VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

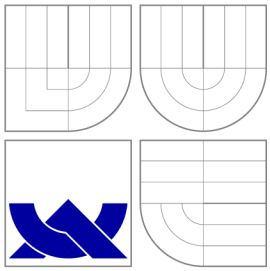
# SPRÁVA PROJEKTŮ STAVEBNÍ FIRMY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

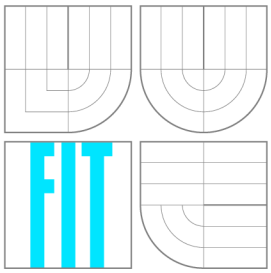
AUTOR PRÁCE
AUTHOR

MAROŠ BARJAK

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# SPRÁVA PROJEKTŮ STAVEBNÍ FIRMY

PROJECT MANAGEMENT OF BUILDING COMPANY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MAROŠ BARJAK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. OTA JIRÁK

BRNO 2011

# Abstrakt

Předmětem této práce je vytvoření informačního systému pro stavební firmu Hydroeco, s. r. o., umožňujícího zejména správu projektů, dokumentaci a vizualizaci projektových podkladů. V úvodu jsou popsané technické prostředky potřebné pro vytvoření a chod aplikace. Následuje popis zpracování požadavků zadavatele a vytvoření modelu systému. Ten je vystavěn na PHP Zend Frameworku, který využívá architekturu Model-View-Controller a objektově orientovaný přístup. Dále byly použity technologie jako HTML, CSS, JavaScript, jQuery, MySQL a pro vizualizaci knihovna JpGraph, Google Maps a nástroj Autodesk Freewheel. V druhé části je popsána samotná implementace, testování a návrh rozšíření systému.



# Abstract

This thesis describes an information system created for a building company Hydroeco Ltd. that enables especially project management, documentation and visualization of company projects. The introductory part contains description of technical resources needed to create and run the application. The next part describes the processing requirements and creation of a model system. The system itself is based on PHP Zend Framework using Model-View-Controller architecture and object-oriented approach. Furthermore, it uses technology such as HTML, CSS, JavaScript, jQuery and MySQL. Visualization is ensured by JpGraph library, Google Maps and Autodesk Freewheel tool. The last part describes the actual implementation, testing and a proposal for extension of the system.



# Klíčová slova

Informační systém, Web, Stavební firma, Modernizace, Vizualizace, Apache, HTML, CSS, JavaScript, jQuery, PHP 5, databáze, MySQL, UML, OOP, Zend Framework, MVC, Google Maps, JpGraph, Autodesk Freewheel

# Keywords

Information system, Web, Building company, Modernization, Visualization, Apache, HTML, CSS, JavaScript, jQuery, PHP 5, database, MySQL, UML, OOP, Zend Framework, MVC, Google Maps, JpGraph, Autodesk Freewheel

# Citace

Maroš Barjak: Správa projektů stavební firmy, bakalářská práce, Brno, FIT VUT v Brně, 2011

# Správa projektů stavební firmy

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Oty Jiráka. Další informace mi poskytl Ing. Ľubomír Zvada, jednatel firmy Hydroeco, s. r. o. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Maroš Barjak
13. května 2011

## Poděkování

V tejto časti by som sa chcel poďakovať vedúcemu práce Ing. Otovi Jirákovi za odborné vedenie a konzultáciu mojich dotazov. Poďakovanie patrí tiež pánovi Ing. Ľubomírovi Zvadovi za poskytnuté informačné schôdzky, telefonickú komunikáciu a možnosť vyvíjať a testovať systém s reálnymi dátami.

# Obsah

# Kapitola 1

# Úvod

Informačné technológie sú vysoko dynamickým, rýchlo rastúcim odvetvím našej spoločnosti. Čo sa používa a je aktuálne dnes, zajtra môže byť nepoužívané a neaktuálne. Informačné systémy dnes už nemajú podobu papierových kartoték či zoznamov. S technologickým pokrokom čoraz viac ľudí má záujem o systémy automatizované pomocou počítača. Je omnoho rýchlejšie, efektívnejšie a pohodlnejšie vyhľadávať, spracúvať či ukladať informácie v elektronickej podobe, ako pracovať s fyzickými archívmi. V komerčnej sfére čas predstavuje peniaze, teda efektivita a rýchlosť vykonanej práce je veľmi dôležitým aspektom. V súčasnosti existuje veľké množstvo technológií pre vytvorenie a chod takého systému. Medzi platenými nájdeme aj voľne dostupné, typickým príkladom je multiplatformový skriptovací jazyk PHP spolu s databázovým systémom MySQL.

Táto práca sa zaoberá návrhom a kompletnou realizáciou špecifického informačného systému pre firmu zaoberajúcu sa projektovaním, autorským dozorom a konzultačnou činnosťou v oblasti vodného hospodárstva a ekologických stavieb. Cieľom je nahradenie zastaralého systému uchovávania a spracúvania informácií, teda prechod na digitálnu kanceláriu. Novo vytvorený systém má priniesť efektívnejší spôsob manipulácie s dátami, dôraz na ich zálohu, možnosť vizualizácie projektových lokalít vo forme grafického náhľadu, vizualizáciu projektových modelov a firemných projektových rozpočtov vo forme grafov.

Úvodná časť práce popisuje teóriu informačných systémov, ich rozdelenie a využitie v spoločnosti. Kapitola tretia postupne predstavuje jednotlivé technológie potrebné pre realizáciu systému. Od serverovej časti a databázových technológií, cez jazyky určené pre tvorbu užívateľského rozhrania, až po nástroje pre vizualizáciu. V štvrtej kapitole je popísaná problematika špecifikácie požiadavkov zadávateľa, ich analýza a návrh samotnej aplikácie. Nasleduje popis implementácie jednotlivých častí, kde čitateľ bude oboznámený s chybami a problémami, ktoré sa často vyskytujú pri vývoji aplikácii tohoto typu. V kapitolách šesť a sedem je zhrnuté testovanie aplikácie nad reálnymi dátami, zhodnotenie dosiahnutých výsledkov a návrh možných rozšírení nad rámec súčasného zadania. Poslednou kapitolou je Záver, kde je obsiahnuté zhodnotenie práce, prínos pre autora a budúcnosť projektu.

# Kapitola 2

# Teória informačných systémov

Pokiaľ sa chceme zaoberať informačným systémom a správne chápať všetky súvislosti s ním spojené, musíme si v prvom rade ujasniť a vysvetliť tento pojem bližšie. Po jeho rozložení dostávame dve slová:

- *Informačný* – vychádzajúci z pojmu informácia
- *Systém* – obecne množina prvkov a väzieb medzi nimi

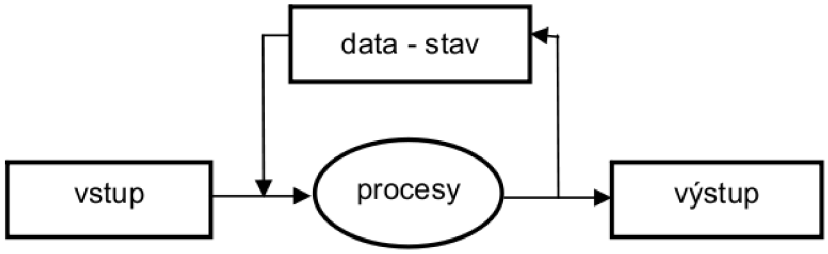
Pojem informácia je z filozofického hľadiska chápaný na veľmi vysokej úrovni abstrakcie. Obecne pod ním rozumieme údaj o reálnom prostredí, o jeho stave a procesoch v ňom prebiehajúcich. Informácia znižuje, alebo odstraňuje neurčitosť systému. Množstvo prijatej informácie predstavuje rozdiel medzi stavom neurčitosti systému pred a po jej prijatí. Chápeme ju ako vstupný tok do systému, pričom ale v oblasti informačných systémov existuje určitá hierarchia jej významu a interpretácie.

Na začiatku sa jedná o voľne rozšírené dáta bez významu. V akejkoľvek reprezentácii skutočnosti, schopné prenosu, uchovania či zpracovania. Po ich interpretácii užívateľom je im priradený význam (sémantika) a stávajú sa informáciou. Na rôznych miestach sveta je priradenie významu ku dátam odlišné (napr. v USA majú opačný zápis pozícií mesiacov a dní v dátume), preto tento proces má svoj význam. Informácie po ich zaradení do súvislostí nazývame znalosti. Získavanie znalostí z informácií a dát je špeciálnou disciplínou. Ide o oblasti ako dolovanie dát (*data mining*), OLAP, datové sklady, datové trhy apod.

Obecný systém sa skladá zo vstupnej a výstupnej časti, kadiaľ do systému vstupujú a vystupujú zdroje. Medzi vstupom a výstupom prebieha transformácia týchto zdrojov, prebiehajú tu procesy. Typickou častou systému je spätná väzba. Nie všetky výstupy procesov sú závislé len na okamžitých vstupoch, ale závisia aj na stave, v ktorom sa systém nachádza. Podľa prepojenia na okolie delíme systémy na *otvorené* – obsahujúce tok zdrojov a *uzavreté* – ktoré nie sú v interakcii s okolím. My sa budeme zaoberať systémom otvoreným.

Informačný systém má schéma podobné systému obecnému (viz obrázok 2.1), akurát transformáciu a spätnú väzbu môžeme pomenovať konkrétnejšie. Nepracuje autonómne, ale modeluje nejaký fyzický systém.

Formálne povedané, medzi informačným systémom a jeho fyzickým obrazom existuje vždy izomorfné zobrazenie. Všetky funkcie, ich parametre, zdroje a výsledky pôvodného systému majú svoje obrazy v systéme modelovanom a to platí aj naopak. Fyzické systémy pracujú so zdrojmi hmotnými, zatiaľ čo informačné systémy ako ich modely pracujú so zdrojmi nehmotnými, teda sú určitou abstrakciou svojho fyzického vzoru. Dôležité je, aby výsledky funkcií modelovaného a fyzického systému boli rovnaké, vtedy hovoríme, že informačný systém je platným modelom svojho vzoru. (*Voľne prevzaté z [14]*)




Obrázek 2.1: Schéma informačného systému [14]

## 2.1 História informačných systémov

Informačné systémy sú v neustálom technologickom vývoji. V minulosti prešli mnohými vývojovými štádiami avšak stále obsahujú zhodné súčasti a to dátovú vrstvu, kde je modelovaný stav systému a množina procesov prebiehajúcich nad týmito dátami a prezentačnú vrstvu, ktorá zprostredkováva užívateľský vstup a výstup.

Na počiatku vývoja v päťdesiatych rokoch tieto súčasti tvorili jeden programový celok čo predstavovalo nemožnosť ich zdielania a ďalšie neprijemnosti. Oddeliť dáta od procesov sa podarilo v šesťdesiatych rokoch, avšak ich popis zostal fyzicky v programoch a nebolo možné ich možné zdielať viacerými procesmi. Následne boli zavedené systémy pre zpracovanie súborov, kde dáta už mohli byť zpracovávané viacerými procesmi súčasne.

Pokrok prišiel v sedemdesiatych rokoch pri vytvorení systému riadenia báze dát (*ďalej* SRBD). Procesy už nemajú priamy prístup k dátam, ktoré sú uložené v databázi. Prístup k nim má na starosti SRDB. Je dosiahnutá nezávislosť definície procesov na dátach, tie sú udržiavané jednotne a ich štruktúry sú navrhované centrálne. Ich redundancia sa zmenšuje a zvyšuje sa konzistencia zavedením pojmu transakcia. Tá ma za úlohu udržať pri zmene stavu databáze atomickosť, konzistenciu, izolovanosť a trvanlivosť.

Ďalšou vývojovou etapou bolo oddelenie užívateľského rozhrania od aplikačných programov. Pri databázových technologiách sa používajú rôzne prístupy pre ukladanie dát. My budeme používať najpoužívanejší relačný model. Nasledujúca sekcia popisuje životný cyklus informačného systému.

## 2.2 Životný cyklus informačného systému

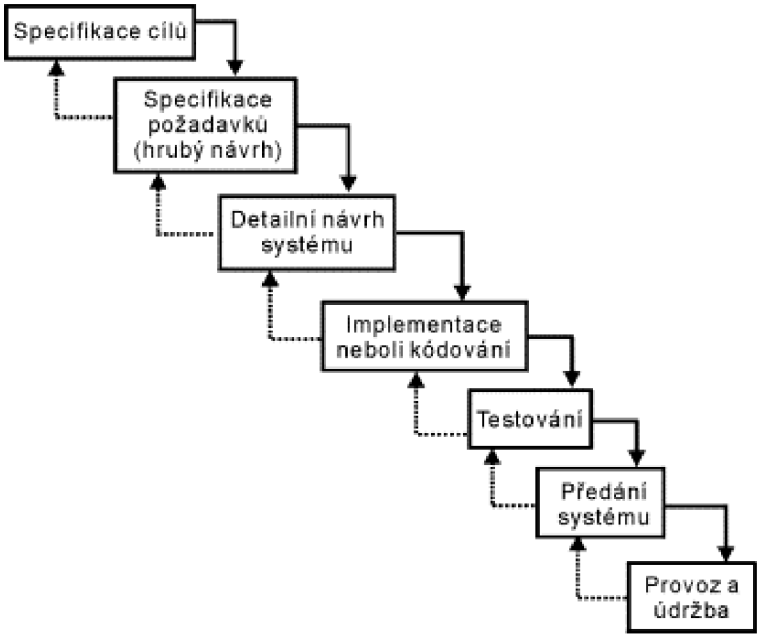
Životný cyklus charakterizuje zmeny, ktorými prechádza softvérový produkt počas svojej existencie. Životnosť informačného systému môžeme vymedziť ako okamžik vzniku požiadavku od zadávateľa až po okamžik konca jeho používania. Rozlíšiť môžeme dve základné obdobia a to obdobie postupného zavádzania a obodbie postupného vyradenia z činnosti.

Životný cyklus softvéru je reprezentáciou softvérového procesu a modelujeme ho pomocou modelu životného cyklu. Definuje fázy, kroky, aktivity, metódy, nástroje a očakávané výstupy softvérového projektu. Existuje niekoľko typov modelov ktoré sá zhodujú v základných fázach, ale líšia sa v ich význame. Medzi základné fázy patria:

- *Analýza požiadavkov a špecifikácia cieľov* – na základe komunikácie s klientom, slúži k pochopeniu účelu projektu a určeniu cieľov.
- *Návrh systému* – formalizácia získaných poznatkov do podoby modelu systému.
- *Implementácia* – vytváranie užívateľského rozhrania a modelovanie procesov v správnom programovacom jazyku.
- *Integrácia, nasadenie a testovanie systému* – zavedenie systému, testovanie s reálnymi dátami a oprava chýb.
- *Chod systému a jeho údržba* – nevyhnutná servisná činnosť.

Modely životného cyklu definujú jednotlivé fázy a interakcie. Opisujú čo sa má vykonávať, ale neopisujú ako sa to má vykonávať. Konkrétne implementácie sú unikátne pre každý projekt. Každý modelový typ má svoje výhody aj nevýhody.

**Vodopádový model** je historicky najstarším modelom, v minulosti úspešne používaný k riešeniu rady veľkých projektov. Odráža snahu softvérového inžinierstva o presadenie systematického vývoja. Etapy životného cyklu na seba nadväzujú a vzájomne sa nepretínajú. Vykonávajú sa postupne podľa presného plánu realizácie. Medzi fázami je možné doplniť, hlavne v praxi často využívanú spätnú väzbu. Tá existuje kvôli situácii, keď potrebujeme v nejakej fázy spraviť zmenu oproti dokumentu vytvorenému vo fázy predošlej. Tento model je zobrazený na obrázku 2.2.




Obrázek 2.2: Vodopádový model životného cyklu [28]

Výhodou modelu je, že pokiaľ sa nevyskytnú problémy je rýchly a lacný. Tento postup je vhodné uplatniť pri návrhu systému, kde je vopred známy problém a postup jeho riešenia. Medzi nevýhody zaraďujeme fakt, že tento model nie je obvykle používaný v rozsiahlych projektoch. Pri vypustený spätnej väzby môže nastať nasledovná situácia. Výsledný produkt predáme zákazníkovi a ten sa dožaduje ďalších zmien, ktoré si uvedomil až v tejto chvíli. Oprava chýb v takomto prípade býva drahá. Pri vývoji tohoto projektu bol použitý práve vodopádový model spolu so zakomponovanou spätnou väzbou. Postupovalo sa sekvenčne po jednotlivých fázach a kedykoľvek chcel klient niečo pridať alebo upraviť, bolo možné vrátiť sa na predošlú fázu a požadovanú zmenu vykonať.

Komplexnejší vývojový model predstavuje tzv. **prototypový model**, ktorý predpokladá zmeny požiadavkov zákazníka v priebehu vývoja a umožnuje reakcie na tieto zmeny. To predstavuje jeho najväčšiu výhodu. Na druhej strane je táto metóda u rozsiahlych systémov príliš náročná.

Medzi základné modely zaraďujeme aj **model spirála**, ktorý je kombináciou prototypového prístupu a analýzy rizík. Základom je neustále opakovanie vývojových krokov takým spôsobom, že v každom ďalšom kroku sa na už overenú časť systému nabalujú časti na vyššej úrovni. Výhodou je, že model analýzou rizík predchádza chybám a umožnuje priebežne konzultovať požiadavky klientov počas vývoja a modifikovať tak systém. Nevýhodou môže byť, že pri tejto metóde je vyžadovaná neustála spoluúčasť zákaznikov. To môže byť nevýhodné pre systémy vyvíjané na zakázku bez účasti budúcich užívateľov. Treba rátať aj s možnými problémami pri plánovaní termínu dodania a vystavení ceny za projekt.

Modelov životného cyklu existuje viacero typov, v tejto práci bol spomenutý len stručný základ pre pochopenie postupov vývoja softvéru. V nasledujúcej sekcii bude zhrnutá klasifikácia informačných systémov.

## 2.3 Klasifikácia informačných systémov

Problematika informačných systémov je rozsiahla, preto dochádza k deleniu podľa jednotlivých kritérií. Toto členenie môžeme uskutočniť nasledovne:

- **Podľa typu modelovaného fyzického systému** – členenie podľa typov zdrojov a oboru v ktorom sa systém využíva (napr. knižnica, účetníctvo, banka apod.).
- **Podľa veľkosti modelovaného fyzického systému** – veľkosť budovaného systému (veľkopodniky, stredné a malé firmy).
- **Podľa typu zpracovania** – faktografické zpracovanie využíva informácie, znalosti. Na druhej strane zpracovanie procesov sa používa pre riadenie technologických procesov, regulačné a meracie operácie.
- **Podľa režimu činnosti** – zpracovanie požiadavkov v reálnom čase (transakčné zpracovanie, technologické procesy) a dávkové zpracovanie dát ktoré je v súčasnosti na ústupe.
- **Podľa prevládajúceho typu ukládaných dát** – číselné a textové dáta (používa väčšina systémov), obrazové údaje (napr. geografické informačné systémy) a multimediálne informácie.
- **Podľa úrovňe rozhodovania** – v závislosti na určení systému pre rôzne úrovňe rozhodovania hovoríme o informačných systémoch pre transakčné spracovanie (OLTP –

*On Line Transaction Processing*) a informačných systémoch podporujúcich rozhodovanie (OLAP – *On Line Analytical Processing*). Pre OLTP systémy je charakteristické veľké množstvo štruktúrovaných, krátkych, atomických a izolovaných transakcií, teda automatizácia kancelárskeho zpracovania dát. Na druhej strane pri OLAP systémoch prevláda veľká intenzita komplexných dotazov, ktoré môžu pristupovať k miliónom záznamom.

Aby bolo možné spracovávať komplexnú analýzu a vizualizáciu, sú data modelované multidimenzionálne vo forme multidimenzionálnej kocky. Dimenzie sú hierarchické a poskytujú nám rôzne pohľady na uložené dáta. Uvedené dva prístupy nie sú jediné v klasifikácii informačných systémov podľa úrovňe rozhodovania, existuje celá hierarchia pre ktorú v tejto práci nie je vyčlenený potrebný priestor. Záujemcovia o hlbšie znalosti môžu využiť literatúru [14].

Informačný systém vytváraný v tejto práci môžeme klasifikovať ako ekonomicko-geografický, malého rozsahu pre firmu zamestnávajúcu 5 ľudí. Zpracovanie požiadavkov prebieha v reálnom čase, pracuje s faktografickými informáciami a ukladá číselné, textové i obrazové dáta. V literatúre ako [24] sú popísané pokročilejšie metodiky spojené s realizáciou informačného systému v podnikoch ako risk manažment, riadenie kvality, inkrementálny vývoj softvérového produktu apod. Dôležitým krokom pri vytváraní systému je analýza a použitie vhodných technológií pre vývoj. Táto problematika je popísaná v nasledujúcej kapitole.

# Kapitola 3

# Použité technológie

Zvolenie správnych technológií je závislé na firme pre ktorú systém vyvíjame. Veľké podniky majú pri výbere širšie možnosti, pretože majú k dispozícií väčší obnos finančných prostriedkov. Našou úlohou je teda zvoliť správne riešenie a našim klientom poradiť adekvátnu kombináciu technických prostriedkov vyhovujúcich pre ich potreby. Ideálnu platformu pre informačné systémy nám poskytuje internet.

Pre naše účely vývoja využijeme dvojicu PHP a MySQL, ktorá je na poli webových a serverových technológií klasickou kombináciou. Tieto technológie sú široko podporované a dostupné zdarma aj pre komerčné použitie. Jazyk PHP využijeme v spojení so Zend Frameworkom [2]. Frameworky obecne pomáhajú vývojárovi uľahčiť klasické implementačné problémy a zvyšujú efektivitu programovania. Programátor je nútený dodržiavať určité pravidlá a koncepcie (napr. návrhové vzory), ktoré vo výsledku prinesú kvalitnejší a prehľadnejší výsledok práce. Pri vývoji aplikácie bolo cieľom zvoliť podmienky pre vznik systému funkčného v prostredí firmy zadávateľa. Aplikácia bola vytváraná pod operačným systémom Windows. Ako vývojový server slúžil WAMP [1].

Z hľadiska užívateľského rozhrania boli použité osvedčené technológie HTML + CSS doplnené klientským skriptovacím jazykom JavaScript a knižnicou jQuery [20]. Medzi prostriedky podporujúce vizualizáciu boli zvolené Google Maps, knižnica JpGraph [5] a nástroj Autodesk FreeWheel [6].

## 3.1 Klientska časť – GUI

Grafické užívatelské rozhranie, tiež GUI (*Graphical User Interface*) je užívateľské rozhranie umožnujúce ovládať aplikácie, programy a samotný počítač pomocou interaktívnych grafických ovládacích prvkov. Pri webových aplikáciách primárne pozostáva z **dokumentu** zapísaného v nejakom značkovacom jazyku, **vizuálneho štýlu** vo forme CSS a **klientskeho skriptovacieho jazyka** umožnujúceho dynamicky modifikovať časti dokumentu (*JavaScript*).

### 3.1.1 XHTML

Celým názvom eXtensible HyperText Markup Language je značkovací jazyk pre tvorbu hypertextových dokumentov vychádzajúci z jazyka HTML. Jazyk HTML je vyvinutý z univerzálneho značkovacieho jazyka SGML (*Standart Generalized Markup Language*). Slúži pre vytváranie webových stránok na internete. Jeho vývoj súvisí s vývojom webových prehliadačov, ktoré spätne ovplyvnili jeho definíciu. Jazyk je charakterizovaný množinou značiek

(*tzv. tagov*) a ich atribútov definovaných pre danú verziu. Medzi tieto značky sú uzatvárané časti textu dokumentu a tým sa určuje ich význam (*sémantika*). Z hladiska významu rozoznávame tri základné skupiny značiek.

- **Štrukturálne značky** – rozvrhujú štruktúru dokumentu a dodávajú mu formu.
- **Popisné značky** – popisujú povahu obsahu elementu.
- **Štylistické značky** – určujú vzhľad elementu pri zobrazení.

Písať a upravovať (X)HTML kód môžeme v klasických textových editoroch, alebo WYSIWYG (*What you see is what you get*) editoroch, ktoré sú určené pre ľudí neznalých tohoto jazyka. Prehľad (X)HTML a jeho použitie je možné nájsť v [15].

### 3.1.2 CSS

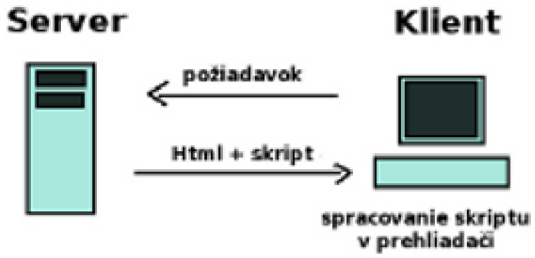
Kaskádovými štýlmi (*Cascading Style Sheet – CSS, [15]*) označujeme deklaratívny jazyk pre definíciu vzhľadu dokumentov zapísaných v značkovacích jazykoch, obvykle (X)HTML, alebo XML. Jeho vznik bol podmienený reakciou na požiadavky pre oddelenie obsahu od formy webových stránok. Tento prístup prináša najmä možnosť zmeny vzhľadu stránky bez zásahu do jej obsahu, väčšiu možnosť formátovania a rýchlejšie strojové spracovanie dokumentu.

Syntax jazyka je založená na deklarácii pravidiel zložených zo selektorov a blokov, ktoré určujú výsledný vzhľad danej množiny elementov. Výsledná aplikácia pravidiel závisí na priorite selektorov. Pod kaskádovými štýlmi si teda môžeme predstaviť nástroj, ktorý vytvorý vizuálny vzhľad pre náš informačný systém či inú webovú aplikáciu.

### 3.1.3 JavaScript a jQuery

JavaScript je interpretovaný skriptovací jazyk so základnou objektovo-orientovanou koncepciou. Ide o klientsky skript. Program je odosielaný spolu s HTML kódom do užívateľovho prehliadača a až tam je vykonaný. Jeho účelom je validácia vstupných prvkov rôznych formulárov, no je využívaný aj ako podpora užívateľského rozhrania.

Prináša do webových aplikácií interaktivitu a najrôznejšie funkcie. V súčasnosti na poli webových aplikácii je často používaný Ajax (*Asynchronous JavaScript and XML, [18]*), ktorý je schopný meniť obsah internetových stránok bez nutnosti ich znovunačítania. JavaScript je často zamieňaný a skloňovaný spolu s jazykom Java, pričom sa jedná o dva odlišné samostatné programovacie jazyky. Na nasledujúcom obrázku je zobrazený princíp JavaScriptu.




Obrázek 3.1: Princíp JavaScriptu [15]

Nad JavaScriptom je vystavaných niekoľko frameworkov podporujúcich Ajax a efekty uživateľského rozhrania. Najpoužívanejším je jQuery [20], ktorý bol využitý v tejto práci. Medzi základné funkcie, ktoré nám jQuery umožnuje patria:

- *prístup k jednotlivým častiam stránky*,
- *zmena vzhľadu stránky*,
- *možnosť meniť a rozširovať obsah stránky*,
- *reakcia na užívateľove úkony*,
- *animácia a efekty*,
- *získanie informácii zo serveru bez nutnosti načítania celej stránky*,
- *výrazné zjednodušenie bežných javascriptových úkonov.*

Medzi ďalšie JavaScriptové frameworky patria napr. [4]. Zaujímavé skripty použiteľné pre webové aplikácie je možné nájsť v [15, 25].

## 3.2 Serverová časť

Riadiaca časť aplikácie fungujúca na webovom serveri má za úlohu generovať užívateľské rozhranie na základe stavu aplikácie a požiadavkov užívateľa, vykonávať aplikačnú logiku a poskytovať úložisko dát vo forme databáze.

### 3.2.1 Databázový systém MySQL

Predstavuje multiplatformový relačný open source databázový systém [19], ktorý je založený na štrukturovanom dotazovacom jazyku SQL (*Standart Query Language, [13]*). Napriek tomu, že MySQL je voľne dostupný pre komerčné účely, môžeme ho pokojne označiť ako životaschopného konkurenta drahých technológií, akými sú Oracle, alebo Microsoft SQL Server. Je vysokovýkonný, prenostiteľný, spoľahlivý a vyžaduje minimálne náklady pre fungovanie.

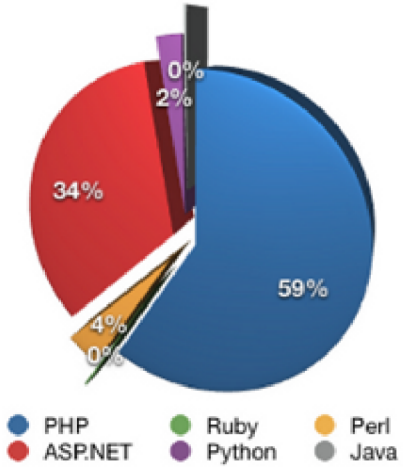
Pojem relačná databáza si môžeme predstaviť ako kolekciu vzájomne poprepájaných dát uložených v podobe textu, čísel, alebo binárnych súborov. Existuje mnoho typov databáz počínajúc jednoduchými, ktoré skladujú ploché súbory, cez relačné až po objektovo-orientované databázy. Napriek faktu, že relačné databáze vyžadujú pri návrhu a implementácii viac úsilia ako je tomu pri databázach bežných, poskytujú vo výsledku väčšiu spoľahlivosť a integritu dát. Okrem toho tiež ponúkajú rozšírené možnosti vyhľadávania a možnosť zdielaného prístupu k dátam.

Vďaka začleneniu databáze do webových aplikácii a možnosti kooperácie s jazykom PHP vykazuje táto kombinácia výraznú flexibilitu. V projekte je využité MySQL verzie 5.1.36 ako súčasť WAMP-u. Bližší popis a samotné funkcie jazyka doplnené príkladmi je možné nájsť v [17].

### 3.2.2 PHP

PHP (*Hypertex Preprocessor, [23]*) je platformovo nezávislý skriptovací jazyk určený primárne pre tvorbu dynamických webových stránok. Je navrhnutý tak, aby vykonal určitú činnosť ako reakciu na výskyt nejakej udalosti. Napríklad užívateľ odošle vyplnený formulár, ktorý je na strane serveru spracovaný a užívateľovy je interpretovaný výsledok. Medzi prednosti jazyka PHP patrí výkon, tesná integrácia s dostupným databázovým systémom, stabilita, prenositelnosť a takmer neobmedzené možnosti rozširovania. Všetky uvedené výhody sú zadarmo, kedže jazyk je voľne šíriteľný. Práve z tohoto dôvodu nastal prudký nárast jeho používania.

Nasledujúci obrázok zobrazuje štatistiky použitia 6 najznámejších technológií pre tvorbu webu vzťahujúci sa k roku 2010. Pomerne prekvapivý je nárast používania jazyka ASP.NET od spoločnosti Microsoft. Zber štatistík prebehol formou procesu neustálych požiadavkov na tri typy hlavičiek – X-powered, Server a Cookie. Preskúmaných bolo 6.7 miliónov domén, na ktorých bolo možné identifikovať použitú technológiu. Pre uloženie 10.8 miliónov hlavičiek bolo potrebných okolo 4.9 gigabajtov.




Obrázek 3.2: Použitie webových technológií v roku 2010 podľa [3]

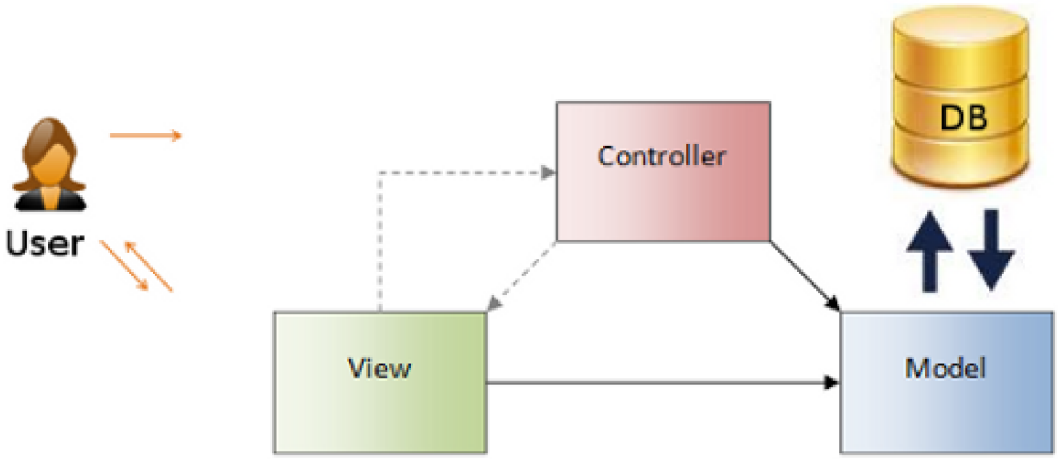
### 3.2.3 Zend Framework

Vývoj v čistom PHP kóde nie je zrovna najštastnejšou volbou, pretože prináša radu nevýhod ako napr. zlú údržbu aplikácie, rozšíritelnosť apod. HTML elementy sú zlúčené s PHP skriptami a výsledok tvorí neprehľadný veľký kus kódu. Z toho dôvodu je vhodnejšie využiť nejaký z dostupných PHP frameworkov. Porovnanie rýchlosti a niektorých ďalších atribútov najpoužívanejších frameworkov je možné nájsť na [9].

Pre túto prácu bol vybraný Zend Framework [2]. Jedná sa o objektovo-orientovaný webový aplikačný framework implementovaný v PHP 5. Vo svete má širokú podporu a je pomerne kvalitne dokumentovaný. Počas života aplikácii typu akým sa zaoberáme v tejto práci, sa kód stáva čoraz viac ťažšie udržiavateľný. Od klienta prichádzajú ďalšie a ďalšie požiadavky, ktoré musíme do pôvodného kódu na rôznych miestach dopĺňať.

Jednou z metód ako zlepšiť spravovateľnosť aplikácie je rozdeliť kód do troch separátnych sekcií. Tu sa po prvý krát dostávame k pojmu Model-View-Controller architektúra, ktorú Zend Framework využíva. Pozostáva z troch logických častí:

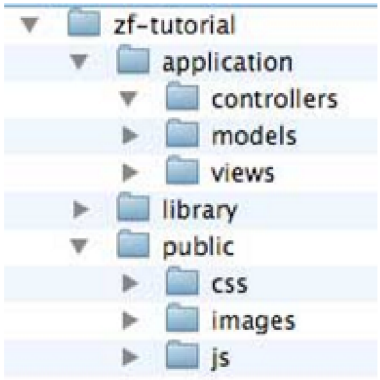
- **Model** – tzv. biznis časť logiky aplikácie, zaisťuje prístup k dátam a manipuláciu s nimi.
- **View** – interpretuje dáta reprezentované modelom do vhodnej podoby vo forme HTML.
- **Controller** – pracuje s modelom a view-om, má za úlohu dodať užívateľovi pri špecifickom požiadavku správne údaje v správnej forme.




Obrázek 3.3: Model-View-Controller architektúra [7]

Existujú určité technické požiadavky, ktoré Zend Framework vyžaduje. V prvom rade je nutné skontrolovať si verziu frameworku a verziu PHP, pretože staršia verzia ako PHP 5.1 môže spôsobovať problémy. Dôležitým krokom je nastavenie konfigurácie serveru v súbore *.htaccess*. Bez povoleného *mod_rewrite* módu by sme neboli schopní zobraziť inú ako titulnú stránku aplikácie.

Čo sa týka adresárovej štruktúry aplikácie nemusí byť striktne dodržiavaná, no manuály túto konvenciu silne doporučujú. Všetko má svoje miesto a vo výsledku má vývojár prehľadnú kontrolu nad jednotlivými častami vyvíjaného projektu.




Obrázek 3.4: Adresárová štruktúra ZF aplikácie

Kontrolér Zend Frameworku je navrhnutý pre podporu internetových stránok s peknými URL. Práve z toho dôvodu sú všetky požiadavky smerované cez jeden súbor *index.php*, nazývaný tiež bootstrapper. Ako nevýhodu Zend Frameworku môžeme zaradiť jeho rýchlosť, ktorá je s porovnaním s ostatnými frameworkami trochu nižšia. Na druhej strane má integrovanú podporu pre rôzne API od vedúcich vendorov ako Google, Amazon, či Flickr. Viac o Zend Frameworku je možné sa dozvedieť v [8].

## 3.3 Technológie pre vizualizáciu

Problematiku vizualizácie pre projekt riešený v tejto práci môžeme rozdeliť do troch častí:

- *Vizualizácia financií vynaložených na projektovú činnosť* – štatistiky vo forme grafov vykreslené pomocou knižnice JpGraph.
- *Vizualizácia geografickej lokality* – v tejto oblasti Google Maps nemá konkurenciu.
- *Vizualizácia projektových modelov* – samotné AutoCadové projekty vo formáte DWG (*natívny formát súborov/výkresov*) vizualizované pomocou nástroja Autodesk Freewheel.

Nasleduje stručný popis jednotlivých technológií.

### 3.3.1 Knižnica JpGraph

Je objektovo-orientovaná knižnica pre vytváranie grafov. Je určená pre PHP verzie 5.1 a vyššej. Vlastní kvalitnú dokumentáciu [5] a vyniká svojou jednoduchosťou implementácie. Knižnica je vyvíjaná pod UNIXOM, ale odzkúšaná a plne funkčná aj pod operačným systémom Windows. Na internete je možné nájsť veľké množstvo zaujímavých skriptov a knižníc pre PHP, avšak problém predstavuje vybrať tie, ktoré sú skutočne kvalitné. JpGraph nepochybne kvalitným nástrojom pre vytváranie grafov je. Umožnuje generovať bodové, sĺpcové, linkové, plošné, koláčové, pavučinové, geografické, burzové, radarové i 3D grafy. Navyše meniť štýl a podobu grafov či jednotlivé popisky je viac než jednoduché.

### 3.3.2 Google Maps

Google poskytuje svoje mapové rozhranie už dlhšiu dobu, jedná sa o jednoducho použiteľný a užívateľsky prívetivý servis poskytujúci nám takmer všetko čo potrebujeme vedieť/vidieť v oblasti mapových technológií. Integrácia statických máp do našich webových stránok, satelitné snímky celého sveta v rôznych podobách, Street View, navigácie, podpora služby v mobilných zariadeniach a mnoho ďalších nástrojov tvoria zlomok zo všetkého, čo nám spoločnosť Google v súčasnosti ponúka.

Pri zavedení technológie 3D posúvneho pohľadu Street View bola spoločnosť kritizovaná, pretože niektorým ľudom kvôli tejto technológií vykradli domy v čase, keď boli na dovolenke. Na druhej strane Google podporil veľa projektov ktoré pomáhali vypátrať nebezpečných zločincov či stratených ľudí ako napríklad miliardára Steva Fosseta, ktorý sa stratil pri lete osobným lietadlom nad Nevadskou púštou. Dokumentáciu ku Google Maps je možné nájsť v [11].

### 3.3.3 Autodesk Freewheel

Je voľne dostupná webová služba [6] pre zdieľanie a vizualizáciu 2D a 3D CAD (*Computer-Aided Design*) modelov bez nutnosti inštalácie či stahovania akéhokoľvek softvéru. Služba funguje na princípe Ajaxovo založeného web servisu. Pracuje s formátom DWF (*Design Web Format*), ktorý predstavuje bezpečný súborový formát vyvinutý spoločnostou Autodesk. Je určený pre distribúciu a komunikáciu CAD súborov.

DWF súbory obsahujú komprimované grafické dáta, preto sú podstatne menšie a rýchlejšie prenositeľné ako originálne CAD výkresy a modely. Program AutoCad pracuje s formátom DWG, tedá je nutné previesť konverziu tohoto formátu do formátu DWF. Táto konverzia je bližšie popísaná v kapitole 5. Pomocou Autodesk Freewheel služby sme schopní okrem prehliadania modelov ich aj tlačit či posielať mailom. Výhoda spočíva v tom, že služba beží na internete a my môžeme pracovať s CAD súbormi aj na počítačoch, na ktorých AutoCad nemusí byť nainštalovaný. Autodesk Freewheel je možné použiť aj v spojení s Google Maps.

Nevýhodu predstavuje fakt, že naše dáta sú zasielané na webový server. Tu sa ponúka otázka bezpečnosti našich informácii. Pri vypracovávaní projektu v tejto práci sú pomocou Autodesk Freewheel služby vizualizované iba projektové modely, ktoré užívatelia systému sami zvolia za publikovateľné. Jedná sa o čiastkové dizajny, ktoré spolu tvoria jeden celok, no po jednotlivých častiach pre nezainteresovaného človeka nedávajú zmysel. Aby bolo možné prehliadať, distribuovať a zdielať aj citlivejšie podnikové dáta, pre túto variantu je použitý špecifický softvér obalený PHP skriptom, ktorý prevádza konverziu CAD modelov do formátu obrázkov.

V tejto kapitole boli zhrnuté technológie potrebné pre realizáciu informačného systému vypracovávaného v rámci tohoto dokumentu. V nasledujúcej kapitole budú rozobrané konkrétne požiadavky zadávateľa, ich analýza a návrh samotnej aplikácie.

# Kapitola 4

# Analýza riešenia

Základom tejto etapy vývoja je analyzovať požiadavky klienta na systém. Kvalitná analýza riešenia by nám mala uľahčiť jeho implementáciu. Musíme zistiť, čo vo výsledku má systém robiť, ako má fungovať, interakciu užívateľov s ním apod. V prvom rade je nutné požiadavky nahromadiť, analyzovať a následne vytvoriť návrh systému vo forme modelu. Tento proces je zhrnutý v nasledujúcich sekciách.

## 4.1 Špecifikácia požiadavkov

Súčasný stav ukladania, zálohovania a spracovania informácií vo firme Hydroeco, s. r. o. nie je príliš uspokojivý. Z postupne hromadiacimi sa novými dátami priamo úmerne klesá miesto pre ich uchovanie. Projekty sú zálohované po preplnených diskoch, usb kľúčoch a cédečkách.

Taktiež vnútropodniková evidencia zaznámenávaná v archívoch nepredstavuje zrovna efektívny a prehľadný spôsob. Novo vytváraný systém má preto priniesť centrálnu správu všetkých informácii s ohľadom na jednoduchú a efektívnu manipuláciu s nimi. Získavanie požiadavkov prebehlo formou osobných stretnutí s vedením firmy a telefonickou komunikáciou.

Informácie, s ktorými bude systém pracovať môžeme zhrnúť v nasledujúcich bodoch:

- **Zadávateľ zákazky** – názov firmy, adresa firmy, IČO, telefonický kontakt, fax.
- **Zmluva na zakázku vystavená zadávateľom** – investor výstavby, cena za projekt, termín dodania projektu, dátum vystavenia zmluvy.
- **Zakázka** – názov, typ (vodovod, kanalizácia, čistička odpadových vôd), stav, miesto výstavby, stupeň vypracovania.
- **Projekt zakázky obsahujúci** – mapy dodané geodétmi, katastrálne mapy, prehľadové mapy, samotné .dwg súbory, záloha projektu.
- **Rozpočet zakázky** – kód cenníka, kód položky, popis položky, množstvo, merná jednotka, jednotná cena, cena za konštrukcie a práce, špecifikovaný materiál, hmotnosť, dph – import súboru.

- **Kniha jázd** – žiadateľ, cesta z/do, dátum cesty, počet členov cesty, stav kilometrov.
- **Služobná cesta** – miesto, dátum, riešená zakázka.
- **Pracovné voľno zamestnancov** – dovolenky, voľné dni.
- **Správa zamestnancov** – užívateľských mien, hesiel, práv.

Podstatu tvorí zakázka, na ktorú sa viaže zmluva, rozpočty a projekty vypracovávané pod ňou. Systém musí umožniť vyhľadávať zakázky podľa názvu, typu, jej stavu, roku, triediť ich v abecednom poradí podľa atribútov. Umožnený musí byť prístup ku všetkým jej častiam, teda ku projektom, zmluvám a rozpočtom. Možnosť pridávať do systému nové údaje a evidovať či mazať už existujúce je samozrejmosťou. Mala by existovať možnosť projekty nahrávať na server, prezerať ich, sťahovať a vytvárať ich zálohu. Ďalším požiadavkom je evidencia služobných ciest, knihy jázd, plánovanie pracovného voľna zamestnancov a správa užívateľov systému. Pri prijatí nového zamestnanca musí byť v systéme možnosť vytvoriť mu osobný účet. Taktiež pri prepustení je žiadané mu účet odstrániť, alebo zmeniť údaje súčasným zamestnancom.

Samostatný požiadavok v systéme tvorí problematika vizualizácie. V prvom rade je našou úlohou vizualizovať geografickú lokalitu vzťahujúcu sa ku konkrétnemu projektu jednotlivých zákazok. Zamestnanci počas projektovania sú často krát nútení vycestovať priamo do terénu na pozorovanie. Môže sa jednať o rôzne vodo-hospodárske stavby umiestnené nad zemou, či pod zemou ako čerpadlá, potrubia, šachty, čističky odpadových vôd či kanalizácie apod. Preto je dôležité poznať terén a lokalitu, kde bude táto stavba realizovaná. Cieľom je teda, aby vyvíjaný informačný systém umožnil grafický náhľad danej oblasti.

Riešenie poskytuje integrácia Google Maps API. Pomocou tohoto rozhrania sme schopní s využitím javascriptu upravovať mapu a jej časti, ako aj sa po mape pohybovať. Máme možnosť selekcie z troch typov máp, pričom najvhodnejší typ predstavuje mapa terénu udávajúca nadmorskú výšku pomocou vrstevníc. Pri tomto riešení bude mať každá zakázka uložená v databázi priradenú jednoznačnú lokalitu, podľa ktorej bude možné získať geografické údaje potrebné na jej zameranie.

Druhý problém predstavuje vizualizácia podkladov, teda samotných projektov vo forme AutoCadových .dwg (*drawing*) súborov. Tieto projekty by mali byť prezerateľné, mala by existovať možnosť manipulovať s nimi prostredníctvom informačného systému na všetkých počítačoch bez nutnosti potrebného softvéru ako AutoCad apod. Riešenie poskytuje webová služba Autodesk Freewheel založená na Ajaxe. S jej pomocou sme schopní vizualizovať projektové modely spolu s ďalšími funkciami ako tlačenie, odoslanie cez mail či export do obrázku. Poslednou požiadavkou je vizualizácia rozpočtov firmy pre jednotlivé zakázky pomocou knižnice JpGraph, ako už bolo spomínané v úvode.

## 4.2 Modelovanie a návrh systému

Pred samotnou implemetáciou musíme vytvoriť model systému, aby sme poznali ako bude fungovať a predišli tak prípadným chybám. V dnešnej dobe informačné systémy nie sú navrhované pomocou pera a papiera ako tomu bolo v minulosti. Dnes je zrejmé, že pri návrhu rozsiahlejších projektov sa nezaobídeme bez CASE (*Computer Aided Software Engineering*) nástrojov vychádzajúcich z jazyka UML (*Unified Modeling Language*).

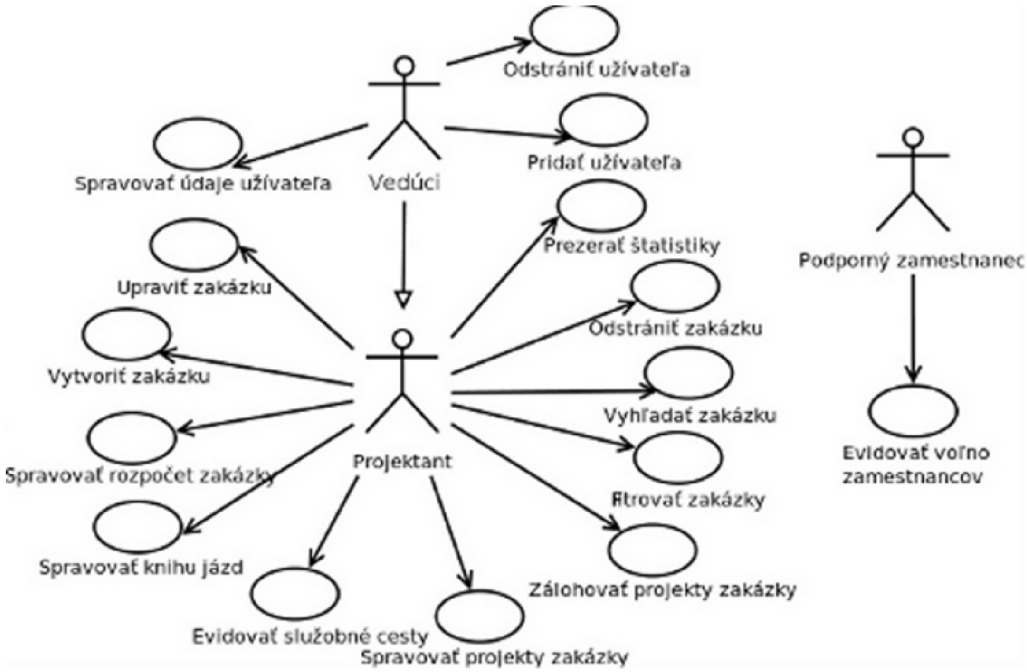
Umožnujú nám efektívne analyzovať a navrhovať systémy pomocou prepojenia jednotlivých techník UML, zdielať modely medzi členmi týmu, teda týmovú spoluprácu a mnoho ďalších funkcií. Existuje mnoho pomerne drahých platených CASE nástrojov. V tejto práci bol využitý program DIA, ktorý je voľne dostupný pod GPL (*General Public Licenses*) licenciou. Jeho funkcionalita plne postačuje pre návrh našeho informačného systému. Správne postupy pre analýzu a návrh informačného systému sú zhrnuté v publikácii [26].

Jazyk UML je výsledkom snahy analytikov a dizajnérov, ktorí v minulosti vytvárali metódy umožnujúce popísať objektovo-orientovanú analýzu a návrh. Je tvorený predovšetkým grafickou notáciou, ktorá vyjadruje analytické a návrhové modely. Umožnuje modelovať jednoduché i zložité aplikácie pomocou rovnakej formálnej syntaxe. Tým pádom sme schopní výsledky práce zdielať s ostatnými návrhármi a programátormi. Vybrané modely sú pochopiteľné aj pre zadávateľa, čo spôsobuje kvalitné vyjasnenie požiadavkov na vytváraný systém. Žiadny diagram nezachytáva navrhovaný systém ako celok, ale sústredí sa vždy práve na jeden pohľad na vyvíjaný systém. Bližšie informácie k jazyku UML a CASE nástrojom je možné nájsť v [16, 10, 21].

Už dlhšiu dobu sa vývojári bez ohľadu na svoju orientáciu na vývojové prostredie snažia modelovať typické interakcie užívateľov so systémom. Pre tieto účely slúžia tzv. Use Case (*Prípady použitia*) modely. Cieľom je aby sme lepšie pochopili skutočné požiadavky užívateľov na budúci systém a zároveň aby sme vymedzili rozsah budovanej aplikácie.
Tieto diagramy zachytávajú funkčnosť a rozsah možností užívateľov pri používaní systému. Všetky akcie sú vyvolávané aktérmi. Aktérom rozumieme externý objekt, ktorý si vymieňa informácie so systémom. V našom prípade sú aktérmi užívatelia používajúci systém. Každý z nich má svoju rolu, ktorá reprezentuje pohľad na systém z určitého uhlu. Model prípadu použitia je popísaný a zobrazený v nasledujúcej sekcii.

### 4.2.1 Model prípadov použitia

Nasledujúci diagram zobrazuje ako jednotliví užívatelia interagujú so systémom. Jedná sa o zjednodušený model zahrňujúci určitú úroveň abstrakcie. Môžeme vidieť aktérov s tromi rôznymi rolami v systéme. Projektant vykonáva rutinnú činnosť spojenú s vypracovávaním a evidenciou projektov. Vedúci pracovník rozširuje rolu projektanta, čo znamená, že môže vykonávať všetky jeho úkony a zároveň navyše je schopný spravovať užívateľov systému. Podporný zamestnanec má na starosti evidenciu pracovného voľna.




Obrázek 4.1: Use Case model zobrazujúci interakciu užívateľov so systémom

Po vytvorení Use Case diagramu sú zostrojené konkrétne prípady použitia systému užívateľom. Jeden z nich môže vyzerať nasledovne 4.1. Vzhľadom na obmedzený priestor v práci a prehľadnosť nebudú rozpísané alternatívne toky, ale iba toky hlavné. Alternatívne toky reprezentujú negatívnu odozvu systému na požiadavok užívateľa a obecne sa častokrát opakujú (*napr. výpis chybového hlásenia*).

| |
|---|
| *Prípad užitia:* VyhľadanieZakázky |
| *ID:* UC04 |
| *Stručný popis:* Vyhľadanie zakázok |
| *Aktéri:* Projektant |
| *Predpoklady:* Existuje aspoň jedna zakázka |
| *Hlavný tok:*<br>1. Prípad užitia sa spustí, keď Projektant vyberie položku „Vyhľadať zakázku“<br>2. Systém zobrazí stránku s nastavením parametrov vyhľadávania<br>3. Po nastavení parametrov Projektantom nasleduje potvrdenie<br>4. Systém zobrazí zakázky vyhovujúce vyhľadávaným kritériám |
| *Následné podmienky:* Bola vyhľadaná zakázka |
| *Alternatívne toky:*<br>VyhľadávanáZakázkaNenájdená |

Tabulka 4.1: Konkrétny prípad použitia systému užívateľom

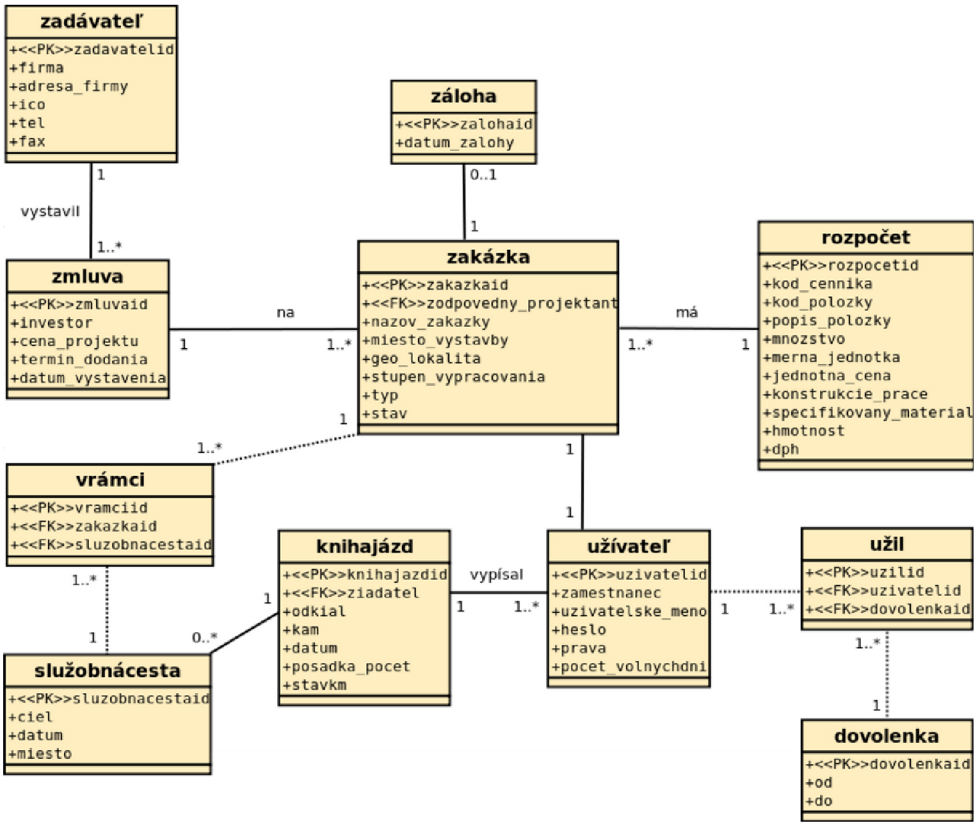
Následne potrebujeme vytvoriť databázový model našej aplikácie. Veľmi častou metodológiou pre konceptuálne modelovanie predstavuje E-R (*Entity-relationship*) diagram. Jeho zostrojenie je popísané v nasledujúcej sekcii.

### 4.2.2 Entity-Relationship diagram

Patrí medzi veľmi používané metodológie pre konceptuálne modelovanie. Hlavné komponenty tvoria entita, modelujúca objekty vyskytujúce sa v reálnom systéme a vzťah určujúci asociáciu medzi nimi. E-R diagram nám v podstate vyjadruje perzistentný model našej databáze, uchováva dáta v kľude.

Na obrázku 4.2 je možné vidieť viacero entitných množín, ktoré obsahujú cudzie kľúče. Cudzí kľúč predstavuje primárny kľúč inej tabuľky, pomocou ktorého sa môžeme na túto tabuľku odkazovať. Je zrejmé, že tabuľka Zakázka obsahuje najviac odkazov, kedže sa jedná o nosný prvok databáze. Musíme byť schopní vedieť sa odkázať na zmluvu, rozpočet, zálohu i projektanta tejto zakázky.

Čiarkovane znázornené vzťahy predstavujú tzv. asociačnú triedu. Je nápomocná v prípade, keď potrebujeme v našom návrhu zjednodušiť zložité vzťahy, akými sú napr. M:N. Na služobnej ceste sú obecne riešené nejaké zakázky. Jedna zakázka može byť riešená na jednej, i viacerých služobných cestách a zároveň na jednej služobnej ceste môže byť riešených viacero zakázok súčasne. Tento vzťah zjednodušíme vytvorením pomocnej asociačnej triedy, v ktorej máme presne uvedené spojenie jednotlivých zakázok so služobnými cestami. V prípade zamestnancov firmy a ich dovoleniek je prevedenie analogické. Po ukončení návrhu E-R diagramu sme schopní pracovať nad našou databázou. V ďalšej kapitole bude popísaná implementácia systému.




Obrázek 4.2: E-R diagram systému

# Kapitola 5

# Implementácia

V tejto časti bude popísaný technologický postup pri implementácii jednotlivých častí vyvíjaného systému. V úvode je riešená datová vrstva, návrh a vytvorenie databáze. Nasleduje návrh aplikácie, uživateľského rozhrania a popis aplikačnej vrstvy, ktorá zastrešuje operácie nad uloženými dátami. Vysvetlené bude tiež riešenie vizualizačných prvkov a konkrétnych komponent systému. Čitateľ porozumie procesu implementácie informačného systému a bude upozornený na chyby, ktoré môžu pri vývoji vzniknúť.

## 5.1 Technické prostriedky

Aplikácia bola vyvýjaná pod operačným systémom Windows Vista. Pre vytvorenie diagramov bol požitý open source program DIA. Verzie jednotlivých technických prostriedkov:

- vývojové prostredie Netbeans IDE 6.9,
- server Apache 2.2.11,
- správa databáze PhpMyAdmin 3.2.0.1,
- PHP 5.3.0,
- MySQL 5.1.36,
- Zend Framework 1.10,
- jQuery 1.4.4,
- jQuery-UI 1.8.10,
- spracovanie grafiky pomocou GIMP 2.6.

## 5.2 Zostavenie štruktúry databáze

Návrh databáze vo forme E-R diagramu je prevedený na jednotlivé konkrétne tabuľky. Pri prevádzaní sa držíme systematických pravidiel, aby nedošlo k nesprávnemu návrhu databáze. Medzi hlavné problémy návrhu patria opakujúce sa informácie (*redundancia*), nemožnosť reprezentovať, alebo získať určitú informáciu, zložitá kontrola integritných obmedzení

a vytváranie nadbytočných tabuliek. Našou snahou je vytvoriť štruktúru, ktorá je udržiavania schopná a modifikovateľná spolu s dátami, ktoré sme schopní jednoducho získať. Pri dobrom návrhu je nám investovaný čas navrátený pri implementácii. Najčastejšie chyby návrhu databáze a postupy pre návrhy správne sú zhrnuté v [12, 27].

Prvým krokom transformácie diagramu do tabuľkovej podoby je odstránenie zložených a viachodnotových atribútov entity. Napr. položka adresa je rozpísaná na dielčie časti ako ulica, číslo a psč. Tento krok bol uvážený už pri samotnom zostrojovaní E-R diagramu a tým pádom pri transformácii sa ním už nemusíme zaoberať. Pri vytváraní tabuliek treba brať ohľad na vzťah medzi entitami, ktoré na ne prevádzame. Cudzí kľúč, teda odkaz na primárny kľúč inej entity obsahuje tabuľka, ktorá bola vo vzťahu s druhou tabuľkou zastúpená vyššou kardinalitou. Pri vzťahu M:N je vytvorená samostatná tabuľka odkazujúca sa na tieto entity.

Pre pomenovanie primárnych kľúčov je vhodné zvoliť si určitú konvenciu, pretože ich pri programovaní často využívame. Každý primárny kľúč je teda nazvaný podľa tabuľky, ktorú identifikuje spolu s príponou „id“. Najčastejšie využitým dátovým typom je typ `VARCHAR, INT a DATE`. Pri nastavovaní dĺžky reťazca, ktorý slúži ako užívateľské heslo sa vynára malá záludnosť. Pre vyššiu bezpečnosť sú heslá uložené v databáze šifrované hashovacou funkciou `MD5()` využívajúcou algoritmus *Message-Digest algorithm*. Tá vyžaduje aby šifrovaný reťazec bol 32 bitový, pretože výsledný unikátny hash reťazec je zložený z 32 hexadecimálnych znakov (128 bitov). V prípade, že pre heslo nastavíme nesprávny počet znakov, funkcia nefunguje správne a dostávame sa do problému. Tento algoritmus nie je rezistentný voči kolízii a niektorým útokom, ktoré sú cielené pre zistenie zašifrovaného reťazca. Pre zvýšenie bezpečnosti môžeme kryptovať viac krát za sebou, alebo heslo saltovať – pripojíme k nemu ďalší dostatočne dlhý reťazec. Pre účely tohoto projektu bolo postačujúce jednoduché základné šifrovanie.

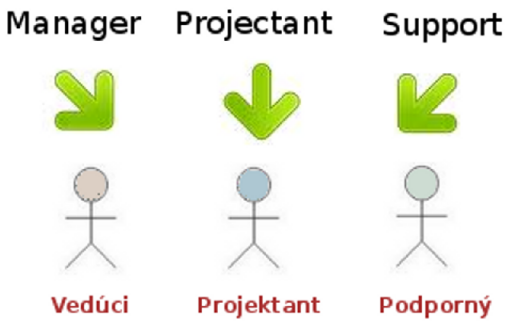
Celkovo systém obsahuje 11 databázových tabuliek. Atribúty ako stav či typ zákazky boli modelované pomocou čísiel. Pri stave teda rozumieme 0 ako rozpracovanú, aktuálnu zakázku a 1 ako dokončenú, archivovanú zakázku s vytvorenou zálohou. Typ zákazky určujú čísla 0, 1 a 2 ako vodovod, čistička odpadových vôd a kanalizácia.

Pri vytváraní štruktúry databáze nebola využitá možnosť kaskádových odkazov, ktoré sú využívané pri vymazávaní tabuliek spolu s ich referenciami. S postupom času by sa nám tieto väzby hromadili a vytratila by sa prehľadnosť. Z toho dôvodu je v tomto projekte vymazávanie tabuliek a ich referencií v plnej réžii vývojára. Presne si sami určujeme, ktoré tabuľky treba odstraňovať v akom poradí. Posledným nemálo dôležitým krokom je export skriptu pre vytvorenie databáze. Poslúži nám ako záloha v prípade chyby i v procese vytvárania databázy počas neskoršej fázy na serveri, kde bude systém umiestnený.

## 5.3 Architektúra systému

Adresárová štruktúra aplikácie je vytvorená v striktnom formáte z dôvodu prehľadnosti a možnosti budúcich rozšírení. Ako je možné vidieť na obrázku 5.1, pre každú rolu v systéme je zostavený samostatný kontrolér. Vo výsledku dekomponujeme systém na menšie časti, ktoré sa stanú lahšie udržovatelnými, oddelenými logickými celkami. Na obrázku nie je zahrnutý *Index* kontrolér, ktorý slúži pre všetkých užívateľov systému bez ohľadu na ich práva. Má na starosti ich autentifikáciu a zastrešuje funkcie zobraziteľné pre všetkých (ako napr. nástenka).

Každý kontrolér obsahuje metódy/akcie, ktoré korešpondujú s konkrétnymi HTML stránkami (*View*). Tieto dve časti sú previazané medzi sebou svojím názvom. Názov metódy a príslušného grafického náhľadu vo forme webovej stránky musia byť totožné. Každá akcia




Obrázek 5.1: Architektúra controllerov k jednotlivým rolám užívateľov

vyvolaná užívateľom vykoná svoju úlohu a zobrazí požadovaný výsledok. Všetky súbory súvisiace s príslušným kontrolérom sú umiestnené v adresári s jeho názvom. V adresári model je uložená celá biznis logika aplikácie. V tejto časti sú uložené triedy pre prácu s databázou. Vo verejnom public adresári je umiestených niekoľko dôležitých podadresárov:

- **scripts** – obsahujúci pomocné skripty a jQuery knižnice,
- **styles** – obsahujúci štýly pre grafický vzhľad aplikácie,
- **images** – obsahujúci obrázky a toolbary použité v systéme,
- **uploads** – obsahujúci adresárovú štruktúru určenú pre projeky zákazky,
- **backup** – obsahujúci zálohu zakázok.

V adresári layout je umiestnený základný dizajn aplikácie. Pridávať štýl každej jednej stránke je nezmyselné, preto pripojíme dizajn centrálne na jednom mieste. Adresár library obsahuje všetky potrebné knižnice Zend Frameworku i knižnice JpGraph pre prácu s grafmi. V Zend Frameworku sú všetky komponenty objektami, i samotné formuláre predstavujú triedy. V konfiguračnom súbore *Application.ini* nastavujeme lokálnu zónu, čas, mód vývoja (produkcia/testovanie), prístup do databáze, ku knižniciam a ďalšie užitočné nastavenia. Konfiguráciu prístupu k databáze môžeme vidieť na nasledujúcom obrázku 5.2. Centrálne na jednom mieste nastavíme potrebné údaje a neskôr už len využívame prístup pomocou prednastaveného adaptéru.

```
resources.db.adapter = "PDO_MYSQL"
resources.db.params.host = "localhost"
resources.db.params.username = "root"
resources.db.params.password = ""
resources.db.params.dbname = "hydroeco"
resources.db.isDefaultTableAdapter = true
...
$db_access = $this->getDefaultAdapter();
```

Obrázek 5.2: Prednastavená konfigurácia pre prístup do databázy

Tieto nastavenia budú časom pri zavedení systému na server samozrejme upravené. Pri návrhu musíme tiež premyslieť otázku bezpečnosti súborov, ktoré budú na serveri uložené.

Treba zabezpečit aby k nim nikto nepovolený nemal prístup. Riešenie poskytuje vytvorenie skrytého *.htaccess* súboru, v ktorom nastavíme aby k súborom v tomto adresári nemal nikto prístup. V konfigurácii PHP prostredia je tiež potrebné zvýšiť veľkosť súborov nahrávaných na server, pretože AutoCadové projekty sú obecne pomerne veľké. Dôležitým prvkom informačného systému sú jednolivé prístupové role užívateľov. Na tie sa pozrieme v nasledujúcej sekcii.

## 5.4 Role a práva užívateľov

Systémové práva majú tri úrovňe. Rola projektanta je určená pre zamestnancov, ktorí vytvárajú a spravujú celú projektovo-zakázkovú činnosť. Vedúci má na starosti správu systému a užívateľov.

Aj keď je jeho rola oprávnená vykonávať aj činnosť projektantov, v implementácii to je vyriešené inak. Vedúci má vytvorené dva účty. Jeden pre správu systému a druhý ako projektant. Je to z dôvodu, aby sa zjednodušilo používanie systému a neplietla evidencia zamestnancov medzi sekciu, kde sú spravované projekty firmy. Podporný zamestnanec má na starosti vnútropodnikové evidenčné činnosti. Tieto role budú v budúcnosti rozširované. Práva v systéme má na starosti trieda `Library_Acl`. Všetky stránky systému berie ako zdroje, ku ktorým následne určuje prístup na základe role užívateľa. Povoluje mu prístup na konkrétny kontrolér a akciu.

```
$this->addRole(new Zend_Acl_Role('projektant'));
$this->addRole(new Zend_Acl_Role('veduci'));
$this->addRole(new Zend_Acl_Role('podporny'));
...
$this->allow('role','controller/action');
```

Obrázek 5.3: Trieda pre prístupové práva, vytvorenie rolí a povolenie prístupu

Pri vstupe do systému sa užívateľom zobrazí odlišné menu podľa typu ich práv. Menu položky sú načítavané zo XML súboru, v ktorom je definované k akému zdroju a k akým právam patrí príslušná stránka. Ďalšou vecou, ktorú je treba zabezpečiť je bezpečnosť prístupu do systému pre jednotlivé role. Projektant by sa nemal dostať do systému s právami vedúceho a ani naopak. Každá rola má svoj vymedzený priestor a po správnosti teda túto situáciu treba ošetriť, inak by bolo jednoduché pomocou prepísania url adresy dostať sa kam potrebujeme. To má na starosti plugin, ktorý je spúštaný ešte pred samotným načítaním stránky. Kontroluje, či má k danej stránke práve prihlásený užívateľ potrebné práva. Ak nemá, je presmerovaný na stránku s prihlásením.

## 5.5 Autentifikácia a vstup do systému

Autentifikácia predstavuje proces overenia identity užívateľa pristupujúceho do systému. Pri vstupe sú skontrolované údaje vyplnené užívateľom do formulára s jednotlivými stĺpcami uloženými v databázy. Pri validnom vyplnení je prístup povolený, pri nevalidnom je vypísaná chybová hláška. Formuláre v celom programe sú tvorené triedami dediacimi od objektu `Zend_Form`. Pre ošetrenie vstupu zadávaného do formulára slúži trieda `Zend_Filter`. Zabraňuje vloženiu čísiel na miesto textu, textu na miesto čísiel, HTML či PHP tagov a ďalších kombinácii ktoré by mohli narušiť chod našej aplikácie. Pri porovnávaní vyplnených

hesiel s heslami v databázy dochádza často k chybe, že zabudneme heslá z formulára zašifrovať funkciou `MD5()` a porovnávame klasický text s hashom uloženým v databázy. To predstavuje pomerne častý zdroj chýb, na ktorý si treba dávať pozor.

Ešte pred samotným procesom validácie však musíme skontrolovať, či už náhodou užívateľ prihlásený nie je. Ak áno, je presmerovaný na úvodnú stránku informačného systému. Pre zvýšnie bezpečnosti bol vytvorený plugin, ktorý pred načítaním každej stránky skontroluje, či je užívateľ prihlásený. Zabraňuje tým nežiaducim vstupom neprihlásených užívateľov do systému (napr. cez url adresu).

## 5.6 Grafické užívateľské rozhranie

Základný dizajn je doplnený o fotografie z kancelárie firmy, aby výsledný informačný systém po vizuálnej stránke modeloval skutočné pracovné prostredie.


Obrázek 5.4: Náhľad na grafické rozhranie vstupu do systému

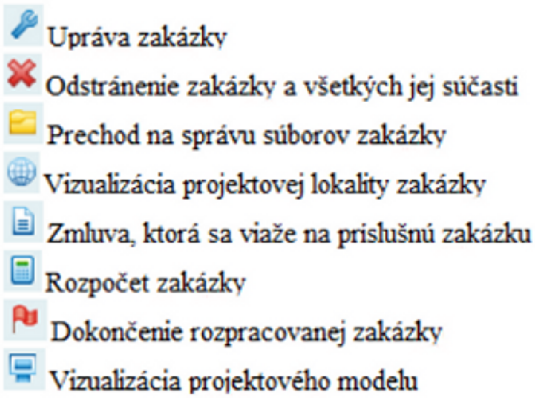
Ako už bolo spomenuté, štýly sú pripojené v layoute aplikácie v centrálnom bode. Layout je registrovaný na mieste, kde sú pridávané pluginy aplikácie – v *Bootstrap* súbore. V niektorých častiach systému je žiadúce layout od stránky odpojiť. Užívateľské rozhranie slúži ako prodstriedok pre ovládanie systému a preto by v jeho návrhu mala byť zahrnutá jednoduchosť i efektivita ovládania. Hlavná funkcionalita systému bude sústredená okolo zakázky, preto je potrebné vymyslieť vhodné rozhranie pre manipuláciu s týmto prvkom. Na nasledujúcom obrázku 5.5 je možné vidieť jeden záznam databáze a rozhranie pomocou ktorého sme schopní so záznamom manipulovať. Toolbary použité v aplikáciách predstavujú jednoduchú navigáciu prepojenú s požadovanou funkcionalitou.

| Názov zakazky | Zodpovedný projektant | Miesto výstavby | Stupeň vypracovania | Typ | Stav | Control-panel |
|---|---|---|---|---|---|---|
| Tatralandia | Lubica Dulova | Liptovsky Mikulas | NV2 | vodovod | rozpracovana | |

Obrázek 5.5: Toolbarové rozhranie pre manipuláciu so záznamom

Zľava doprava toolbary predstavujú funkcie editácie zakázky, zmazania záznamu, prechod na súborový systém, grafický náhľad oblasti výstavby projektu zakázky, prehliadanie zmluvy na zakázku, rozpočtu, možnosť uzatvorenia a zálohy rozpracovanej zakázky a možnosť vizualizácie projektového modelu.

Kedže koncoví užívatelia, ktorí budú aplikáciu využívať predtým informačný systém nepoužívali, užívateľské rozhranie bolo implementované s cieľom jednoduchého ovládania. Na nasledujúcom obrázku je rozhranie zobrazené prehľadnejším spôsobom.




Obrázek 5.6: Prehľad funkcionality rozhrania

## 5.7 Operačná logika

Databáza je vytvorená pri integrácii aplikácie na server spustením skriptu, ktorý vytvorí databázovú štruktúru. V tejto chvíli je backend našej aplikácie vytvorený a my môžeme s databázou začať pracovať.

Zakázka je nosnou častou databáze, z ktorej sa budeme odkazovať na zmluvu k nej, zadávateľa ktorý ju vystavil, jej rozpočet, zálohu i projektanta ktorý ju má na starosti. Pri vkladaní do databáze postupujeme systematicky. V prvom rade vkladáme údaje o zadávateľovi, následne údaje spojené so zmluvou. Aby sme dodržali referencie a validné prepojenie všetkých častí po vložení záznamu na ktorý sa chceme odkazovať, musíme získať jeho primárny kľúč ktorý vložíme do dalšej tabuľky. Identifikáciu zadávateľa potrebujeme vložiť do tabuľky zmluvy, aby sme sa vedeli odkázať kto danú zmluvu vystavil. Pre zistenie primárneho kľúču aktuálne vloženého záznamu slúži funkcia `mysql_insert_id()`.

Takto postupujeme až pokiaľ nemáme v databáze uloženú zakázku a sme schopní sa odkázať na všetky jej súčasti. Pri mazaní je postup podobný. Nie je možné zmazať rozpočet alebo zmluvu nejakej zakázky, môžeme ich len upravovať. Pri vymazaní by sa stratil odkaz a bola by narušená konzistencia v databáze. Súčasti zakázky sú vytvárané i mazané súčasne s ňou. Pri jej vytvorení sa automaticky vytvorí aj adresárová štruktúra pre projekty danej zakázky. Názov adresára je názov primárneho kľúča zakázky. Pri mazaní je adresár rekurzívne prechádzaný a ruší sa odkaz na súbory uložené na disku (funkcia `unlink()`), až nakoniec je celý odstránený.

Rozpočet zakázky obsahuje väčšie množstvo atribútov, preto jeho vloženie do databáze prebieha formou importu súboru vo formáte XML. Pre jeho načítanie a rozdelenie na jednotlivé elementy bola použitá trieda `XMLReader`. Keď už máme databázu naplnenú zobrazíme jednotlivé záznamy do prehľadnej tabuľky. Počet záznamov je obmedzený iba veľkosťou

miesta na serveri, teda predpokladáme existenciu veľkého počtu záznamov. Preto je potrebné implementovať paginator (*číslovanie stránok*). Nastavíme presný počet zobrazených záznamov na jednu stránku a môžeme medzi nimi listovať.

Obecný postup pri operáciách nad databázov je v prvom rade vyplnenie formulárov, filtrov a nastavení užívateľom. Tie sú vyhodnotené v Controlleri, ktorý sa na dáta dotazuje prostredníctvom Modelu. Následne je výsledok zobrazený užívateľovi na stránku – View.

### 5.7.1 Vyhľadávanie

Veľká výhoda v manipulácii s elektronickými dokumentami spočíva najmä v možnosti vyhľadania toho čo potrebujeme. V našom informačnom systéme sme schopní vyhľadať zakázku podľa roku, typu, stavu, projektanta ktorý ju vypracuváva či názvu. Vyhľadávacie filtre je možné aplikovať aj na ostatne súčasti spojené so zakázkou a všetky ostatné prvky uložené v databáze. Princíp činnosti je v základe rovnaký. Na základe výberu typu filtru predáme kontroléru parametre, na základe ktorých sa dotáže cez model na nami požadované špecifické dáta.

Informačný systém musí byť schopný po celý čas interakcie s užívateľom udržiavať kontext. Cez webový prehliadač však komunikácia so serverom prebieha protokolom Http, ktorý je bezkontextový. Aby sme mohli predávať parametre zo stránky na stránku existuje niekoľko typov ako toho možeme docieliť. Môžeme ustanoviť `Session`, uložiť dáta do `Cookie`, alebo propagovať kontext pomocou url adresy. Tretí spôsob má nevýhodu, že url adresa má obmedzenú veľkosť. V našom prípade sú za jej pomoci predávané len jednoduché identifikátory či názvy adresárov a súborov, takže tento spôsob je postačujúci a zároveň jednoduchý.

## 5.8 Správa súborov

Aktuálna metodika uchovania projektov vo firme spočíva v nahromadení viacerých typov súborov v jednom adresári s názvom zakázky. Nový systém má za úlohu zaviesť okrem centrálnej správy súborov aj lepší manažment. Užívateľ sa nemusí o nič starať, jednoducho nahráva súbory do systému. Ten ich podľa koncovky triedi a vytvára prehľadnú adresárovú štruktúru 5.7. Prehľadové, katastrálne a geodetické mapy sú umiestnené do zložky mapy. Zmluvy a evidenčné listy sú umiestnené do zložky zmluvy a samotné AutoCadové projekty a ostatné súbory do adresára projekty. Na nasledujúcom obrázku je možné vidieť pohľad na súborový systém v prostredí aplikácie.


Obrázek 5.7: Súborový systém

Pri nahrávaní AutoCadových projektov dochádza na pozadí k automatickej konverzii týchto modelov na obrázky a pdf súbory. To má na starosti špecifický softvér obalený PHP skriptom. Tieto výsledné súbory sú umiestnené do poslednej zložky s názvom projekty-export. Grafy rozpočtov pre jednotlivé zakázky sú exportované do adresára s názvom roz-

počet. Všetky tieto súbory v odlahčenom vyexportovanom formáte je možné jednoduchšie distribuovať napr. prostredníctvom mailu. Touto cestou sme schopní zobraziť AutoCadové projekty aj na počítačoch, kde tento softvér nie je nainštalovaný. Ďalšou výhodou je možnosť pripojenia obrázkových modelov a grafov do dokumentácie.

Sťahovanie súborov je implementované nastavením http hlavičiek pre prehliadač. Jediné čo potrebujeme je predať si parametrom identifikátor zákazky, názov adresára a súboru ktorý chceme stiahnuť. Následne stačí vytvoriť hlavičky a súbor je možné získať. Pri dokončení rozpracovanej zakázky je vytvorený archív jej súborov a projektov nazvaný podľa dátumu vytvorenia a umiestnený do zložky s názvom príslušnej zakázky. Vytvorenie archívu zabezpečuje trieda `Ziparchive`. Adresár zakázky je rekurzívne prechádzaný a súbory v ňom sú pridávané do výsledného archívu.

## 5.9 Vizualizačné prvky

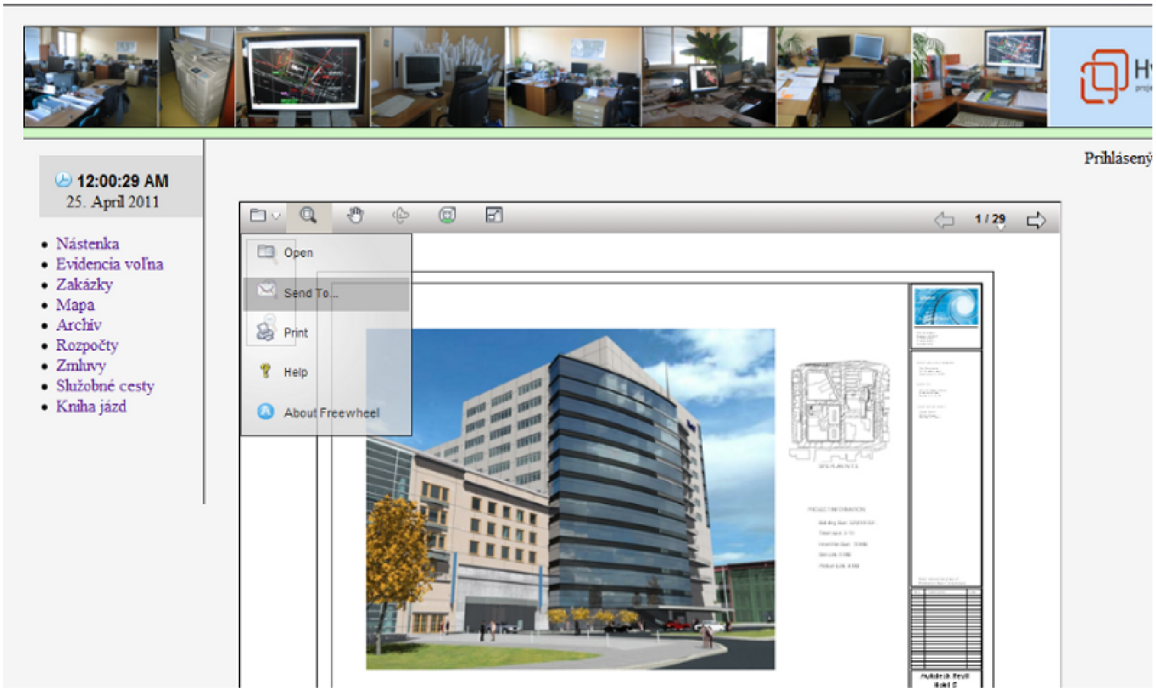
Pri vytvorení každej zakázky sa do databáze uloží geografická lokalita, podľa ktorej bude možné oblasť výstavby identifikovať na mape. Rozhranie Google Maps je implementované pomocou JavaScriptu. V prípade, že užívateľ nemá v prehliadači JavaScript povolený, je na to upozornený. Mapa je vytvorená v zvolenom označenom HTML kontajneri, ktorým typicky býva `DIV` element. Javascript tento kontajner identifikuje pomocou funkcie `document.getElementById()`.

Ďalším krokom je získanie lokácie, ktorú máme záujem zobraziť. Pre tieto účely slúži trieda `GClientGeocoder`, ktorá komunikuje s Google servermi a príjma súradnice geokódovaných adries. Pomocou metódy `GetLocations(address,callbackfunction)` je vykonaná konverzia adresy na súradnice zemepisnej dĺžky a šírky. Callback funkcia má na starosti postarať sa o odpoveď z Google serveru. Tá je spracovaná a požadovaná lokalita je zobrazená na mape. S mapou potom môžeme pracovať a pridávať do nej rôzne značky, popisky, nastavovať priblíženie apod. Do druhého typu mapy v informačnom systéme sú pridané oblasti pre aktuálne zakázky na ktorých firma pracuje. Jedná sa o zakázky vypracovávané v časovom rozmedzí súčasnosti až dvoch rokov dozadu. Pri dotazovaní sa na tieto dáta bola využitá sql funkcia `CURRENT_DATE()` a kľučové slová `YEAR` a `INTERVAL`.

Aby sme boli schopní vizualizovať projektové modely pomocou webovej služby AutoDesk Freewheel, musíme previesť ich konverziu na formát `DWF` a umiestniť ich do verjného adresára aby ku nim mal server prístup. Následne stačí vizualizovaný prvok umiestniť na stránku medzi `<iframe>` tagy so zdrojovou cestou k súboru. Na server môžeme poslať spolu s cestou parametre určujúce výšku a šírku zobrazovaného modelu.

Kedže môžu byť na server zaslané citlivé dáta, užívateľovi je na začiatku celého procesu zobrazený dialóg, bez ktorého potvrdenia sa konverzia ani vizualizácia neuskutoční. Dialóg predstavuje jedinú možnú cestu ako sa dostať k vizualizácii modelu. Pri jeho potvrdení sa nastavuje `$_SESSION`. Tá je odregistrovaná už počas zobrazenia modelu. V tomto mieste je nastavenie kontrolované. Ak užívateľ pristúpil na stránku bez potvrdenia, je presmerovaný. Je to ošetrenie prípadu, keď by užívateľ zadal do prehliadača priamu url adresu a snažil sa tak dialóg obísť. Prevedenie formátu dwg na dwf prebieha rovnakým spôsobom ako konverzia dwg do formátu obrázku, alebo pdf súboru.

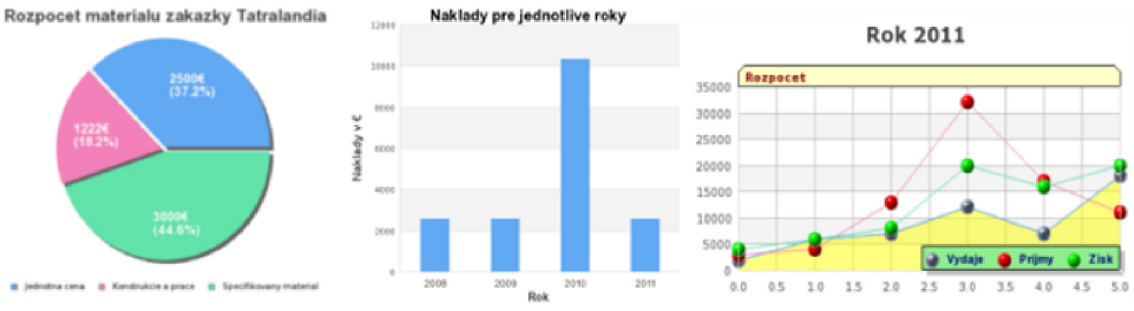
Knižnicu JpGraph pre vytváranie grafov je potrebné začleniť do prostredia našej aplikácie. Po nastavení správnych ciest sme pripravení vytvárať objekty reprezentujúce grafy. Po ich vytvorení ich naplníme potrebnými hodnotami na ktoré sa dotážeme do databáze. Existuje veľa druhov nastavení pre popisky a vizuálnu stránku grafov. Pri použití `_IMG_HANDLER` parametru máme možnosť grafy zobrazovať na stránke a zároveň ich exportovať do formátu




Obrázek 5.8: Vizualizácia projektového modelu

obrázku uložitelného na miesto, kde potrebujeme. Knižnica JpGraph má jednu nevýhodu a to tú, že grafy nemôžu byť zobrazené na jednej stránke spolu s HTML kódom. Riešenie má dva spôsoby. Prvý spôsob je vložiť do stránky obrázok, ktorý sa odkazuje na svoj zdroj na PHP súbor obsahujúci kód pre vytvorenie grafu. Pri tejto možnosti je problematické do externého PHP súboru predávať hodnoty získané z databáze.

Pri projekte bola použitá druhá varianta. Na stránke zobrazíme iba grafy a HTML kód odpojíme. Najlepšou cestou je otvoriť stránku s grafom v novom, čistom okne webového prehliadača. Využité boli doposiaľ tri typy grafov. Koláčový graf bol použitý pre rozloženie financií v rozpočte konkrétnej zakázky. Stĺpcový graf je použitý pre prehľady nákladov na realizáciu zakázky v posledných rokoch. Posledným použitým grafom je čiarový, ktorý reprezentuje vnútropodnikové financie.




Obrázek 5.9: Náhľad na použité grafy v systéme

Sekcia rozpočtov a grafov bude v budúcnosti rozširovaná, doposiaľ boli navrhnuté technické princípy ich realizácie a základné manipulácie s databázovými údajmi.

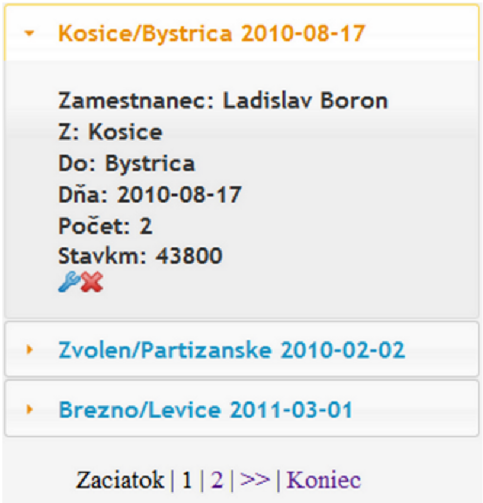
## 5.10 Komponenty využívajúce rozhranie jQuery

V závere kapitoly bude popísané riešenie dvoch vybraných prvkov systému využívajúcich rozhranie jQuery. Toto rozhranie zlepšuje vizuálnu podobu stránky a pridáva interaktivitu do nášho systému. Načítava sa iba určitá časť stránky pričom zbytok ostáva bez zmeny. Po zhrnutí riešenia týchto častí nasleduje kapitola o testovaní aplikácie nad reálnymi dátami.

### 5.10.1 Kniha jázd

Kniha jázd je prvou komponentou využívajúcou rozhranie jQuery. Na webových stránkach rozhrania [22] si máme možnosť zvoliť dizajn, ktorý pre našu aplikáciu použijeme. Jednotlivé záznamy ciest v systéme je možné roztvoriť s efektom harmoniky (*accordion object*), ktorý nám rozhranie poskytuje. Tu nastáva problém, pretože záznamy sú zobrazované interaktívne, ale hlavný obsah stránky je po zväčšení výšky natiahnutý a dochádza tak k načítaniu celej stránky. To má za dôsledok, že interaktivita, ktorú rozhranie prinieslo je okamžite narušená trhavým efektom.

Riešenie poskytuje pomocná jQuery funkcia, ktorá vypočítava a upravuje výšku hlavného obsahu stránky. Pri prvej zmene objektu `accordion` je výška stránky zväčšená o výšku obsahu záznamu umiestneného v danom objekte. Pri každej ďalšej zmene výška ostáva, pretože otvorený je vždy len jeden kontajner objektu. Pri jeho uzatvorený je výška stránky znížená opäť na pôvodnú veľkosť.




Obrázek 5.10: Accordion objekt z jQuery rozhrania

Kniha jázd je vypĺňaná zamestnancom pri každej služobnej ceste, preto tieto dve tabuľky spolu úzko súvisia. Samotný záznam o služobnej ceste v knihe jázd nie je možné vymazať, pretože by bola narušená konzistencia. Kniha však obsahuje aj druhý typ záznamu a ten už vymazať môžeme. Jedná sa o situáciu, keď si zamestnanec zoberie služobné auto pre súkromné účely. Vtedy sa nejedná o služobnú cestu, teda žiadny odkaz neexistuje. Hlavnou úlohou knihy jázd je evidencia stavu služobného vozidla, teda počet najazdených kilometrov a mapovanie služobných ciest.

### 5.10.2 Kalendár

Predstavuje ďalší prvok rozhrania jQuery. Plní pomocnú úlohu pri výbere dátumu počas plánovania pracovného voľna zamestnancov. Plán voľna neuvažuje víkendy, preto boli z kalendára odstránené. Existuje niekoľko spôsobov ako toho docieliť. Najjednoduchší je nastaviť v štýle kalendára, aby víkendy neboli vôbec zobrazené.

Pre vynechanie štátnych sviatkov je možné pred vykreslením kalendára zavolať callback funkciu, ktorá z neho odstráni vopred definované dátumy. Notácia dátumu kalendára jQuery a dátumu používaného v databáze je odlišná, preto ich treba upravovať. Pre tieto účely boli použité PHP funkcie `explode()` a `str_replace()` v kombinácii s konkatenáciou reťazcov.

Každému zamestnancovi vedúci pracovník nastaví na začiatku roka počet voľných dní pre daný rok. V systéme je tento počet zobrazený a je možnosť prehliadať dovolenky zamestnanca v minulosti. Naplánovať pracovníkovi dovolenku je možné v prípade, keď má na to dostatočný počet voľných dní a vybrané dátumy predstavujú validné rozmedzie, tzv. nie je možné dovolenku naplánovať do minulosti apod.


Obrázek 5.11: Kalendár pre plánovanie pracovného voľna

V prípade nevalidného výberu dátumu, alebo nedostačujúceho počtu voľných dní zamestnanca je užívateľovi systému zobrazená chybová hláška s popisom chyby.

# Kapitola 6

# Testovanie systému

Hlavným účelom testovania je overenie či aplikácia funguje spôsobom akým má a zistenie chýb, ku ktorým mohlo dojsť pri implementácii. Prenesieme sa do role užívateľa používajúceho systém. Predávame systému prázdne, alebo nezmyselné dáta a snažíme sa nájsť chybu vo funkčnosti. Kontrolujeme, či niečo nie je možné vylepšiť a zvýšiť tak užívateľove pohodlie pri používaní systému. Fáze vývoja implementácie a testovania spolu úzko súvisia. Testovanie nám poskytuje spätnú väzbu, kde pri objavení nefunkčnej súčasti systému máme možnosť ju opraviť v implementácii.

Pre účely testovania bolo pripravených 30 rôznych reálnych zákaziek, pričom každá obsahuje projekty, zmluvy, dokumentácie, mapy, rozpočty a ostatné pomocné súbory. Testovanie prebiehalo počas implementácie aplikácie (*Test-driven development*) a po jej dokončení v 4 systematických fázach.

Prvá etapa spočíva v testovaní validácie všetkých vstupných prvkov systému, obecne formulárov a dialógov. Do pola, ktoré očakáva číselnú hodnotu nemôže byť umožnené zadať text a naopak. Preto tieto kombinácie úkonov musíme ošetriť a kontrolovať či sú vypisované adekvátne chybové hlásenia. V tejto fázy je zároveň optimalizovaná funkčnosť aplikácie vo webových prehliadačoch Mozzila Firefox a Google Chrome.

Druhá fáza testuje databázovú logiku a zachovanie konzistencie uložených dát. Po sérii vkladaní, editovaní a mazaní sa kontroluje či väzby medzi tabuľkami neboli narušené.

Tretia fáza kontroluje súborový systém a vizualizačné prvky. Prebieha kontrola existencie súborov, ich triedenia, rýchlosťi nahrávania na server, archívu súborov, kontrola či nahraté alebo stiahnuté súbory neboli poškodené a ďalšie podobné úkony.

V poslednej fázy napĺňame systém ďalšími dátami a sústredíme sa na výkonnosť aplikácie. Zend Framework nedosahuje príliš uspokových výsledkov čo sa týka svojej rýchlosti v porovnaní s ostatnými frameworkami. Pri nahrávaní väčších súborov dochádza na pozadí zároveň k ich konverzii do formátu obrázku a pdf súboru. Tento proces trvá v niektorých prípadoch až niekoľko sekúnd, čo predstavuje v budúcnosti života aplikácie optimalizovania hodný faktor.

Pri priebehu testovania bolo odhalených niekoľko chýb a nedostatkov v aplikácii, ktoré boli spätne opravené. Jednalo sa napr. o chyby v url odkazoch, neefektívnu prácu s databázou, neošetrené vstupy formulárov, načítanie XML súboru apod. Tým je ukončená prvá časť testovania. Druhá časť testovania bude prebiehať, keď bude aplikácia umiestnená na cieľový server. Súčasný stav projektu stojí na výbere správneho hostingu s potrebnými technológiami za rozumnú cenu. Následne bude zakúpená doména na ktorú bude kompletný systém presunutý. Zhodnotenie výsledkov, ďalšie rozšírenia systému a budúcnosť projektu je opísaná v nasledujúcej kapitole.

# Kapitola 7

# Zhodnotenie výsledkov a návrh rozšírení

Výsledný systém bol predvedený budúcim užívateľom, ktorí si prácu s ním mohli vyskúšať aj osobne. Stanovené požiadavky boli splnené a výsledok uspokojivý. V závere sa ukázalo, že pre vývoj bola zvolená vhodná kombinácia technologických prostriedkov. Vyhľadanie určitého softvéru ako napríklad programu pre konverziu AutoCadových modelov na obrázky nebolo vôbec jednoduché. Bolo treba nájsť čo najlacnejšie riešenie s konzolovou podporou. Navyše väčšina programov nefungovala správnym spôsobom, až pokým sa nenašiel ten adekvátny. Požiadavky klienta na systém boli špecifické, odlišujúce sa od bežných informačných systémov. Vypracovanie celého projektu bolo pomerne časovo náročné.

Zend Framework sa osvedčil ako výhodný vývojový nástroj. Pozitívom je výsledná architektúra aplikácie. Systém je jednoducho udržiavateľný a uľachčená je aj možnosť jeho rozširovania. Na druhej strane negatívum predstavuje rýchlosť frameworku. Možné optimalizačné rozšírenie v tomto smere môže priniesť trieda `Zend_Cache`, pomocou ktorej sme schopní výrazne zrýchliť chod našej aplikácie. Uplatní sa najmä pri dotazoch na databázu, keď výsledok uložíme do pomocného prechodného súboru, z ktorého pri ďalšom požiadavku čítame a nemusíme sa znova dotazovať do databáze. To nám pri komplexnejších dotazoch môže výrazne ušetriť čas.

Čo sa týka súborového systému, vysoko efektívne rozšírenie by predstavovalo prechod na Ajaxový súborový systém. Ten je vysoko interaktívny a budí dojem, ako by sme pracovali s adresármi a súbormi priamo na našom disku, i keď všetko beží v prostredí internetu. Táto varianta je však zložitejšia v rozsahu implementácie a hlavný problém predstavuje nedostatok dostupnej literatúry a dokumentácie. Osobne som uvažoval o tejto alternatíve, ale nepodarilo sa mi vyhľadať zdroje, ktoré by ma nasmerovali v budovaní aplikácie tohoto typu. Preto je táto možnosť ponechaná ako alternatíva, ktorá môže byť uplatnená v budúcnosti.

Počas vývoja systému i po jeho dokončení sa od klienta objavujú nové požiadavky. Súčasti systému sú upravované a funkcionalita je postupne rozširovaná. Vývoj aplikácie ešte nie je ukončený. Do budúcnosti bude značne rozšírený modul firemných rozpočtov. Ten vo svojom aktuálnom stave predstavuje technické riešenie problému ako na to, ale funkcionalita je len základná. Je tomu tak vzhľadom na nedostatok dát a času zo strany klienta. Pridaný bude tiež modul pre plánovanie úloh jednotlivým projektantom v určitom časovom horizonte. Konečnosť aplikácie tohoto typu nie je nikdy ohraničená. V podstate v informačnom systéme neustále môžeme niečo vylepšovať a pridávať do neho nové prvky.

# Kapitola 8

# Záver

Predmetom tejto práce bolo vytvorenie informačného systému pre firmu zaoberajúcu sa projektovaním, autorským dozorom a konzultačnou činnosťou v oblasti vodného hospodárstva a ekologických stavieb. V úvode je čitateľ oboznámený s téoriou informačných systémov a základnými pojmami, ktoré ho uvedú do problematiky. V nasledujúcich kapitolách sú postupne opísané použité technológie a celý proces tvorby výslednej aplikácie. Požiadavky klienta sú po analýze prevedené do formy modelu, ktorý tvorí stavebný základ pre budovaný systém. Nasleduje popis implementácie jednotlivých súčastí systému a ich testovanie.

Kombinácia použitých technológií sa v závere osvedčila ako vhodná a výsledný systém spĺňa požadovanú funkcionalitu. Aplikácia nie je bezchybná. Ako to už býva na poli softvérových produktov, neustále je čo vylepšovať. Pri testovaní aplikácie bol programový kód do určitej miery refaktorizovaný (zprehladnenie zdrojového kódu – napr. spojenie viacerých tried do jednej univerzálnej).

Práca na projekte nie je ukončená. V súčasnosti sa hľadá kvalitný hosting za prijateľnú cenu na ktorý bude aplikácia vo výslednej podobe presunutá. V systéme bude rozširovaný modul rozpočtov a pridaný nový modul pre plánovanie úloh projektantom. Vyhľadávanie v systéme bude doplnené o jQuery Autocomplete, čo predstavuje automatické inteligentné dopĺňanie pojmov, ktoré vpisujeme do formulárov. Proces prechodu papierovej kancelárie na digitálnu vo forme informačného systému býva obťažny a často krát stroskotá. Preto bola snaha vytvoriť systém skutočne vylepšujúci aktuálny stav, ktorého ovládanie bude dostatočne pochopitelné pre jeho budúcich užívateľov.

V komerčnej sfére sú obecne pri realizácii informačného systému do tohoto procesu zapojení viacerí pracovníci od konzultantov, analytikov, cez vývojárov, grafikov až po testerov. Aj keď aplikácia v tejto práci nebola takého rozsiahleho charakteru, bolo treba do jej realizácie investovať určitý čas, kedže bola vytváraná jedným človekom. Z toho dôvodu som si z každej etapy či už analýzy alebo vývoja, odniesol určité vedomosti a skúsenosti. Pred začatím samotnej práce bolo mojím hlavným cieľom naučiť sa pracovať so Zend Frameworkom, ktorý som pred tým nikdy nepoužíval. Osvojenie si práce s ním a získanie všetkých nových znalostí a skúseností spojených s tvorbou, modelovaním a testovaním tohoto systému hodnotím ako hlavné prínosy pre moju osobu. Celá práca nebola vypracovávaná ako účelová školská povinnosť, ale so záujmom o danú problematiku a oblasť.

# Literatura


[1] WampServer official site. [online], [cit. 2011-05-09].
URL http://www.wampserver.com/en/

[2] Allen, R.: *Zend Framework in Action*. Manning Publications Co., 2009, iSBN 1933988320.

[3] Alshanetsky, I.: Usage statistics. [online], [cit. 2011-03-08].
URL http://phpadvent.org/2010/usage-statistics-by-ilia-alshanetsky

[4] Andrew, P.: Top 10 Javascript Frameworks. [online], [cit. 2011-05-09].
URL http://speckyboy.com/2008/04/01/top-10-javascript-frameworks-which-do-you-prefer/

[5] Asial corp.: JpGraph Manual. [online], [cit. 2011-04-15].
URL http://jpgraph.net/download/manuals/chunkhtml/

[6] Autodesk Inc.: Autodesk Freewheel. [online], [cit. 2011-04-22].
URL http://freewheel.autodesk.com/

[7] Bernard, B.: Úvod do architektúry MVC. [online], [cit. 2011-02-11].
URL http://zdrojak.root.cz/clanky/uvod-do-architektury-mvc/

[8] Böhmer, M.: *Zend Framework: programujeme webové aplikace v PHP*. Computer Press, 2010, iSBN 978-80-251-2965-4.

[9] Daněk, P.: Velký test PHP frameworků: Zend, Nette, PHP a RoR. [online], [cit. 2011-03-28].
URL www.root.cz/clanky/velky-test-php-frameworku-zend-nette-php-a-ror/

[10] Fowler, M.: *Destilované UML*. Grada, 2009. 1. vyd. 173 s. : il., iSBN 978-80-247-2062-3.

[11] Google Inc.: Google Maps API Family. [online], [cit. 2011-04-18].
URL http://code.google.com/apis/maps/

[12] Hernandez, M. J.: *Návrh databází*. Grada, 2005, iSBN 80-247-0900-7.

[13] Hodan, P.: Úvod do SQL. [online], [cit. 2011-04-7].
URL http://www.root.cz/clanky/uvod-do-sql/

[14] Hruška, T.; Křivka, Z.: *Informační systémy (IIS, PIS), Pojem informačního systému - Data - Procesy - Transakce*. Brno: FIT VUT v Brně, 2008.

[15] Janovský, D.: Jak psát web. [online], [cit. 2011-03-24].
URL http://www.jakpsatweb.cz/

[16] Kanisová, H.; Müller, M.: *UML srozumitelně*. Computer Press, a.s., 2007, iSBN 80-251-1083-4.

[17] Ľuboslav Lacko: *PHP a MySQL Hotová řešení*. CP books, 2005, iSBN 80-251-0397-8.

[18] Láslo, P.: Ajax - Úvod. [online], [cit. 2011-05-09].
URL http://programujte.com/?akce=clanek&cl=2008062101-ajax-uvod

[19] Oracle corp.: MySQL official site. [online], [cit. 2011-04-28].
URL http://www.mysql.com/

[20] jQuery Project, T.: jQuery library official site. [online], [cit. 2011-04-20].
URL http://jquery.com/

[21] Ústav informačních systémů: Objektově orientované modelování systémů [online]. [online], 2004 [cit. 2011-03-12].
URL https://www.fit.vutbr.cz/study/courses/IUS/private/oo_modelovani/

[22] The jQuery Project and jQuery Ui Team: jQuery User Interface. [online], [cit. 2011-04-14].
URL http://jqueryui.com/

[23] Ullman, L.: *PHP a MySQL: názorný průvodce tvorbou dynamických WWW stránek*. COMPUTER PRESS, 2004, iSBN 80-251-0063-4.

[24] Vymětal, D.: *Informační systémy v podnicích : teorie a praxe projektování*. Grada, 2009, iSBN 978-80-247-3046-2.

[25] Václavek, P.: *Javascript Hotová řešení*. Computer Press, 2003, iSBN 80-7226-854-6.

[26] Zendulka, J.; Bartík, V.; Šárka Květoňová: *Analýza a návrh informačních systémů AIS*. Brno: FIT VUT v Brně, 2006.

[27] Zendulka, J.; Rudolfová, I.: *Databázové systémy IDS - studijní opora*. Brno: FIT VUT v Brně, 2006.

[28] Šmíd, V.: Životní cyklus informačního systému. [online], [cit. 2011-02-27].
URL http://www.fi.muni.cz/~smid/mis-zivcyk.htm

# Dodatek A

# Obsah CD

V obsahu priloženého CD je možné nájsť zdrojové kódy aplikácie, základné knižnice pre Zend Framework, technickú správu vo formáte pdf, latexové zdrojové súbory, programovú dokumentáciu, manuál pre užívateľov, súbor README pre základný popis a súbor INSTALL, ktorý popisuje konfiguráciu a spustenie aplikácie. Adresárová štruktúra umiestnená na CD:

- **BP-pdf** – technická správa vo formáte pdf.
- **BP-zdrojove** – latexové zdrojové súbory technickej správy.
- **Dokumentacia**

  **Html-doc** – programová dokumentácia (index.html).

  **Manual** – manuál pre užívateľov systému.
- **Hydroeco** – adresár so samotnou aplikáciou a všetkými zdrojovými kódmi, navigácia medzi aplikačnými adresármi sa nachádza v súbore Readme.
- **Db** – adresár obsahujúci skript pre vytvorenie a naplnenie databáze.
- **ZF-library** – štandartné knižnice Zend Frameworku.
- **Install** – popis konfiurácie a spustenia aplikácie.
- **Readme** – základný popis projektu, licencie a navigácie.