaturu ze které jste čerpali a kterou jste v práci citovali. V případě odborného časopisu uvádíme název časopisu kurzívou a uvádíme strany, na kterých se citovaný článek nachází. V případě knihy nebo učebnice uvádíme kurzívou název dané knihy. Pokud se jedná o jiný zdroj např. working paper, národní zprávu apod., neuvádíme kurzívou nic.

1. Troels Arvin, Comparison of different SQL implementations , 2011,
<http://troels.arvin.dk/db/rdbms/#functions>
2. phpMyAdmin development team, phpMphpMyAdmin 3.5.0. development documentation,
<http://www.phpmyadmin.net/documentation/>
3. phpMyAdmin development team, phpMyAdmin – translations, 2011,
http://www.phpmyadmin.net/home_page/translations.php
4. Vrána Jakub, phpMinAdmin, 2007, <http://php.vrana.cz/phpminadmin.php>
5. Vrána Jakub, Adminer – vlastnosti, <http://www.adminer.org/cs/#features>
6. Bruce Stewart, An Interview with the Creator of Ruby, 2001,
<http://linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html>
7. Hunt Andrew, Thomas David, The Pragmatic Programmer: From Journeyman to Master, 1999
8. Roy Thomas Fielding, Architectural Styles and the Design of Network-based Software Architectures, 2000
9. Rails documentation Team, Getting started with Rails, 2011,
http://guides.rubyonrails.org/getting_started.html
10. Gartner Group, MySQL – Market share, 2008,
<http://www.mysql.com/why-mysql/marketshare/>
11. Fulton Hal, *Ruby, kompendium znalostí pro začátečníky i profesionály*, 2009, Zoner Press
12. Oracle and/or its affiliates, MySQL 5.1 Reference manual, 2001

14. SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

Zkratka	Popisek
DBMS	Database management systém, systém řízení báze dat
UC	Use case, případ užití
RoR	Ruby on Rails
MVC	Model View Controller
DAO	Data acces object

15. SEZNAM OBRÁZKŮ

Obrázek 1: Textové rozhraní databáze MySQL.....	12
Obrázek 2: Grafické rozhraní programu SQL Developer společnosti Oracle.....	13
Obrázek 3: Úvodní obrazovka aplikace phpMyAdmin.....	17
Obrázek 4: Uživatelské rozhraní aplikace Adminer v české verzi.....	20
Obrázek 5: Uživatelské rozhraní programu phpPgAdmin.....	21
Obrázek 6: Přihlašovací obrazovka.....	37
Obrázek 7: Přehled dostupných databází.....	37
Obrázek 8: Přehled tabulek v databázi.....	38
Obrázek 9: Nabídka akcí po výběru databáze.....	38
Obrázek 10: Informace o tabulce.....	38
Obrázek 11: Informace o sloupcích.....	38
Obrázek 12: Přehled záznamů v databázi.....	39
Obrázek 13: Formulář pro vložení záznamu.....	39
Obrázek 14: Formulář pro úpravu záznamu.....	40
Obrázek 15: Vyhledávání v tabulce.....	40

16. SEZNAM TABULEK

Tabulka 1: Use case 1 - zobrazení dat v databázi.....	22
Tabulka 2: Use case 2 - vložení dat do databáze.....	23
Tabulka 3: Use case 3 - Vyhledávání dat v databázi.....	24
Tabulka 4: Use case 4 - Úprava dat v databázi.....	25
Tabulka 5: Use case 5 - Mazání dat z databáze.....	26
Tabulka 6: Přehled nastylovaných HTML elementů při použití jQTouch.....	32
Tabulka 7: Přehled animací jQTouch.....	32

17. SEZNAM PŘÍLOH

1. Příloha 1 – CD s aplikací

PŘÍLOHA 1 – CD S APLIKACÍ