ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra grafiky a interakce [DCGI]

Administrační nástroj pro databázové systémy (správa dat, vizuální editor struktury)

Administration tool for database systems (správa dat, vizuální editor struktury)

Bakalářská práce

Studijní program: Softwarové technologie a management

Studijní obor: Web a multimedia

Vedoucí práce: RNDr. Ondřej Žára

Kseniia Razina

Praha 2017

Čestné prohlášení

Prohlašuji, že jsem zadanou bakalářskou práci zpracovala sama s přispěním vedoucího práce a používala jsem literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé bakalářské práce nebo její části se souhlasem katedry.

V Praze dne 25.5.2017 Kseniia Razina

Poděkování

Děkuji RNDr. Ondřejovi Žárovi, vedoucímu bakalářské práce, za jeho cenné rady a připomínky, jimiž mi ochotně pomáhal. Dále děkuji své rodině za podporu při studiu a Petrovi Kostyukovi za jeho cenné rady a připomínky.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Kseniia Razina

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: Administrační nástroj pro databázové systémy (správa dat, vizuální editor struktury)

Pokyny pro vypracování:

Seznamte se s problematikou správy relačních databází a s frameworkem Nette pro psaní interaktivních webových stránek v jazyce PHP. Následně navrhněte a implementujte dvě komponenty pro administrační rozhraní relačních databází.
První komponentou bude nástroj pro správu dat, umožňující CRUD operace nad databázovými tabulkami. Pokud bude spravovaná relační databáze podporovat integritní omezení na úrovni vazby mezi tabulkami (Foreign keys), respektujte při úpravách dat povolené množiny hodnot těchto cizích klíčů.
Druhou komponentou bude vizuální editor databázové struktury, implementovaný jako webová single-page aplikace. Prozkoumejte dostupné klientské technologie pro tvorbu bohatých interaktivních aplikací v JavaScriptu a rozhodněte, které z nich jsou pro tento nástroj nejvhodnější. Soustřeďte se zejména na zlepšení uživatelského zážitku při práci s nástrojem, v konstrastu s programem phpMyAdmin, kterému je nízký uživatelský komfort dlouhodobě vytýkán.
Vzniklý nástroj kvalitativně porovnejte s existujícími řešeními (phpMyAdmin, Adminer) tak, aby bylo patrné, v čem je vámi implementovaný přístup pro uživatelský zážitek vhodnější. Pro potřeby tohoto porovnání otestujte vytvořený nástroj na uživatelích, kteří jsou zvyklí spravovat relační databáze. Nedílnou součástí práce bude dokumentace, včetně programátorské a automaticky generované.

Seznam odborné literatury:

- Michael Kofler: Mistrovství v MySQL 5, Computer Press 2007, ISBN 978-80-251-1502-2.
- Steven D. Nowicki, Ed Lecky-Thomson: PHP 6 programujeme profesionálně, Computer Press 2010, ISBN 978-80-251-3127-5.
- Nette Framework 2.4, https://doc.nette.org/cs/2.4/.

Vedoucí: RNDr. Ondřej Žára

Platnost zadání: do konce zimního semestru 2018/2019

L.S.

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 4.4.2017

Abstrakt:

Bakalářská práce řeší úlohu vytvoření administračního rozhraní relačních databází. Obsahuje dvě komponenty. První komponenta je nástroj pro správu dat, umožňující CRUD operace nad daty v tabulkách. Druhá komponenta je vizuální editor databázové struktury, implementovaný jako webová single-page aplikace. Zadaný úkol, který má zefektivnit a zpřehlednit administrativní práci s relačními databázemi, byl řešen pomocí technologií html, css, javascript, PHP, které autorka zkombinovala tak, aby výsledné řešení úkolu bylo nejvhodnější. Tento systém je porovnán s existujícími řešeními jako jsou phpMyAdmin, Adminer. Přínos této práce lze vidět zejména ve zefektivnění práce s více databázemi a zlepšení přehlednosti při vytváření databázových struktur.

Klíčová slova: databáze, databázová struktura, administrační rozhraní, grafický editor

Abstract:

This bachelor thesis deals with the task of creating a relational database administration interface. It contains two components. The first component is a data management tool that enables CRUD operations over data in database tables. The second component is a visual database structure editor implemented as a single-page web application. The specified task, which is to streamline and clarify the administrative work with relational databases, was solved using html, css, javascript, PHP, which combined the author to make the final solution work the most appropriate. This system is compared with existing solutions such as phpMyAdmin, Adminer. The benefit of this work can be seen especially in streamlining the work with multiple databases and improved clarity during the creation of database structures.

Key words: database, database structure, administration interface, graphical editor

Obsah

1. Úvod

Úkolem této bakalářské práce je vytvořit administrační nástroj pro databázové systémy, který bude umět spravovat data a také bude obsahovat vizuální editor struktury databáze. V současné době většina aplikací (a to jak menší tak ty velké) využívají databázové systémy. Nemusí se přitom jednat jen o webové aplikace, ale také o desktopové, mobilní, distribuované a jiné. Důležitější než software nebo hardware se stávají data a důležité není jen přístup k datům, ale i manipulace s nimi včetně práce s jejich strukturou.

Aplikace je napsána ve frameworku Nette s použitím jQuery a obsahuje dvě hlavní části. První část je zaměřena na vytvoření nástroje pro správu dat, umožňující CRUD operace nad daty v tabulkách a také vytvoření možnosti připojení více databází přes formulář. Ve druhé části je vytvořen vizuální editor databázové struktury, vypracovaný jako single-page aplikace. Samotné ukládání vzniklé struktury již není součástí této bakalářky, ale využívá komponentu pro ukládání změn ve struktuře do databáze, kterou vypracoval kolega Petro Kostyuk v rámci své bakalářské práce. Výše popsáné komponenty jsou součástí většího systému, jehož cílem bylo vytvořit komplexní administrační nástroj pro databázové systémy. A proto zdrojový kód, který je dodán společně s touto zprávou a obsahuje nejenom mnou vypracované komponenty, ale také další části, které nejsou součástí této bakalářské práce.

Bakalářská práce obsahuje následující části. V první části je popsána rešerše problematiky databází, nástrojů pro jejich správu a výběr vhodného programovacího jazyka a klientské technologie. Druhá část se zabývá konkrétními požadavky, analýzou a návrhem aplikace. Diagramy v této práci byly vytvořeny pomocí nástroje Enterprise Architect. V třetí části je popsána konečná implementace. Dané řešení bylo navrženo s ohledem na moderní OOP nároky, kvůli čitelnosti kódu a znovupoužitelnosti jednotlivých komponent. Aplikace je rychlá a má rychlou odezvu díky využití jQuery na klientské straně. V průběhu řešení práce bylo porovnáno řešení s existujícími řešeními jako jsou Adminer a phpMyAdmin. Závěr práce vyhodnocuje celkový způsob řešení této aplikace.

2. Rešerše aplikace

V analytické části se vymezují základní pojmy a použité technologie, které se použily během programování. Na začátku je čtenář seznámen s relačními databázemi a problematikou nástrojů pro jejich správu. V další části jsou uvedeny důvody pro výběr programovacího jazyka a frameworku, a taktéž je provedena analýza existujících klientských technologií pro tvorbu single-page aplikací v JavaScriptu.

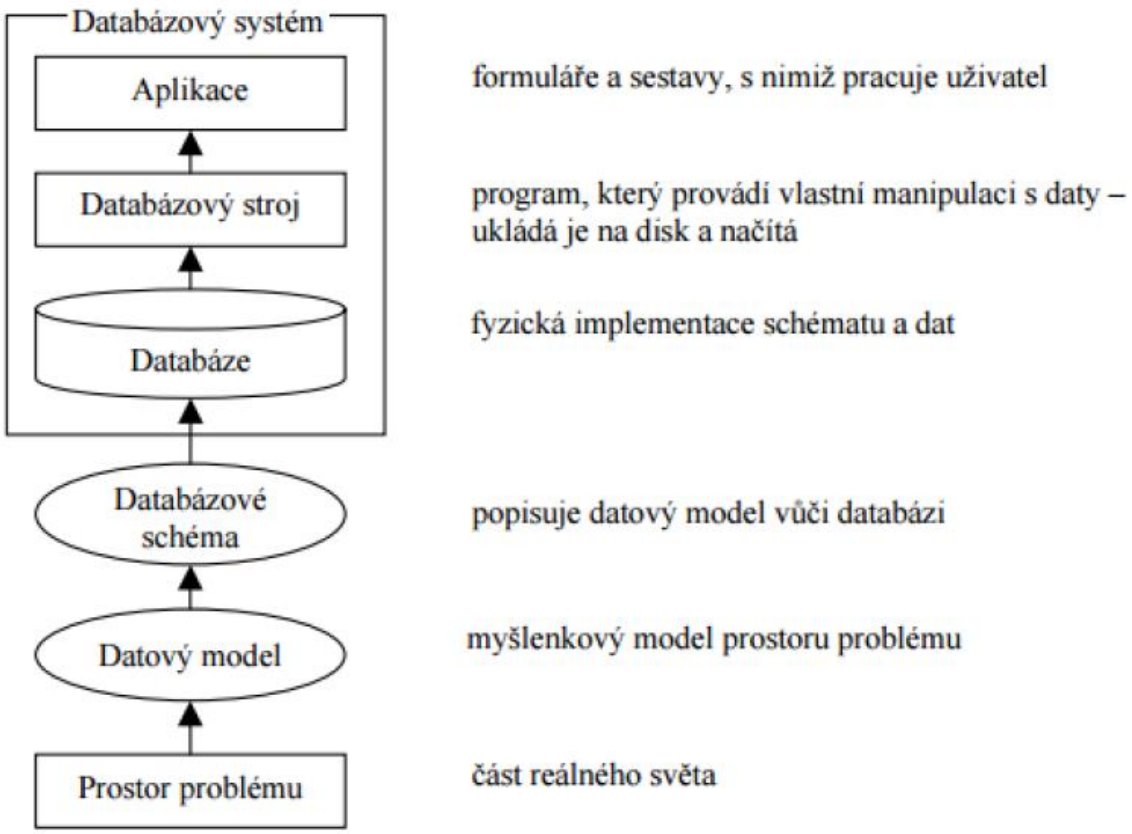
2.1 Relační databáze

Databáze představuje propracovaný systém souborů s pevnou strukturou pro ukládání dat.
Databáze může sloužit k:

- definici dat
- stanovení vztahů mezi ukládanými daty
- určení přístupu do databází
- realizaci operací nad daty

V dnešní době jsou relační databáze často používány, jejichž historie začala v roce 1970 po publikaci článku E. F. Codda o relačních databází. Relační databáze pohlížejí na data jako na tabulky a jsou charakterizovány tím, že jsou docela jednoduché a ukládají data ve vztazích. Jinými slovy relační databáze pracuje s daty za pomoci relačního modelu, totiž databázový systém je chápán jako množina relací.

Celou technologii relačních databází dobře ilustruje schéma uvedena na obr. č. 1. Z tohoto obrázku je vidět, že relační databáze se zabývají modelováním části reálného světa, tzv. prostoru problému. Z tohoto prostoru vytváříme datový model omezený na konkrétní množinu objektů a jejich vzájemných vztahů. Datový model pak převádíme do fyzické reprezentace, kterou následně implementujeme ve zvoleném systému.



Obr. č. 1: Terminologie relačních databází [5]

Tabulky v relačních databázích mají sadu vlastností. Nejdůležitější z nich jsou následující:

- tabulka nemůže mít dva stejné řádky
- každá tabulka by měla obsahovat informace o jednom typu objektu
- sloupce jsou uspořádány v určitém pořadí, které je stanoveno při vytvoření tabulky
- tabulka nemusí mít ani jeden řádek, ale musí mít alespoň jeden sloupec
- každý sloupec má jedinečný název a všechny hodnoty v jednom sloupci musí mít stejný typ (číslo, text, datum ...)
- v průsečíku každého řádku a sloupce může být pouze atomická hodnota, tj. hodnota, která se neskládá ze skupiny hodnot.

Takové tabulky tvoří základ struktury databáze. Pří návrhu relačních databází jsou používány striktně popsány datové struktury – schéma. Schéma databáze obsahuje popis obsahu, struktury a integritní omezení. Při návrhu tabulek je nezbytností vzít také v úvahu rozsah budoucích dat.

Podobné relační databáze se často využívají v aplikacích z důvodu své jednoduchosti a dobré jasnosti. Mezi neznámější zástupce relačních databázových systémů patří MySQL, PostgreSQL, Oracle a další. Pro menší webové aplikace se obvykle používají databáze MySQL ve spojení s PHP.

2.2 Problematika nástroje pro správu databáze

Problematikou správy databází se začali odborníci zabývat teprve relativně nedávno - s příchodem a rozvojem moderních databází. Existuje mnoho společností, poskytujících celou řadu multifunkčních aplikací pro usnadnění řízení, vývoje a správu. Jsou to například: MySQL Workbench, Navicat, PHPMyAdmin, HeidiSQL, Adminer a mnoho dalších. Tyto programy umožňují provádět správu serveru, spouštět SQL dotazy a zobrazovat obsah databází a tabulek. Některé z nich jsou určeny pro instalaci na počítači, jiné fungují prostřednictvím webového rozhraní. Každý program má svoje výhody a nevýhody, ale všechny poskytují funkce jako:

- přihlášení k definovanému serveru, pod uživatelským jménem a heslem
- běžné CRUD operace
- okno pro SQL dotazy
- výpis tabulek
- bezpečnostní opatření a další

Podobné aplikace se konstantně zlepšují, snaží se poskytnout co nejintuitivnější, nejjednodušší uživatelské rozhraní a dodat funkcionalitu, která před tím buď neexistovala, nebo v nějaké aplikací už byla evidována, ale ještě není dokonalá.

Aplikace, které pracují prostřednictvím webového rozhraní, jsou pro nás užitečnější než ostatní nástroje kvůli těmto důvodům:

- schopnost běžet přímo na serveru, což je vhodné pro webhosting, kdy je zakázán vzdálený přístup k databázi
- nepotřebují instalací, stačí zkopírovat do složky
- díky tomu, že nepotřebují instalace, nic nezapisují do registru a při deinstalaci stačí jenom odstranit složku

Pokud jde o funkčnosti administračních nástrojů, z mého úhlu pohledu a dle vlastního průzkumu je nejzásadnějším funkčním měřítkem reprezentace modelu databáze v grafické podobě a vizuální zobrazení vztahů mezi tabulkami. Podobné zobrazení informace umožňuje lepší pochopení databází a usnadňuje její vytvoření díky možnosti vidět nejen v tomto okamžiku vytvářenou tabulku, ale i celou strukturu.

Běžně dostupný program, který by řešil obdobnou problematiku - fungoval by prostřednictvím webového rozhraní, byl by bezplatný a zároveň by nabízel nejenom vizuální reprezentaci struktury databáze, ale zároveň by měl i možnost editace struktur bez okamžitého uložení na server, v současné době neexistuje (MySQL Workbench vizualizaci i editaci nabízí, ale je to standalone program, který požaduje instalaci). Právě z tohoto důvodu se tato práce převážně zaměřuje na vytvoření komponenty pro aplikaci pracující prostřednictvím webového rozhraní, která by umožňovala editaci během vizualizace databázové struktury.

2.3 Výběr vhodného programovacího jazyku a frameworku

V dnešní době existuje velké množství programovacích jazyků a různých frameworků, které je vylepšují a jsou efektivní. K nejznámějším jazykům patří C, C++, Java a PHP. V této bakalářské práci pro programování na straně serveru bylo rozhodnuto vybrat jazyk PHP.

PHP je především skriptovací jazyk, který je určen pro tvorbu dynamických webových stránek. Představuje hypertextový preprocesor, který přímo na serveru interpretuje HTML stránky a usnadňuje přistup k databázím. Je nezávislý na platformě, podporován u většiny webhostingových serverů a má velkou zásobu knihoven pro různorodé účely. PHP se dá docela rychle naučit a v současné době se většinou využívá ve webových aplikací.

Pro usnadnění práci také bylo rozhodnuto použít framework. Z PHP vznikla řada frameworků jako Zend, Nette, Smarty. Každý z těchto frameworků má své výhody a nevýhody, ale pro účely práce mi Nette framework přišel nejužitečnějším.

Nette Framework je open source framework pro tvorbu interaktivních webových aplikací, využívá čistý objektový návrh a je založen na použití komponent. Kromě toho používá událostmi řízené modelování, podporuje AJAX, DRY, KISS, MVC a vytvořený kód je znovupoužitelný.

Mezi nesporné výhody Nette patří následující vlastnosti:

- Nette disponuje sadou velmi silných ladících nástrojů pro odhalování a ošetřování případných chyb
- dává silný nástroj pro tvorbu formulářů a plnou moc nad jejich vzhledem
- používá technologii, která eliminuje výskyt bezpečnostních děr a jejich zneužití, jako je například XSS, CSRF, UTF-8 attack atd.
- obsahuje velmi silný validační jazyk
- podporuje automatický překlad a další funkce

Právě na základě těchto předností a existence aktivní komunity uživatelů v České Republice byl pro tento semestrální projekt vybrán Nette Framework.

Jelikož projekt má být rozšiřitelný pro možnost připojení k různým RDB (MySQL, PostgreSQL, SQlite a další), bylo pro připojení k databázím použito Doctrine DBAL. Doctrine DBAL je knihovna vyvinutá pro jazyk PHP a vytváří abstraktní vrstvu nad těmito RDB a umožňuje k nim přistupovat přes jednotné rozhraní.

2.4 Výběr klientské technologií pro tvorbu single-page aplikací v JavaScriptu

Nedůležitější část závěrečné práce byla potřeba vytvořit vizuální editor databázové struktury, vypracovaný jako single-page aplikace.

Pod single-page jsou chápané aplikace, u kterých je veškerá funkcionalita a logika umístěny

na jedné stránce. V rámci semestrálního projektu byly prozkoumány dostupné klientské technologie pro tvorbu single-page aplikací v JavaScriptu, zejména AngularJS, React a jQuery. Tyto technologie jsou stručně popsané dále:

1. AngularJS je javascriptový webový framework vyrobený od Google. Jeho cíl – rozšíření aplikací založených na vzoru MVC, jakož i zjednodušení jejich testování a vývoje. Jeho výhodami jsou:

- funkcionalita Two Way Data-Binding, která řeší synchronizaci stavů mezi modelem a pohledem
- implementace Dependency Injection, která řeší závislost mezi jednotlivými komponentami programu
- testovatelnost
- znovupoužitelnost komponent a další

2. React je javascriptová knihovna pro vytváření webových komponent vyrobená Facebookem. Největší výhoda této technologie je její oddělení od DOMu, který React vytváří sám. Struktura HTML je definována skládáním javascriptových funkcí.

3. jQuery je knihovna pro JavaScript s velmi jednoduchou syntaxí. Jeho výhodami jsou:

- podstatně menší množství kódu a snadnější na pochopení v porovnání s JavaScriptem
- podrobná dokumentace a aktivní komunita uživatelů, vždy připravená poskytnout pomoc
- jednoduché použití Ajax
- široké množství pluginů

Pro vizuální editor bylo rozhodnuto vybrat knihovnu jQuery, a to z toho důvodu, že AngularJS a React na rozdíl od jQuery "nutí" programovat tak, jak se předpokládá na základě jejich vytvořené struktury, což z našeho hlediska není úplně vhodně. Zatímco jQuery je jenom balíček nástrojů, který k ničemu nezavazuje a pouze umožňuje jeho využívání.

3. Sběr požadavků, analýza, návrh

Tato část bakalářské práci se zabývá přehledem projektu, konkrétními požadavky na funkčnost aplikaci, analýzou a návrhem programu.

3.1 Přehled projektu

Projekt bude sloužit pro účely administračního rozhraní relačních databází. Poskytne systém pro správu dat a rovněž umožní pohodlně zobrazení databázové struktury.

Hlavním cílem projektu bylo vytvořit webovou aplikaci a v rámci ní navrhnout a implementovat dvě komponenty pro administrační rozhraní relačních databází. Jedna poskytne komfortní systém pro správu dat a umožní CRUD operace a druhá představuje vizuální editor implementovaný jako webová single-page aplikace, který usnadní práci díky zobrazení dat ve vizuální podobě.

Tyto komponenty jsou součástí většího celku, jehož cílem je vytvořit komplexní administrační nástroj pro databázové systémy. Tato aplikace bude obsahovat všechny standardní funkcionality pro administraci databází, a kromě toho i následující komponenty: nástroj pro správu struktury a porovnávání databází, grafický editor struktury databáze a stromový pohled na data.

Vedlejším cílem projektu bylo nastudování základní problematiky správy relačních databází a seznámení s frameworkem Nette pro psaní interaktivních webových stránek v jazyce PHP. Analytická část této bakalářské práce byla základem pro implementační část aplikace.

Důraz byl kladen na jednoduchost použití tak, aby s aplikací pak dokázali pracovat i méně zkušení uživatelé počítačů.

K přínosům a výhodám aplikace patří:

- Možnost současného připojení více databází
- Přehledné zobrazení tabulek a možnost vidět jasnou databázovou strukturu
- Vizuálně představovány vztahy mezi tabulkami
- Schopnost programu běžet přímo na serveru, díky realizaci nástroje přes webovou aplikaci
- Systém umožňuje pro editaci struktury databáze použít přehledný grafický editor
- Progresivní přístup

Hlavním přínosem projektu je druhá komponenta aplikaci – vizuální editor fungující prostřednictvím webu. Podobný typ programu na trhu je nový.

Jelikož program je vytvářen pro využití, jako systém řízení báze dat, jeho zainteresované osoby jsou zejména programatoří.

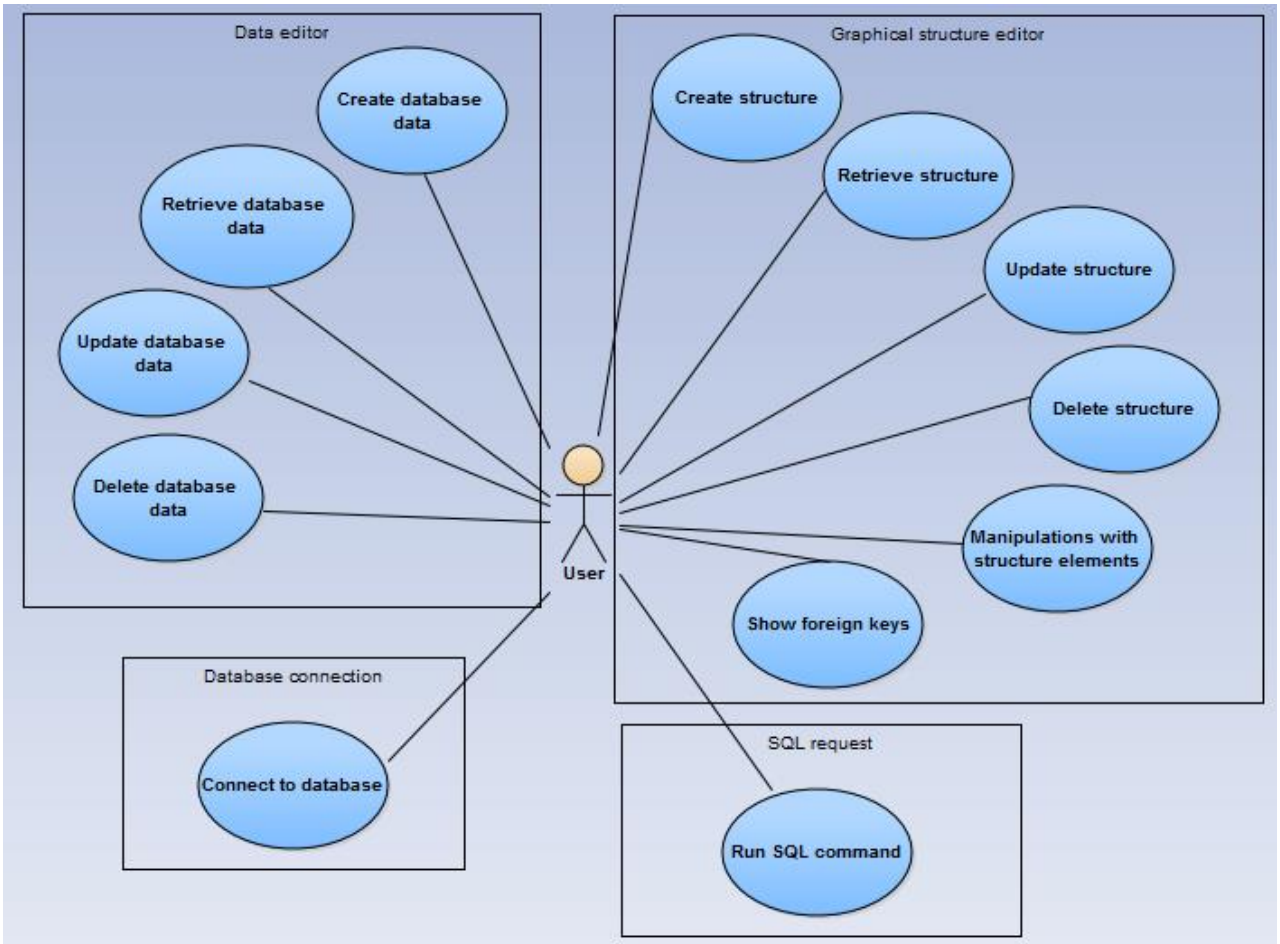
3.2 Uživatele a katalog požadavků

Protože pro použití aplikace není nutná ani registrace, ani vykonání administrační činnosti, systém bude podporovat jenom jeden typ uživatelů. Tento uživatel bude schopen bez jakéhokoli omezení využívat všechny funkce, které aplikace nabízí.

Katalog požadavků zachycuje požadavky na aplikaci. Požadavky se dělí na funkční a nefunkční. Ve funkčních požadavcích je popsané co daný systém bude umět. Nefunkční požadavky definují celý systém jako celek a popisují požadavky na hardware.

3.2.1 Funkční požadavky

Na obr. č. 2 jsou uvedeny funkční požadavky kladené na aplikace.



Obr. č. 2: Katalog požadavků

Data editor

- create database data - vytváří řádky tabulek
- retrieve database data - načítá řádky tabulek
- update database data - upravuje řádky tabulek
- delete database data - odstraňuje řádky tabulek

Graphical structure editor

- create structure - vytváří strukturu databáze, totiž přidává nové tabulky, sloupce
- retrieve structure - načítá tabulky přislušné databáze
- update structure - upravuje strukturu databáze, totiž jména tabulek a atributy sloupců
- delete structure - odstraňuje tabulky a sloupce
- manipulations with structure elements - manipuluje tabulky databáze
- show foreign keys - ukazuje cizí klíče

Database connection

- connect to database - připojuje se k databáze

SQL request

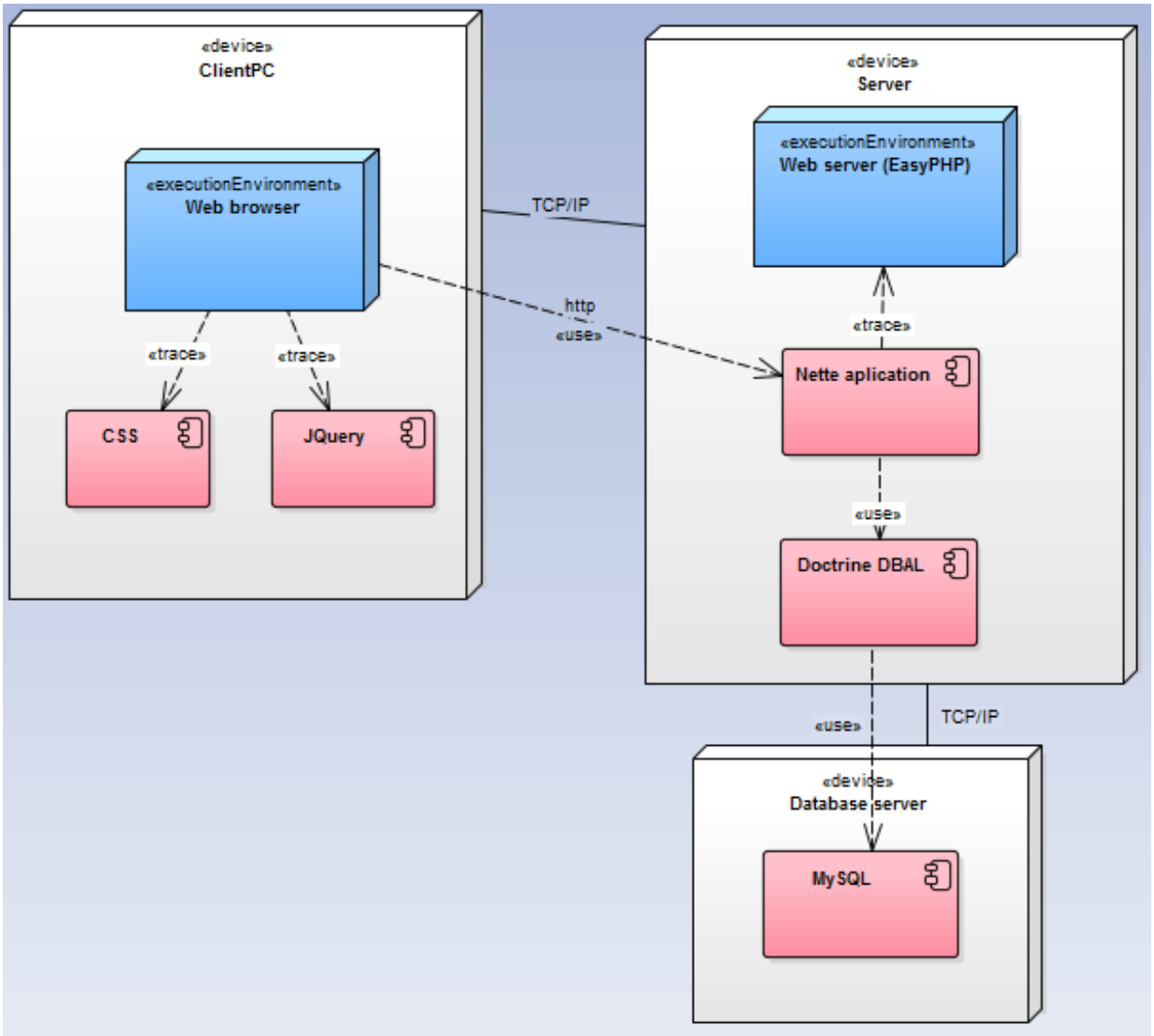
- run SQL command - spouští SQL dotaz

3.2.1 Nefunkční požadavky

- Grafické uživatelské rozhraní
- Program bude vyvíjen ve frameworku Nette (serverová část) a v jQuery (klientská část)
- Software bude kompatibilní s nejnovějsími verzemi prohlizečů Chrome a Firefox
- Bude podporováno uložení dat v DB a bude používat doctrine abstraktní vrstvu pro přístup do ní
- Systém by měl být intuitivní a uživatel by neměl mít problémy s užíváním aplikace
- K provozu aplikace musí být na straně serveru nainstalován PHP server

3.3 Diagram nasazení

Diagram nasazení patří mezi diagramy, které ukazují závislosti mezi rozmístění hardwarových komponent a procesy mezi nimi a softwarem. V diagramu se používají artefakty, uzly a vztahy mezi nimi. Na obr. č. 3 je znázorněn diagram nasazení pro administrační nástroj. V současné době se u většiny projektů rozděluje hardware pro serverovou aplikaci a databázi a na obr. č. 3 je vidět, že databáze je na jiném stroji než server nebo klient.



Obr. č. 3: Diagram nasazení

3.4 Grafický návrh aplikace

Grafický návrh aplikace je jedna z důležitých části programu. Většina takových návrhů se skládá z výběru barevné palety, tvarů objektů, jejich rozmístění a výběru fontu písma.

Podle typografických pravidel se rozlišují dva typy písma: patkové a bezpatkové. Patkové písmo obsahuje “patku” viz obr. č. 4 napravo. Patka spojuje text a používá se pro delší texty, aby čtenář text lépe četl. Příkladem je písmo “Times New Roman”.

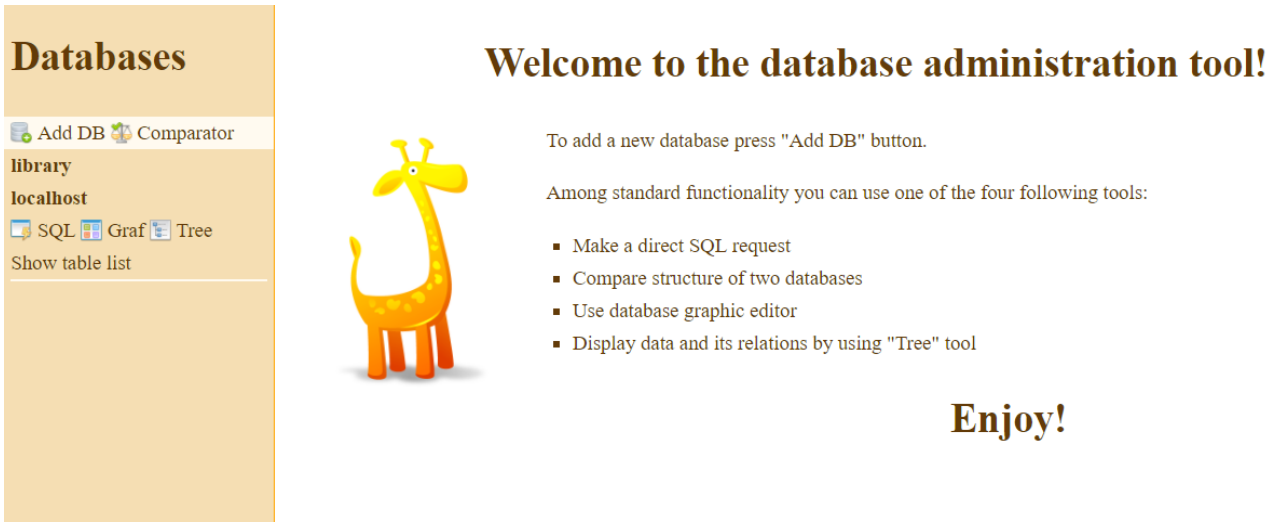
Ahoj Ahoj

Obr. č. 4: Bezpatkové písmo vlevo, patkové písmo napravo

Naopak bezpatkové písmo viz obrázek č. 4 vlevo. Příkladem je písmo Tahoma, Arial a používá se pro menší texty. Písma lze kombinovat, ale ve většině případů se nedoporučuje kombinovat více stylů písma. Působí to rušivě na uživatele. Co se týká této bakalářské práce, na front-end bylo využito patkové písmo.

Dále lze dělit písmo na proporcionální a neproporcionální. Proporcionální písmo má místo pro znaky podle šířky znaků. Neproporcionální písmo má pro všechny znaky stejné, příkladem je písmo Courier. Dnes se již používá akorát na zvýraznění třeba zdrojového kódu v textu. V minulosti tento typ písma byl spojen s psaním na psacím stroji.

Z pohledu barev byla nejdříve zvolena primární barva oranžová. Pak sekundární, která je na kruhu barev podobná, v tomto případě světle oranžová. Dále se dá použít komplementární barva, ale chtělo se docílit neutrálního barevného schématu, a proto na pozadí byla použita bílá barva. Pro písmo byla použita tmavě hnědá barva a černá barva, protože jsou kontrastní vůči ostatním barvám a dodržují barevné schéma. Na obr. č. 5 je vidět barevný návrh úvodní stránky aplikace.



Obr. č. 5: Úvodní stránka aplikace

4. Implementace

V této části jsou popsány principy fungování frameworku Nette, struktura projektu a konečná implementace aplikace.

4.1 Princip architektonického stylu frameworku Nette

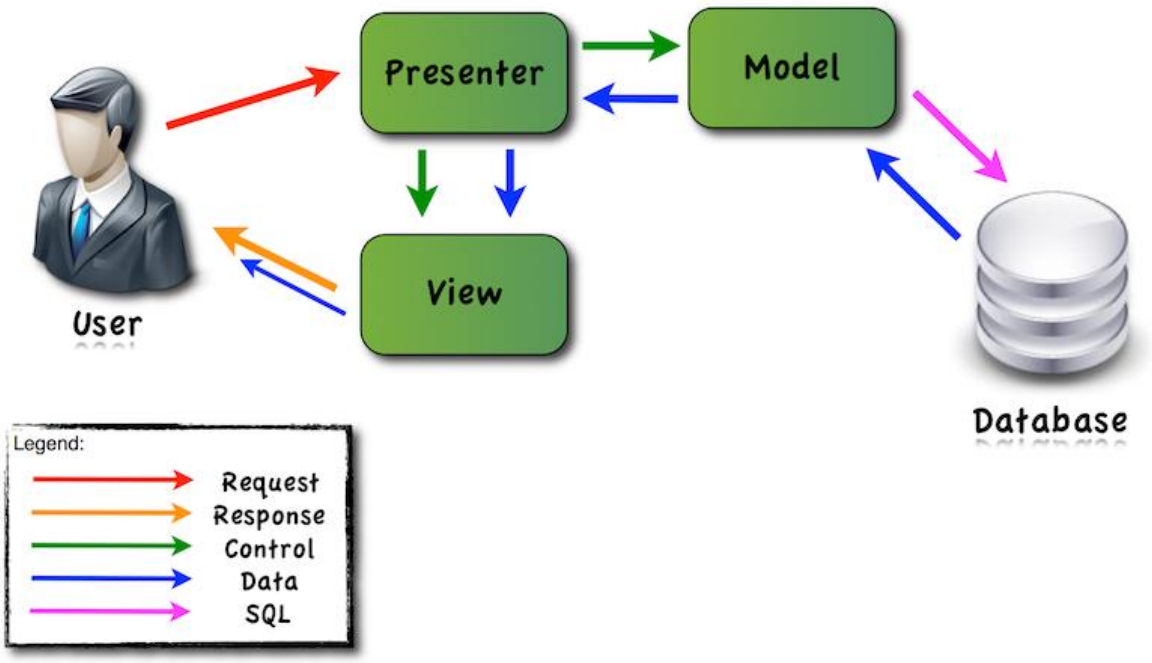
Jelikož Nette je klasický MVC framework, celá aplikace stojí na třech typech částí:

1. Presenter – část zabývající se řízením aplikace a poskytující funkce prostředníka mezi modelem a pohledem.

2. Model – část obsahující logiku aplikace. Zabývá se výpočty nebo prací s databází a přípravou dat pro předání presenteru.

3. Pohled (šablona) – část zabývající se prezentací získaných dat do vhodné podoby v prohlížeči. Nette framework používá šablonovací jazyk Latte, který do HTML šablon umožňuje vkládat data z PHP pomocí speciálních značek.

Obě komponenty závěrečné práce fungují za použití tohoto výše popsaného architektonickém vzoru, který je znázorněn na obr. č. 6

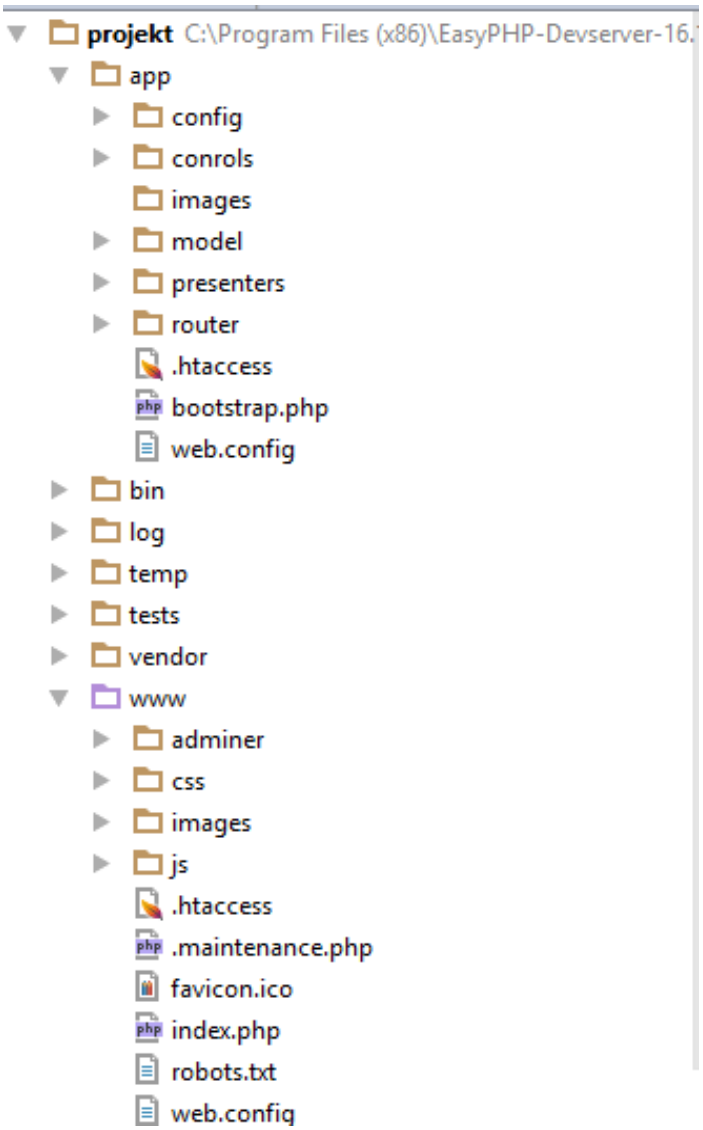


Obr. č. 6: MVP architektura frameworku Nette [3]

4.2 Struktura projektu

Struktura projektu obsahuje několik hlavních částí viz obr. č. 7:

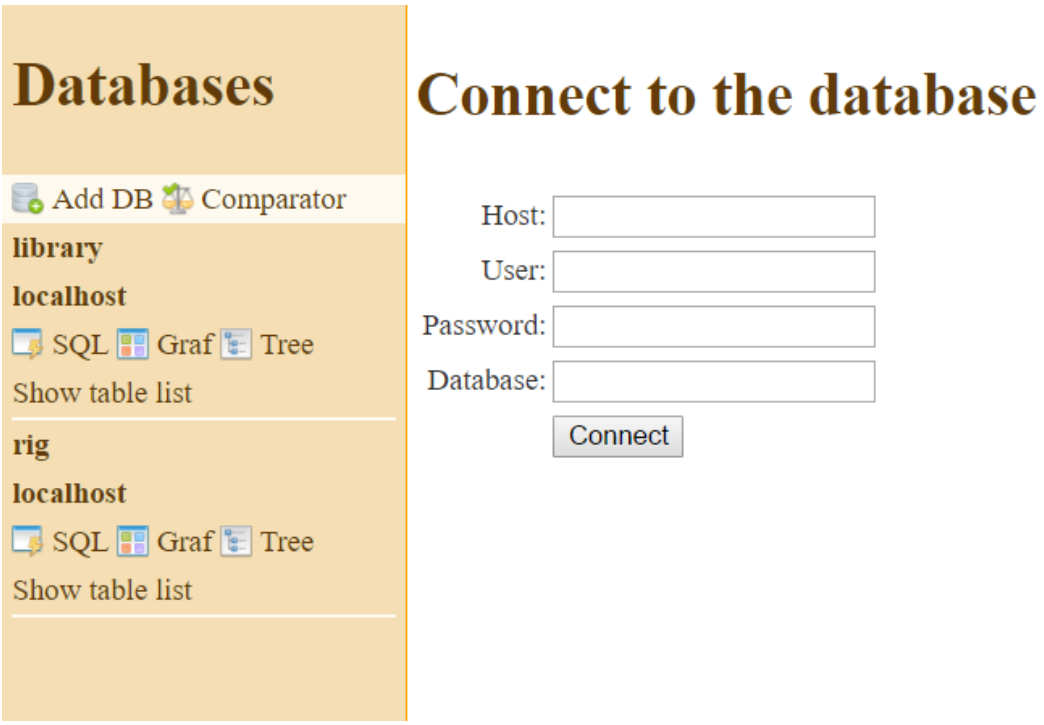
- app - adresář webové aplikace
- config - konfigurační soubory
- model - třídy modelů pro řízení logikou aplikace
- presenters - třídy presenterů pro řízení aplikací
- presenters/templates - šablony pro prezentaci získaných dat
- router - konfigurace URL adres
- bootstrap - spouštěcí soubor (zajistí načtení všech potřebných souborů a knihoven a předá řízení Nette aplikaci)
- log - chybové logy
- temp - místo pro dočasné soubory
- vendor - zde se nachází knihovna Nette
- www - tento adresář je přístupný z internetu. Mimo jiné obsahuje javascript a css soubory nezbytné pro normální běh aplikace a navíc v něm se nachází obrázky



Obr. č. 7: Struktura projektu

4.3 Implementace nástroje pro správu dat

První část bakalářské práce kromě vytvoření nástroje pro správu dat, umožňující základní operace pro práci s databází, je také zaměřena na vytvoření možnosti připojení více databází přes formulář (viz obr. č. 8).



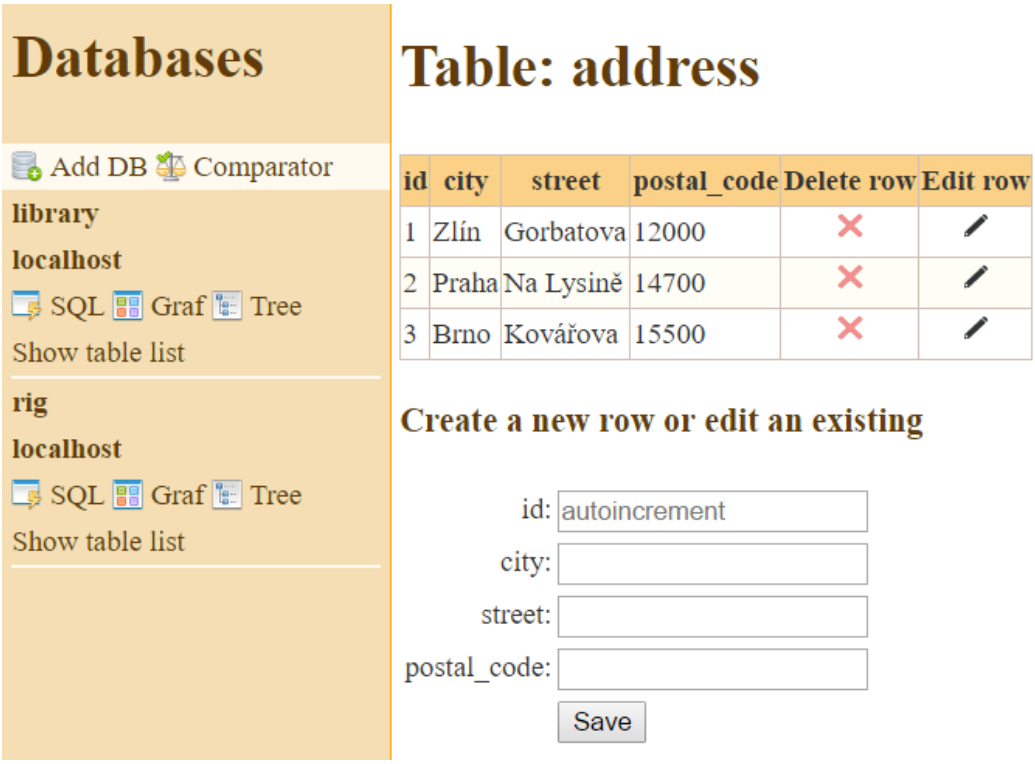
Obr. č. 8: Připojení databáze

Pro připojení k databázi se používá Doctrine DBAL. Vlastním připojením databází do session proměnné se zabývá metoda addParameters (viz obr. č. 9). Nejdříve ve formuláři pro nové připojení k databází vyplníme údaje o databázi (host, user, password, database name). Po odeslání tyto údaje dostane presenter DatabasePresenter. Ten zavolá metodu addParameters na DatabaseManager. Na základě předaných hodnot se pokusí vytvořit Doctrine připojení k databázi. Pokud se připojení podaří, uložíme předané parametry do session. Pokud ne, vyhodí se výjimka. Později je možné požádát DatabaseManager o získání připojení k databázi. Ten se podívá, jestli už bylo vytvořené připojení. Pokud ano, tak připojení vrátí, pokud ne, tak ho podle parametrů v session vytvoří a pak ho vrátí.

```php
public function addParameters($host, $user, $password, $database){
    if ($host == '' && $user == '' && $password == '' && $database == '') {
        throw new \Exception('');
    }

    $connectionParams = array(
        'dbname' => $database,
        'user' => $user,
        'password' => $password,
        'host' => $host,
        'driver' => 'pdo_mysql',
    );
```

Obr. č. 9: Implementace - připojení databáze

Nástroj pro správu dat, neboli první komponenta v rámci semestrálního projektu, umožňuje CRUD operace. Použití nástroje je znázorněno na obr. č.10. V případě CRUD operací se práce s serverem uskuteční pomocí odkazů na příslušné metody v prezenterech a také pomocí formulářů v prezenterech.



Obr. č. 10: Nástroj pro správu dat

Pro zpracování CRUD operací byl použit QueryBuilder viz obr. č.11.

```php
/**
 * Deletes the specific row from the table
 * @param $databaseIndex - index of current database
 * @param $tableName - name of the requested table
 * @param $where - deleting condition
 */
public function deleteRow($databaseIndex, $tableName, $where) {
    $conn = $this->dbm->getConnection($databaseIndex);

    $queryBuilder = $conn->createQueryBuilder();
    $queryBuilder->delete($tableName)->where($where);
    $queryBuilder->execute();
}
```

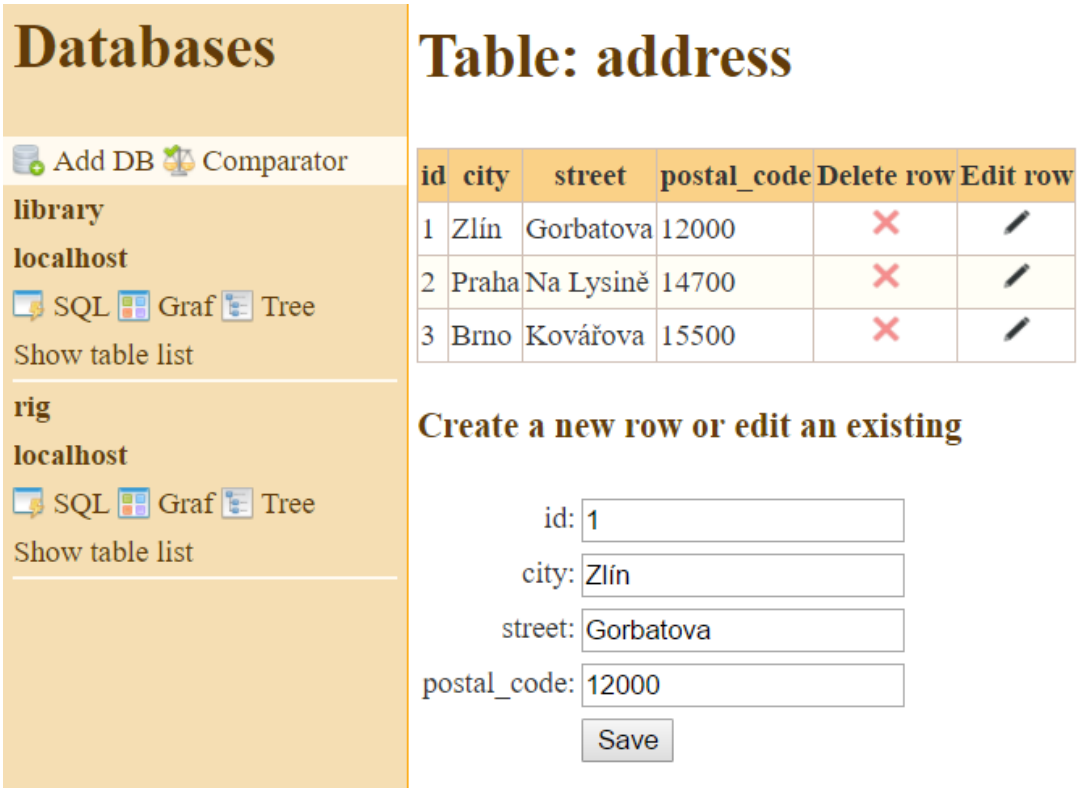
Obr. č. 11: Implementace - standardní SQL dotaz pro delete operaci

QueryBuilder je zvláštní třída, umožňující vytvoření databázového dotazu. Zavoláním createQueryBuilder() získáme instanci třídy QueryBuilder a pak postupným voláním dalších metod na této třídě skládáme požadovaný dotaz. Ukázka implementace QueryBuilder pro edit operaci je na obr. č.12 a ukázka efektu v UI je na obr. č.13.

```php
if ($where) {
    $queryBuilder->update($tableName);
    foreach ($columns as $column) {
        $columnName = $column->getName();
        $queryBuilder->set($columnName, ":$columnName") //
        ->setParameter($columnName, $values->$columnName);
    }
    $queryBuilder->where($where)
```

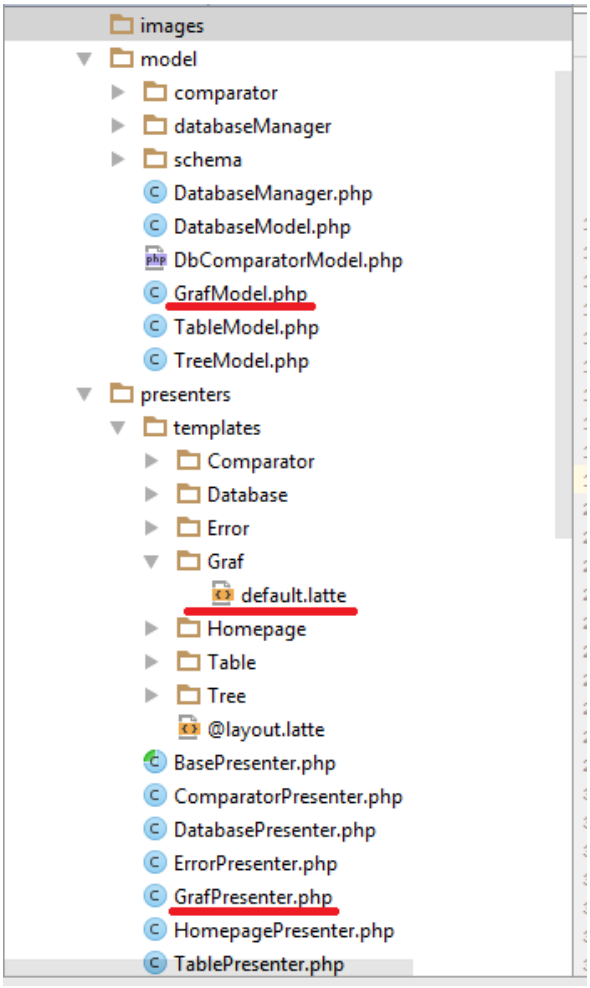
Obr. č. 12: Implementace - využití QueryBuilderu pro edit operaci

Kromě snadnosti použití a schopnosti se postarat o správný tvar konečného SQL, byl QueryBuilder také vybrán kvůli ochraně před útoky typu SQL Injection.



Obr. č. 13: Zpracování edit operace

Bez ohledu na typu zpracovávané operace jsou všechny dotazy předané prezenteru, který zavolá metodu modelu a výsledek je pak opět přes prezenter předán do šablony, tedy jde o použití MVC přístupu (viz obrázek č. 14).



Obr. č. 14: Použití MVC přístupu u souborů druhé komponenty aplikace

Důležitou částí první komponenty bakalářské práce bylo vytvoření možnosti vykonávat SQL dotazy. Knihovna Ace byla využita pro user-friendly efekt zvýraznění řádku v SQL dotazů. Ukázka použití je na obr. č.15 a ukázka efektu v UI je na obr. č.16.

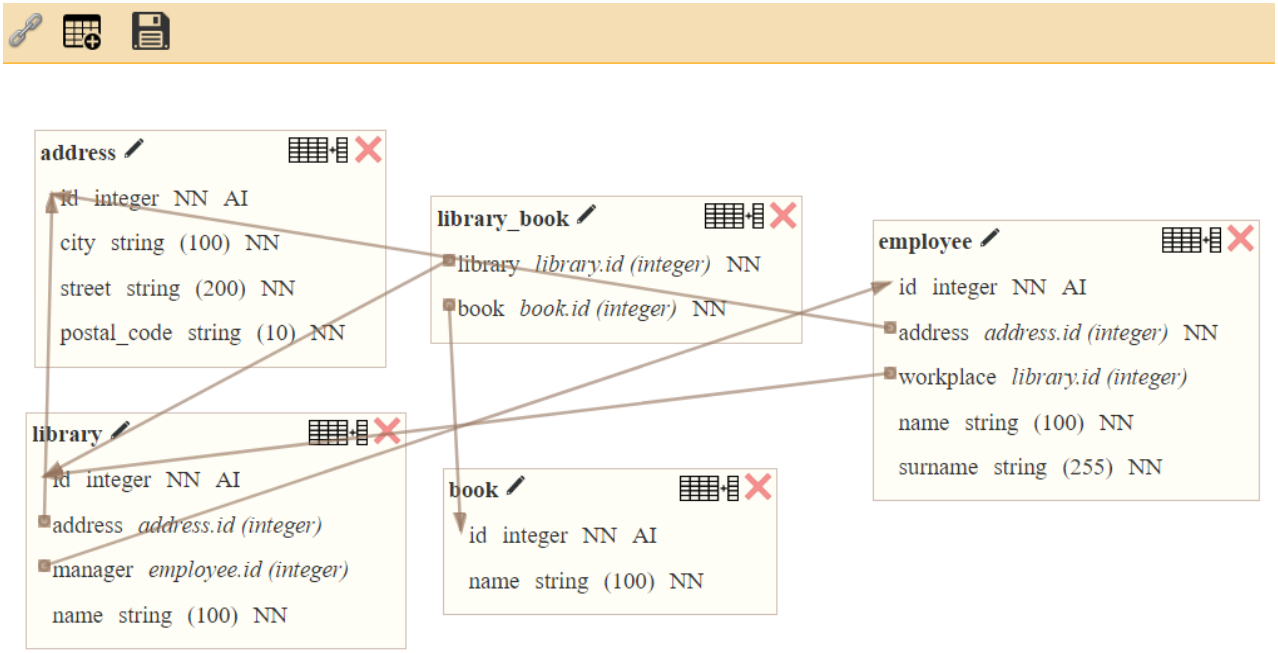
```
{block scripts}
    <script src="{$basePath}/js/ace-min/ace.js" type="text/javascript" charset="utf-8"></script>
    <script>
        window.editor = ace.edit("editor");
        window.editor.setTheme("ace/theme/solarized_light");
        window.editor.getSession().setMode("ace/mode/sql");
        window.editor.focus();

        $('.submit').click(function(){
            var data = window.editor.getValue();
            $('textarea').val(data);
            $('input[type=submit]').click();
        });

    </script>
{/block}
```

Obr. č. 15: Implementace - využití Ace knihovny pro user friendly interface

Obr. č. 16: Ukázka použití knihovny Ace

4.4 Implementace vizuálního editoru

V této části je popsána druhá komponenta bakalářské práce - vizuální editor databázové struktury. Ukázku vizuálního editoru je možné vidět na obr. č.17. Samotný program je navržen jako single-page aplikace. Pod single-page jsou chápané aplikace, u kterých je veškerá funkcionalita a logika umístěny na jedné stránce. Server se použije jako zdroj a úložiště dat a javascript se použije pro volání funkcí.

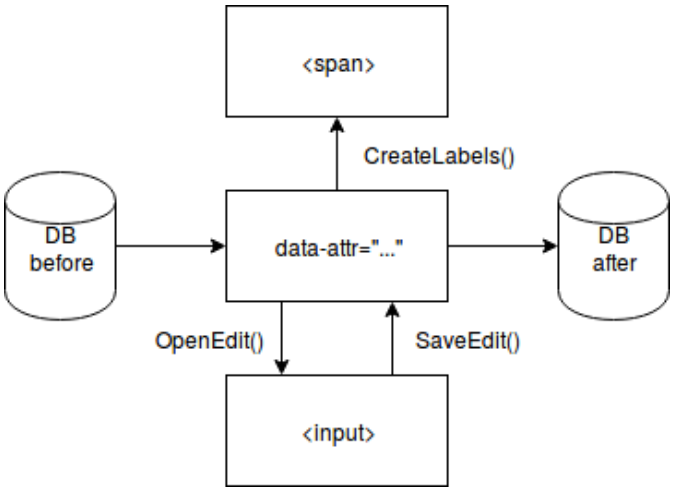


Obr. č. 17: Ukázka použití grafického editoru s cizími klíči

Funkcionalita vizuální editor funguje tak, že server nejdříve projde strukturu editované

databáze a na základě nich vygeneruje HTML stránku, na které je každá tabulka reprezentovaná obdélníkem, ve kterém jsou údaje o dané tabulce (název, sloupce, typy, ...).

S pomocí knihovny jQuery jsou vytvořené funkce, které umožňují uživateli přímo na této stránce údaje editovat, bez toho, aby docházelo ke komunikaci se serverem. Během editace může uživatel postupně doplňovat jakékoliv údaje, aniž by musely být úplné (například sloupec bez nadefinovaného typu). Díky tomu může vytvářet tabulky tak, jak je vymýšlí, aniž by ho zatěžovalo vyplňování všech formalit korektní tabulky. Další výhoda je, když uživatel zjistí, že chce udělat reroll a nechce uložit změny tak úpravy jen neuloží. Až bude hotový, stiskne uložit, a dojde k validaci dat na stránce, jejich následné serializaci a jejich odeslání na server. Pokud najde validace v datech nějaké neúplné údaje nebo chyby, uživatele na tyto chyby upozorní a on je bude muset opravit a znovu odeslat. Pokud validace projde v pořádku, dojde k uložení struktury do databáze.



Obr. č. 18: Ukázka úpravy hodnot

Proces úpravy hodnot během edit operace v grafickém editoru je znázorněn na obr. č.18. Z obrázku je vidět, že údaje o struktuře (název, typ, ...) se dočasně ukládají do atributů příslušného sloupce. Také se do atributů ukládají údaje o existenci cizích klíčů a informace do jaké tabulky a z jakého sloupce cizí klíč ukazuje. Celkem se úpravou hodnot zabývají tří funkce naprogramované pomocí knihovny jQuery:

- funkce attrToEdit načítá hodnoty z atributů a vkládá je do prvku input vybraného pro editaci sloupce
- funkce editToAttr vezme hodnotu z input prvku a zapíše jí do atributu
- funkce attrToSpan bere hodnoty z atributů a generuje hodnoty pro zobrazení uživateli

Ukázka implementace popsáné funkcionality na obr. č.18 je znázorněna na obr. č. 19.

```
/**
 * Takes values from the attributes and generates displayed values
 * @param column - JQuery column object for further processing
 */
function attrToSpan(column) {
    var valName = column.attr('data-name');
    var valType = column.attr('data-type');
    var valLength= column.attr('data-length');
    var valNull = column.attr('data-null');
    var valIncr = column.attr('data-incr');
    var isFk = column.attr('data-is-fk');
    var fkTable = column.attr('data-fk-table');
    var fkColumn = column.attr('data-fk-column');

    column.find('.view .name span').html(valName);
    column.find('.view .type span').html(valType);
    if (valLength != '0') column.find('.view .length span').html(valLength);
    column.find('.view .null span').html(valNull  == 'true' ? 'NN' : '');
    column.find('.view .incr span').html(valIncr == 'true' ? 'AI' : '');

    if (isFk == 'true') {
        var fkTableType = $('.desktop .table[data-id=' + fkTable +'] ' +
            '.column[data-id=' + fkColumn +']').attr('data-type');
        column.find('.view .type span').addClass('fK');
        column.find('.view .type span').html(fkTable + "." + fkColumn +
            ' (' + fkTableType + ')');
    } else {
        column.find('.view .type span').removeClass('fK');
        column.find('.view .type span').html(valType);
    }
}
```

Obr. č. 19: Implementace - zprácování edit operace

Na obr. č.20 je ukázka implementace vytváření cizích klíčů. Jedná se o jQuery kód, který znázorňuje směr cizího klíče. Začátek a konec čáry se dynamicky spočítají a jsou udělány pomocí SVG tagů marker. Linie se kreslí znovu při každém pohybu tabulkou, a proto se na začátku funkce vždy mažou. Informace o tom, zda tento sloupec má cizí klíč a kam má vést se bere z atributů. Jelikož čáry jsou rovné, neberou v úvahu umístění tabulek, a proto můžou překážet při práci se strukturou. Z tohoto důvodu se tato funkcionalita dá vypnout.

```
/**
 * Creates visible lines for showing all existing foreign keys
 */
function createLines() {
    var svg = $('.lines');
    $('svg.lines line').remove();
    svg.height($(document).height());
    svg.width($(document).width());
    $('.column[data-is-fk=true]').each(function () {
        var columnFrom = $(this);
        var columnToName = columnFrom.attr('data-fk-column');
        var tableToName = columnFrom.attr('data-fk-table');
        var tableTo = $('.table[data-id=' + tableToName + ']');
        var columnTo = tableTo.find('.column[data-id=' + columnToName + ']');

        var newLine = document.createElementNS('http://www.w3.org/2000/svg','line');
        newLine.setAttribute('x1',columnFrom.offset().left + 4);
        newLine.setAttribute('y1',columnFrom.offset().top + 10);
        newLine.setAttribute('x2',columnTo.offset().left + 4);
        newLine.setAttribute('y2',columnTo.offset().top + 10);
        newLine.setAttribute('marker-end', "url(#arrow)");
        newLine.setAttribute('marker-start', "url(#circle)");
        svg.append(newLine);
    })
}
```

Obr. č. 20: Implementace - vytváření cizích klíčů

Na obr. č.21 je vidět implementace, aby uživatel mohl obdélníky, které reprezentují tabulky, přesouvat libovolně. Využívá se k tomu funkce draggable z knihovny jQuery UI. Jak je z kódu vidět nejprve je potřeba zakrýt čáry reprezentující cizí klíče a poté je opět vykreslit, aby cizí klíče ukazovaly na správnou polohu.

```
/**
 * Inits the ability to move the tables
 * @param element - JQuery table object for further processing
 */
function initDraggable(element) {
    element.draggable({
        start: function() {
            deleteLines();
            $(".desktop").append($(this)); // inside desktop put this table
        },
        stop: function () {
            createLines();
        }
    });
}
```

Obr. č. 21: Implementace - aktivace draggable funkce u prvků

Na obr. č.22 je znázorněna validace dat. Data, která posíláme na server musí být správná. Pravidla validních dat:

- jméno tabulky musí být vyplněno
- tabulka musí mít aspoň jeden sloupec
- sloupec musí být pojmenován
- sloupec musí mít typ

Pokud tabulka nebo sloupec je označen atributem delete, tak ho funkce ignoruje. Validací ostatních dat se zabývá SQL server. Pokud se objeví chyba, Doctrine DBAL vyhodí výjimku a program ji ukáže, aby uživatel věděl, kde je problém.

```
function correctDataDuringSave() {
    var faults = [];

    $('.desktop .table:not(.deleted)').each(function() {
        var tableElement = $(this);
        var tableName = tableElement.attr('data-name');
        if (tableName == '' ) {
            faults.push('Table name is missing!');
        }
        if (tableElement.find('.column:not(.deleted)').length < 1) {
            faults.push('Table \'' + tableName + '\' must have at least one column!');
        }
        tableElement.find('.column:not(.deleted)').each(function () {
            var columnElement = $(this);
            var columnName = columnElement.attr('data-name');
            if (columnName == '' ) {
                faults.push('Column name in table \'' + tableName + '\' is missing!');
            }
            if (columnElement.attr('data-type') == '') {
                faults.push('Column type in table \'' + tableName +
                    '\' and column \'' + columnName + '\' is missing!');
            }
        })
    });
    if (faults.length == 0) {
        return true;
    } else {
        alert(faults.join('\n'));
        return false;
    }
}
```

Obr. č. 22: Implementace - validace dat

Než data pošleme do databáze tak je potřeba upravit jejich formu/strukturu. Grafický editor se postará o správné formátování dat, serializuje je do pole a posílá presenteru. Ten je připraví do struktury, ze které je možné vytvořit objekt třídy EditableSchemaDatabase. Práce s touto třídou a ukládání struktury do databáze již bylo součástí práce kolegy Kostyuka. Na obr. č. 23 je ukázka implementace formátování dat.

```
var arrayOfAllColumns = {}; // array of all columns

tableElement.find('.column .view').each(function(){
    var viewElement = $(this).closest('.column');
    var oldColumnName = viewElement.attr('data-id');

    var arrayOfOneColumn = {}; // array of one column, i.e. one name, type, etc
    arrayOfOneColumn['name'] = viewElement.attr('data-name');
    if (viewElement.attr('data-is-fk') == 'true') {
        var fkTable = viewElement.attr('data-fk-table');
        var fkColumn = viewElement.attr('data-fk-column');
        var fkTableType = $('.desktop .table[data-id=' + fkTable +'] ' +
            '.column[data-id=' + fkColumn +']').attr('data-type');
        arrayOfOneColumn['type'] = fkTableType;
    } else {
        arrayOfOneColumn['type'] = viewElement.attr('data-type');
    }
    arrayOfOneColumn['length'] = viewElement.attr('data-length');
    arrayOfOneColumn['notNull'] = viewElement.attr('data-null')=='true';
    arrayOfOneColumn['autoIncrement'] = viewElement.attr('data-incr')=='true';
    arrayOfOneColumn['new'] = viewElement.hasClass('new');
    arrayOfOneColumn['deleted'] = viewElement.hasClass('deleted');
    arrayOfOneColumn['identifier'] = viewElement.attr('data-id');
    // array of one column into array of all columns
    arrayOfAllColumns[oldColumnName] = arrayOfOneColumn;
});
```

Obr. č. 23: Implementace - preprocessing dat

5. Porovnání s Adminerem a phpMyAdmin

V kapitole 4 jsou popsány implementace hlavních částí systému. Výhody našeho řešení jsou uvedeny v tabulce č.1. Jak je vidět hlavními výhodami jsou funkcionality:

- grafický editor pro práci se strukturou
- uložení změn i nových dat do databáze až po potvrzení
- připojení více databází najednou

Tyto funkcionality ostatní řešení nemají.

Funkcionalita	**Naše řešení**	**Adminer**	**phpMyAdmin**
Přístup do db pomocí SQL dotazů uživatele	ANO	ANO	ANO
Grafický editor pro práci se strukturou (databáze, tabulky, atributy, cizí klíče)	ANO	NE	NE
Uložení změn i nových dat do db až po potvrzení	ANO	NE	NE
Možnost práci s daty	ANO	ANO	ANO
Připojení více databází najednou	ANO	NE	NE

Tab. 1: Srovnání softwaru s existujícími řešeními

6. Testování

Tabulka č.2 ukazuje hlavní funkcionality, které byly otestovány průchodem testera (tester testoval každou funkcionalitu). Také je v ní zaznamenán vstupy a výsledek testování.

Testovaná funkčnost	Vstup	Výsledek
Připojení k databázi	Host, uživatelské jméno, heslo a název databáze	Aplikace je připojena k databázi
Vykonání SQL dotazu	SQL dotaz	SQL dotaz proběhl
Vytvoření tabulky	Kliknutí na příslušné tlačítko	Tabulka je vytvořena
Úprava tabulky	Označení dané tabulky a zadání nového názvu	Název tabulky upraven
Smazání tabulky	Označení dané tabulky	Tabulka je označena pro smazání
Vytvoření sloupce	Označení dané tabulky a kliknutí na příslušné tlačítko	Sloupec je přidán
Úprava sloupce	Označení daného sloupce a změna jeho atributů	Atributy jsou upraveny
Smazání sloupce	Označení daného sloupce	Sloupec je označen pro smazání
Vytvoření řádku dat v tabulce	Zadání vstupních dat pro jednotlivé sloupce v tabulce	Řádek dat je vytvořen
Výpis dat z tabulky	Výběr dané tabulky v UI pro výpis	Data z tabulky jsou vypsané
Úprava řádku dat v tabulce	Označení daného řádku v tabulce a zadání nových dat	Řádek dat je upraven
Smazání řádku dat v tabulce	Označení daného řádku tabulky	Řádek tabulky je smazán
Uložení dat do databáze	Všechny změny provedené v grafickém editoru	Všechny změny jsou uložené do databáze

Tab. 2: Tabulka testů

7. Instalace

1. Rozbalte system.zip a jeho obsah nahrajte na webový server (např. EasyPHP Devserver). Server musí podporovat soubory .htaccess a musí mít povolený mod_rewrite.
2. Spusťte prohlížeč a otevřete v prohlížeči adresář www.

8. Závěr

Bakalářská práce byla vypracována na základě zadání katedry počítačové grafiky a interakce, fakulty elektrotechnické, Českého vysokého učení technického v Praze v rámci studijního programu Softwarové technologie a management, obor Web a multimédia.

Cílem této práce bylo vytvořit funkční administrační rozhraní relačních databází, které bude obsahovat komponentu na vytvoření možnosti připojení více databází přes formulář a správu dat, umožňující CRUD operace nad databázovými tabulkami a komponentu vizuální editor databázové struktury, implementovaný jako webová single-page aplikace.

Z podrobné analýzy projektu vyplynulo vhodné řešení, které nasměrovalo řešitele k využití MVP architektury, frameworku Nette a jQuery. V implementační části jsou vysvětleny postupy, které byly použity. Jsou zde ukázány ukázky kódů, z kterých je patrná implementace.

Každá funkcionalita v této práci byla otestována viz kapitola 6. Práce obsahuje porovnání s ostatními softwary, které řeší podobnou problematiku. Od ostatních řešení, která jsou na internetu k dispozici, má tento software funkcionality, které ostatní nemají: grafický editor pro práci se strukturou, uložení změn i nových dat do databáze až po potvrzení, připojení více databází najednou. Výhody těchto funkcionalit jsou zjevné jako zefektivnění práce vývojáře, který pracuje s více databázemi zároveň. Další výhody jsou lepší přehlednost při vytváření struktur v databázi a umožnění provádění všech změn na straně klienta bez toho, aby byla každá změna ihned propagována do databáze. Všechny změny jsou uloženy najednou až uživatel o uložení změn požádá.

9. Seznam zkratek

OOP	Object-oriented Programming (objektově orientované programování)
PHP	Personal Home Page (skriptovací programovací jazyk)
RDB	Relational Database
MYSQL	My Structured Query Language (systém pro řízení databází)
SQL	Structured Query Language
DBAL	Database Abstraction Layer
DOM	Document Object Model
MVC	Model View Controller
MVP	Model View Presenter
DRY	Don't Repeat Yourself
KISS	Keep It Simple, Stupid
SW	Software
HW	Hardware
DB	Databáze
CRUD	Create, Read, Update, Delete
AJAX	Asynchronous Javascript And Xml
GUI	Graphical User Interface
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface
URL	Uniform Resource Locator
UTF	UCS Transformation Format

10. Seznam tabulek a obrázků

V této kapitole je uveden seznam tabulek a obrázků použitých v textu bakalářské práce.

10.1 Tabulky

10.2 Obrázky

11. Obsah nahraných souborů

- pdf soubor s textovou částí bakalářské práce
- zip soubor, který obsahuje zabalený celý software
- doc soubor, což je text práce ve zdrojovém formátu
- html soubory s automaticky generovanou dokumentací PHP tříd
- html soubory s automaticky generovanou dokumentací javascript funkcí

Dokumentace je generovaná z celého projektu, nejen z části, která byla vypracována mnou. Obsahuje tedy jak mé třídy související s komponentami pro práci s daty tabulek, připojením více databází přes formulář, voláním přímých SQL dotazů a grafického editoru schématu databáze, tak i třídy související s klasickým editorem struktury, stromového procházení záznamů, porovnávání databází a ukládání změn ve struktuře databáze, které v rámci své bakalářské práce vypracoval Petro Kostyuk.

12. Použitá literatura

[1] Michael Kofler: Mistrovství v MySQL 5, Computer Press 2007, ISBN 978-80-251-1502-2

[2] Steven D. Nowicki, Ed Lecky-Thomson: PHP 6 programujeme profesionálně, Computer Press 2010, ISBN 978-80-251-3127-5

[3] Nette Framework 2.4 [online]. Dostupné z: https://doc.nette.org/cs/2.4/

[4] Základy relačních databází, jejich využití v programování webu [online]. [cit. 2017-01-11]. Dostupné z: http://gml.vse.cz/data/oppa-webdesign/zaklady-db.html

[5] Teorie databázových systémů [online]. [cit. 2017-01-11]. Dostupné z: http://physics.ujep.cz/~lmeduna/ujep_materialy/hromadne.pdf

[6] Úvod do databází [online]. [cit. 2017-01-11]. Dostupné z: http://ecdl.uzlabina.cz/e-learning/modul5.pdf

[7] Databáze nejsou jen MySQL [online]. 2002/01/05 [cit. 2017-01-09]. Dostupné z: https://www.interval.cz/clanky/databaze-nejsou-jen-mysql/

[8] Databáze [online]. [cit. 2017-01-09]. Dostupné z WWW: https://cs.wikipedia.org/wiki/Datab%C3%A1ze

[9] Nette Framework [online]. [cit. 2017-01-13]. Dostupné z WWW: https://cs.wikipedia.org/wiki/Nette_Framework

[10] AngularJS [online]. [cit. 2017-01-15]. Dostupné z: https://cs.wikipedia.org/wiki/AngularJS

[11] Mrozek Jakub. Začínáme s AngularJS [online]. 2012/11/30 [cit. 2017-01-12]. Dostupné z: https://www.zdrojak.cz/clanky/zaciname-s-angularjs/

[12] Prokop Petr. Single Page Application (SPA) [online]. 2015/09/11 [cit. 2017-01-12]. Dostupné z: https://rychlikoderi.cz/single-page-application-spa/