TECHNICKÁ UNIVERZITA V LIBERCI
Ekonomická fakulta

# WEBOVÁ ŘEŠENÍ VE STAVEBNICTVÍ

## Bakalářská práce

*Studijní program:* B6209 – Systémové inženýrství a informatika
*Studijní obor:* 6209R021 – Manažerská informatika

*Autor práce:* **Petr Vaněk**
*Vedoucí práce:* Mgr. Tomáš Žižka

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Economics

# WEB SOLUTIONS IN CIVIL ENGINEERING

## Bachelor thesis

*Study programme:* B6209 – System Engineering and Informatics
*Study branch:* 6209R021 – Managerial Informatics

*Author:* **Petr Vaněk**
*Supervisor:* Mgr. Tomáš Žižka

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
**Ekonomická fakulta**
Akademický rok: **2013/2014**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Petr Vaněk** |
| Osobní číslo: | **E11000520** |
| Studijní program: | **B6209 Systémové inženýrství a informatika** |
| Studijní obor: | **Manažerská informatika** |
| Název tématu: | **Webová řešení ve stavebnictví** |
| Zadávající katedra: | **Katedra informatiky** |

Zásady pro vypracování:

1. Webové aplikace jako moderní nástroj informačních služeb
2. Přístupy k návrhu a nástroje pro tvorbu webových aplikací
3. Analýza prostředí a požadavků na aplikaci pro odhad stavebních nákladů
4. Návrh a realizace webové aplikace
5. Zhodnocení navrženého řešení a perspektivní možnosti

Rozsah grafických prací:

Rozsah pracovní zprávy: **30 normostran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

**LURIG, Mario. PHP reference: beginner to intermediate PHP5. 1st ed. Lulu: Raleigh, N.C., 2008. ISBN 978-143-5715-905.**
**KOFLER, Michael a Bernd ÖGGL. PHP 5 a MySQL 5: průvodce webového programátora. 1. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1813-9.**
**LAVIN, Peter. PHP - objektově orientované: koncepty, techniky a kód. 1. vyd. Praha: Grada, 2009. ISBN 978-80-247-2137-8.**
**DARIE, Cristian, Bogdan BRINZAREA, Filip CHERECHES-TOSA a Mihai BUCICA. AJAX a PHP: tvoříme interaktivní webové aplikace profesionálně. 1. vyd. Brno: Zoner Press, 2006. ISBN 80-868-1547-1.**
**GUTMANS, Andi, Stig Sather BAKKEN a Derick RETHANS. Mistrovství v PHP 5. vyd. Brno: CP Books, 2005. ISBN 80-251-0799-X.**
**Elektronická databáze článků ProQuest (knihovna.tul.cz).**

Vedoucí bakalářské práce: **Mgr. Tomáš Žižka**
Katedra informatiky

Konzultant bakalářské práce: **Ing. František Benč, Ph.D.**
ředitel společnosti Aspe

Datum zadání bakalářské práce: **30. října 2013**

Termín odevzdání bakalářské práce: **7. května 2014**

doc. Ing. Miroslav Žižka, Ph.D.
děkan

Technická univerzita v Liberci
Ekonomická fakulta

L.S.

doc. Ing. Jan Skrbek, Dr.
vedoucí katedry

V Liberci dne 30. října 2013

# Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

# Anotace

Cílem této práce bylo popsat využití webových aplikací jako nástroje moderních informačních služeb a vytvoření takové aplikace pro celoživotní ocenění budoucího stavebního díla především pro potřeby investora. V úvodní části je seznámení s problematikou stavebnictví a webových aplikací. V další části je seznámení s webovými aplikacemi, s technologiemi, které využívají a jejich přehled na českém trhu. Následuje analýza požadavků na samotnou aplikaci a popis implementace. Celá aplikace byla vytvořena pomocí značkovacího jazyka HTML, skriptovacích jazyků PHP a JavaScript za pomoci českého frameworku Nette a využívá relační databáze MySQL.

# Klíčová slova

Stavební software, PHP, HTML, MySQL, Nette, JavaScript, AJAX

# Annotation

The aim of this bachelor thesis was to describe uses of web applications as tools of modern information services and to create such an application for whole life costs of planned projects for the needs of an investor. First part of this work is an introduction into civil engineering and web applications. The next part consists of introduction into web applications with technologies they use and their summary on czech market. Analysis of requirements on the application itself and its implementation follows. The whole application was created with the help of the HyperText Markup Language, the skripting languages PHP and JavaScript with help of czech framework Nette and it is using relational MySQL database.

# Keywords

Civil engineering software, PHP, HTML, MySQL, Nette, JavaScript, AJAX

# Obsah

# Seznam ilustrací

# Seznam tabulek

# Seznam použitých zkratek

| | |
|---|---|
| AJAJ | Asynchronous JavaScript and JSON |
| AJAX | Asynchronous JavaScript and XML |
| CSS | Cascade Style Sheets |
| DFD | Data Flow Diagram |
| DSP | Dokumentace pro stavební povolení |
| DÚR | Dokumentace pro územní rozhodnutí |
| ERA | Entity Relation Atribute |
| FSD | Function Structure Diagram |
| FTP | File Transfer Protocol |
| HTML | HyperText Markup Language |
| HTTP | HyperText Transfer Protocol |
| IDE | Integrated Development Environment |
| JSON | JavaScript Object Notation |
| MVC | Model view controler |
| MVP | Model view presenter |
| OID | Object ID |
| P3A | Princip tří architektur |
| PHP | Personal Homepage |
| ŘSD | Ředitelství silnic a dálnic |
| SQL | Structured Query Language |
| UML | Unified Modeling Language |
| XML | Extensible Markup Language |
| ZDS | Zadávací dokumentace stavby |
| ZP | Záměr projektu |

# Úvod

Protože stavební zakázky a projekty jsou v dnešní době stále složitější a obsahují více a více stavebních dílů díky novým technologiím, je pro investora stále složitější odhadnout cenu takového projektu. Dříve se pro ocenění staveb používaly různé oficiální i neoficiální ceníky a třídníky. Díky tomu jak se dnes rychle všechny technologie vyvíjí a doba si žádá mnohem rychlejší rozhodování na straně investora, je téměř nemožné tyto ceníky, které se většinou vedly jen ve formě nějaké tabulky v tabulkovém procesoru, efektivně využívat. Naštěstí se rozvíjejí i informační technologie, a tak se tyto ceníky stávají součástí různých desktopových aplikací. Ovšem ani to není ideální, protože tyto ceníky a jejich položky je třeba každý rok aktualizovat. Díky rozvoji internetu se však objevila nová možnost. Jedná se o webové aplikace, kde jsou aktualizace velice jednoduché, protože klienti využívají aplikaci, která běží na serveru, a tak mají vždy přístup k nejnovější verzi. Navíc se nemusí starat o zálohování nebo bezpečnost svých dat, neboť to je zajištěno společností, která tuto aplikaci poskytuje.

Cílem této práce je zhodnotit využitelnost webových aplikací jako moderního nástroje informačních služeb pro potřeby stavebnictví. Nejprve je třeba prozkoumat, čeho jsou moderní aplikace schopny a jakých prostředků využívají. Tyto prostředky, budou pak ještě blíže přiblíženy a popsány. Budou zde také popsány možné přístupy k návrhu aplikací. V poslední části bude práce zaměřena na vývoj webové aplikace pro oceňování stavebních projektů pro investory a projektanty s názvem EstiCon.

Díky své praxi ve firmě Valbek, spol. s r.o., jsem se právě k vývoji takové aplikace dostal. Společnost Valbek byla založena v roce 1990 se zaměřením především na projektování silnic a mostů. Postupně se zaměření Valbeku rozšířilo také na tunely, pozemní a vodohospodářské stavby, geodetické práce, ekologické studie, vývoj a prodej stavebního softwaru a poradenskou činnost.

Původní aplikace, která byla vyvíjena ve společnosti IBR Consulting s.r.o., byla napsána v programovacích jazycích PHP, JavaScript a běžela na nerelační databázi. Rozšíření této aplikace se ukázalo jako velice problematické, a tak bylo rozhodnuto, že pro IBR Consulting bude vyvinuta nová aplikace, postavena na PHP frameworku Nette pro snadnější znovupoužitelnost zdrojového kódu a větší bezpečnost aplikace. Nette zároveň

velice urychlí tvorbu základních prvků webové aplikace, jako jsou formuláře. Tento framework je postaven na modelu Model View Presenter (MVP), což povede k ještě větší přehlednosti zdrojového kódu. Zároveň se přešlo na relační databázi.

O vzhled celé aplikace se postarají kaskádové styly CSS a značkovací jazyk HTML s využitím šablonovacího systému Latte, který je součástí Nette.

# 1 Rešerše – webové aplikace

V následující kapitole bude rozebráno, proč jsou webové aplikace stále hojněji využívány v informačních službách a přehled webových aplikací ve stavebnictví na českém trhu.

## 1.1 Webové aplikace v moderních informačních službách

Webové aplikace jsou aplikace, které ke své funkci potřebují pouze webový prohlížeč, který slouží jako samotný klient a internet nebo intranet jako prostředek pro komunikaci se serverem, na rozdíl od aplikací typu server-klient, které potřebují nainstalovaného klienta na každém zařízení, kde chceme aplikaci využívat. Velkou nevýhodou těchto aplikací jsou také aktualizace. Pokud se aktualizuje serverová část tak to většinou vyžaduje též aktualizaci klientské části a to zpravidla na stovkách pracovních stanic, což je velice nákladné jak finančně, tak také časově. I aktualizace samotné klientské části je stejně problematická a nákladná. Další velkou nevýhodou pevných klientů je kompatibilita na různých operačních systémech a na různých zařízeních. Dnes chtějí mít uživatelé své aplikace jako je třeba email dostupné všude. Na chytrém telefonu, laptopu nebo třeba tabletu a toho je téměř nemožné docílit s pevným klientem, pokud nejsme ochotni na zajištění této kompatibility investovat nemalé prostředky. Oproti tomu, webové prohlížeče jsou součástí téměř všech těchto zařízení a zobrazení samotných webových stránek na různých zařízeních je řešeno většinou přes kaskádové styly, které nejsou závislé na samotném zařízení nebo operačním systému, ale na webovém prohlížeči. Webová aplikace funguje totiž tak, že server zpracuje uživatelské požadavky a na jejich základě vygeneruje klasickou webovou stránku, která se uživateli zobrazí v prohlížeči. Samotná aplikace totiž běží pouze na serveru a to je asi nejsilnější ale zároveň i nejslabší stránkou webových aplikací. Pokud chce vývojář aplikaci aktualizovat, aktualizuje ji pouze jednou na serveru, kde daná aplikace běží a v tu chvíli mají všichni uživatelé nejnovější verzi. Pokud ovšem server přestane z nějakých důvodů fungovat, aplikace přestane být dostupná pro uživatele.

Dříve byla velkou nevýhodou webových aplikací především rychlost internetu a statický obsah webových stránek. Pokud chtěl uživatel otevřít jen malý formulář s několika vstupy, musela se mu stejně vždy načíst celá nová stránka, což práci velice zpomalovalo. Díky moderním technologiím, jako je například JavaScript, Asynchronous JavaScript and XML

(AJAX) a Asynchronous JavaScript and JSON (AJAJ)[1], které umožňují načítat data přímo ze serveru pomocí skriptů na straně uživatele, aniž by se musela obnovovat celá webová stránka, však nevýhoda mizí a moderní webová aplikace tak může vypadat a chovat se stejně nebo i lépe než pevný klient.[2] Tato vlastnost by se měla ještě posílit větší podporou značkovacího jazyka HTML5 ze strany prohlížečů.

Pro většinu firem je mnohem jednodušší také zavést novou aplikaci do provozu. Technici nemusí obcházet každou stanici, na které má aplikace fungovat a zde ji instalovat. Stačí, když ji naimplementují na server, který se stará o firemní intranet a aplikace bude okamžitě dostupná pro všechny zaměstnance, kterým je určena.

V dnešní době, kdy je internet dostupný téměř všude a každý větší podnik má svůj vlastní intranet, se webové aplikace stávají velice silným nástrojem informačních služeb. Důkazem je třeba společnost Valbek, kde jsem se setkal se širokou paletou webových aplikací, které využívá ve svých informačních službách. Jedná se třeba o emailového webového klienta nebo manažerský informační portál, který podává informace pro rozhodování nejvyššímu vedení podniku. Na intranetu se také nachází webová aplikace na rezervování zasedacích místností nebo aplikace na výkaz docházky. I na Technické univerzitě v Liberci se setkáváme s emailovým webovým klientem, s aplikací na rezervaci obědů nebo s informačním systémem pro studenty STAG.

[1] LOUDON, Kyle. *Developing large web applications*. 1st ed. Sebastopol, Calif.: O'Reilly/Yahoo! Press, 2010. ISBN 05-968-0302-8.
[2] MACCAW, Alex. *JavaScript web applications*. 1st ed. Sebastopol, CA: O'Reilly, 2011. ISBN 978-144-9303-518.

## 1.2 Webové aplikace ve stavebnictví

Většina webových aplikací, které se využívají ve stavebnictví, je stejná jako v jiných odvětvích. Jedná se třeba o emailové klienty na internetu nebo služby pro podporu nejvyššího vedení. Nachází se zde však také specifické aplikace. Jde především o stavební software, který pomáhá s projektováním a oceňováním nových staveb.

V současné době se na českém trhu nachází široká paleta stavebních softwarů, bohužel téměř u všech se jedná pouze o pevného klienta. Webové aplikace, zabývající se problematikou stavebnictví jsou pouze tři. Jedná se o Buildpass, Aspe – Online a EstiRoad.

### 1.2.1 Buildpass

Tato webová aplikace byla vyvinuta na Katedře ekonomiky a řízení ve stavebnictví na Fakultě stavební na ČVUT v Praze. Je zaměřena především na plánování obnovy a údržby již realizovaných staveb. Cílovou skupinou této aplikace jsou především majitelé těchto objektů.[3] Aplikace má také velice dobrou dokumentaci, která obsahuje i teorii údržby, obnovy a celoživotních nákladů. Pro investora není tato aplikace příliš vhodná, protože vyžaduje velice podrobnou znalost konstrukce a stavebních dílů stavebního objektu, které nejsou v rané fázi projektu ještě známy nebo nejsou dostatečně podrobné.

### 1.2.2 Aspe Online

Aspe Online je webovou nadstavbou pevného klienta. Tento program je stavební softwarový systém vyvíjený firmou Valbek, spol. s r.o., který je zaměřen především na stavební část projektu. Dají se v něm sestavit rozpočty připravované stavby, pomáhá sledovat náklady v průběhu stavby a na konci podá ucelený přehled o celém projektu. Zde však jeho činnost končí. Pro investora je jistě důležité vědět, kolik bude stát samotná stavba, ale neméně důležité je také vědět, kolik bude stát obnova a údržba této stavby.

Mezi přednosti této aplikace patří ceníky, které se každoročně aktualizují, stabilní vývojový tým a také import a export do formátu XC4, který je oficiálním formátem pro komunikaci ve stavebnictví v České Republice.

[3] ANNON. Buildpass – obnova a údržba objektů. [online]. [cit. 2014-02-8]. Dostupné z: http://www.buildpass.eu/

### 1.2.3 EstiRoad

EstiRoad je webová aplikace vyvíjená společností IBR Consulting, s.r.o. a je zaměřena, stejně jako Aspe, na plánovací část projektu. Ve svých kalkulacích ovšem také nezohledňuje obnovu a údržbu jednotlivých stavebních dílů, a tak není pro investora zcela vhodná.

Předností EstiRoadu je především přehledné zpracování ceníků a jejich snadné využití při tvorbě stavebního projektu. Obsahuje též kalkulační vzorce pro jednotlivé stavební položky a zohledňuje složku rizik pro různé varianty projektu. Stejně jako Aspe umí import z XC4 formátu.

# 2 Přístupy k návrhu

Při navrhování informačních systémů a programů se využívají především dva způsoby analýzy a návrhu. Jedná se o strukturovaný a objektový přístup. Oba jsou založeny na základních principech, kterými jsou modelování a abstrakce. Oba přístupy také využívají princip tří architektur (P3A), který dělí návrh systému na tři úrovně. První úroveň je konceptuální, která se snaží modelovat realitu a je nezávislá na technologické nebo implementační úrovni. Druhá úroveň je technologická. Ta vychází z konceptuální úrovně a rozpracovává ji z technologického hlediska. Řeší, jak bude systém v dané technologii realizován. Poslední je úroveň implementační nebo také fyzická. Je závislá na předchozích dvou úrovních a určuje, čím bude technologické řešení realizováno. Jedná se třeba o volbu konkrétního vývojového prostředí, jazyka nebo architektury, které jsou stanovené v technologickém modelu. Hlavním rozdílem mezi strukturovaným a objektovým přístupem je v tom, že strukturovaný přístup řeší data a procesy odděleně a objektový přístup je bere jako celek.[4]

## 2.1 Strukturovaný přístup

Strukturovaná analýza se dělí na dvě větve, kde každá z těchto větví slouží k analýze informačního systému z jiného hlediska. Funkční větev analyzuje především interakci jednotlivých částí systému mezi sebou a okolím. Pro tuto analýzu využívá Data Flow Diagram (DFD), Function Structure Diagram (FSD), Diagram stavů a přechodů nebo Model řízení. Datová větev se naopak zabývá systémem z hlediska uložení dat v databázi. Nejčastěji se k tomu využívá Entity Relation Atribute (ERA) diagram.[5]

## 2.2 Objektový přístup

Objektový přístup vychází ze strukturovaného přístupu, ale spojuje funkční a datový model do jednoho objektového modelu. Cílem je především věrnější vyjádření reality pomocí

[4] ZÁDOVÁ, Vladimíra. Konceptuální modelování v návrhu IS/ICT. [online]. [cit. 2014-01-21]. Dostupné z: http://multiedu.tul.cz/~vladimira.zadova/multiedu/DATABAZOVE_SYSTEMY/DBS_koncept_do_PDF

[5] ZÁDOVÁ, Vladimíra. Návrh IS: Strukturovaný přístup. [online]. [cit. 2014-01-21]. Dostupné z: http://multiedu.tul.cz/~vladimira.zadova/multiedu/informacni_systemy_III/IS3__09_10___struktur__pristup.pdf

objektů, které mají jak své vlastnosti, tak také svoje funkce. Pro objektový přístup je také charakteristické objektové paradigma. Tímto paradigmatem je zapouzdření, dědičnost, polymorfismus a Object ID (OID). Zapouzdření řeší přístup k funkcím a vlastnostem určité třídy, které dále upřesňují modifikátory private, protected a public. Dědičnost je vlastně uplatnění generalizace. Třídy od sebe mohou dědit vlastnosti a funkce. Třída, od které je děděno se nazývá předek a třída, která dědí, se nazývá potomek. Na dědičnost navazuje polymorfismus, který umožňuje potomkovi zděděnou funkci předefinovat. OID se pak stará o unikátnost každého objektu. Je jedinečné, neměnné a po skončení existence daného objektu nebude už nikdy znovu přiřazeno. Objektově orientované modely mají jediný standard a tím je Unified Modeling Language (UML). Ten definuje všechny diagramy a modely, které se v objektovém návrhu využívají. Mezi nejčastěji využívané patří diagram tříd, Use Case Diagram, diagram aktivit a diagram stavů a přechodů.[6]

[6] ZÁDOVÁ, Vladimíra. Objektový návrh IS. [online]. [cit. 2014-01-21]. Dostupné z: http://multiedu.tul.cz/~vladimira.zadova/multiedu/informacni_systemy_III/IS3__10_11__objekt__pristup.pdf

# 3 Agilní vývoj

Při tvorbě softwaru se využívají v zásadě dva způsoby vývoje. Jeden z nich je rigorózní (někdy také rigidní), z něhož nejznámější technikou je vodopádový přístup, a druhý je agilní.

Při vodopádovém vývoji softwaru se postupně vykonávají jednotlivé části projektu a vrátit se jde vždy jen k předchozí části. Pokud zákazník bude chtít změnit výslednou podobu produktu, není to již možné. Výhodou tohoto přístupu jsou jasně stanovené finanční a právní stránky projektu a také je vhodnější na velké projekty. Zákazník se většinou na vývoji vůbec nepodílí.

Pro agilní vývoj je naopak charakteristické to, že požadavky je možné měnit i během vývoje a zákazník se na výsledku přímo podílí. Základem agilního vývoje je agilní manifest[7], který vznikl přímo ze zkušeností samotných programátorů. Klade důraz na flexibilitu, funkční software a především na spokojeného zákazníka, kterému má pomáhat zvyšovat svoji konkurenceschopnost. V dnešních rychle se měnících podmínkách na trhu je důležité, aby výsledný produkt reflektoval potřeby zákazníka, k čemuž je agilní vývoj ideální. Jeho nevýhodou je špatná použitelnost na velkých projektech, protože základem je komunikace mezi vývojáři, cože je ve velkých týmech obtížné.

Agilní vývoj softwaru probíhá po iteracích, na jejichž konci se vždy nachází funkční verze. V nulté iteraci se udělá funkční základ, z kterého budou další iterace vycházet. Každá iterace má pak tři fáze. V první fázi se analyzují změny, v další fázi se tyto změny implementují a v poslední fázi se hotový produkt předvede zákazníkovi. Pokud není zákazník spokojen, následuje další iterace, a naopak pokud je spokojen, ukončí se vývoj a následuje údržba, případně po čase další rozvoj.[8]

[7] BECK, Kent aj. Manifesto for Agile Software Development. [online]. [cit. 2014-01-21]. Dostupné z: http://agilemanifesto.org/

[8] KNESL, Jiří. Agilní vývoj: Úvod. [online]. [cit. 2014-01-21]. Dostupné z: http://www.zdrojak.cz/clanky/agilni-vyvoj-uvod/

# 4 Nástroje pro tvorbu webových aplikací

Pro tvorbu klasických desktopových aplikací se většinou využívá pouze jeden programovací jazyk a jedno vývojové prostředí. Vývoj webových aplikací ovšem není tak jednoduchý, a pokud má být aplikace uživatelsky přívětivá, je třeba využít hned několika programovacích jazyků. Aby měl programátor alespoň trochu usnadněnou práci, každý z těchto jazyků má širokou nabídku různých knihoven a frameworků, které zpřehledňují a zjednodušují syntaxi. V následující kapitole budou rozebrány programovací jazyky, které se ve vývoji webové aplikace EstiCon používají, vývojová prostředí a v neposlední řadě také frameworky a knihovny.

## 4.1 Vývojové prostředí

Téměř žádný programátor se v dnešní době neobejde bez nějakého vývojového prostředí neboli IDE (Integrated Development Environment). Tyto programy mají práci programátora především zrychlit a usnadnit pomocí různých funkcí jako je našeptávač, automatická kontrola a formátování zdrojového kódu nebo zvýraznění zdrojového kódu v různých jazycích. Tyto funkce jsou obzvláště užitečné v programování webových aplikací. Ty je možné programovat v podstatě v každém textovém editoru, ale špatně zvolené IDE může práci značně zpomalovat.

### 4.1.1 NetBeans

Jedná se o open source projekt, o jehož vývoj se stará především česká pobočka Sun Microsystems.[9] Mezi největší výhody tohoto IDE patří výborný našeptávač, který prohledává všechny soubory v daném projektu, a je tak schopen vývojáři našeptávat jména všech tříd, funkcí a proměnných, které v projektu používá. Další výhodou je kontrola zdrojového kódu, kdy IDE samotné pozná, zda programátor zrovna píše v HTML, PHP nebo JavaScriptu a je schopné ho samo upozorňovat na chyby v syntaxi. V nejnovější verzi má i podporu Nette frameworku. Standardem je pak otevření více souborů současně nebo zvýraznění zdrojového kódu v různých jazycích. I přes aktivní vývoj mají NetBeans stále

[9] STROBL, Roman. Happy birthday NetBeans – Interview with Jaroslav "Yarda" Tulach. [online]. [cit. 2014-01-21]. Dostupné z: https://netbeans.org/community/articles/interviews/yarda-tulach.html

velice špatnou podporu File Transfer Protocolu (FTP), a protože při vývoji aplikace EstiCon jsou všechny soubory uloženy přímo na serveru, práce se tak kvůli potřebě manuálního ukládání na server velice zpomaluje. Tato potřeba vyplývá z toho, že aplikace je vyvíjena v týmu, v němž někteří členové pracují z domova, ale zároveň je tým natolik malý, že není třeba používat systém pro sdílení a správu zdrojových kódů. Také je díky svému našeptávači velice náročný na systémové zdroje.

### 4.1.2 Notepad++

Notepad++ je freewarový textový editor, který vyvíjí Don Ho.[10] Stejně jako NetBeans umí i Notepad++ zvýrazňovat části zdrojového kódu, podle toho v jakém jazyce jsou napsány, otevřít více souborů současně a zvýrazňovat párové tagy, příkazy a závorky. Bohužel neumí kontrolovat samotnou syntaxi a jeho našeptávač není zdaleka tak pokročilý jako v NetBeans. Oproti NetBeans má však velice dobrou podporu FTP a jeho náročnost na výpočetní výkon je také řádově nižší.

*Tabulka 1: Srovnání vývojových prostředí NetBeans a Notepad++*

| **NetBeans** | **Notepad++** |
|---|---|
| + našeptávač<br>+ kontrola kódu<br>+ automatické formátování<br>+ označení kódu v různých jazycích<br>+ práce s více soubory<br>+ podpora Nette | + označení kódu v různých jazycích<br>+ práce s více soubory<br>+ zvýrazňování párových tagů, příkazů a závorek<br>+ podpora FTP<br>+ nenáročnost na hardwarové prostředky |
| - chybí podpora FTP<br>- náročnost | - našeptávač<br>- automatické formátování |

Zdroj: vlastní

V tabulce **Chyba! Nenalezen zdroj odkazů.** je vidět, že NetBeans při lokálním vývoji usnadňuje vývojáři v mnohých ohledech práci. Vývoj však probíhá v týmu, a někdy je také nutná práce z domova, a tak je vývoj na lokálním serveru nemožný. I samotná náročnost na zdroje je u NetBeans řádově vyšší a občas může způsobit i nestabilitu techniky, na které

[10] HO, Don. Notepad++ Author. [online]. [cit. 2014-01-21]. Dostupné z: http://notepad-plus-plus.org/contributors/author.html

vývoj probíhá. Tyto dva důvody vedly k tomu, že bylo vybráno vývojové prostředí Notepad++.

### 4.1.3 MySQL Workbench

MySQL Workbench je volně šiřitelný software pro návrh a správu MySQL databáze. Základní a nejdůležitější funkcí je tvorba ERA diagramu ve standartu UML, který je pro modelování velkých relačních databází nutností. Zvládá také Forward a Reverse Engineering, což znamená, že je schopen z ERA diagramu vygenerovat SQL příkaz pro tvorbu fyzické databáze a naopak je schopen z fyzické databáze vytvořit model pomocí ERA diagramu, což je velice praktické, pokud je vývojář nucen pracovat s databází, na jejímž vývoji se nepodílel.[11] Pro lepší orientaci se dá model dělit i na více dílčích ERA diagramů a jednotlivé entity se dají v těchto různých diagramech využívat pomocí systému drag and drop a jejich vazby jsou při tom zachovány. Dají se zde také tvořit jednotlivé SQL dotazy, které umí automaticky formátovat pro lepší čitelnost nebo exportovat jednotlivé diagramy do několika různých formátů, například PDF nebo PNG, které jsou čitelné téměř na všech počítačích. To, že je toto vývojové prostředí dostupné zdarma a velice usnadňuje tvorbu fyzické databáze díky možnosti vygenerování SQL kódu přímo z modelu, jsou hlavní důvody, proč bylo toto vývojové prostředí vybráno.

## 4.2 Programovací jazyky

Pro tvorbu webových aplikací se využívá velké množství programovacích jazyků. Patří mezi ně skriptovací jazyky jako je třeba PHP a JavaScript, které se starají o logiku programu. Dále se využívají dotazovací jazyky pro práci s databázemi a daty obecně. Nejznámější dotazovací jazyk je asi Structured Query Language (SQL). Posledním druhem programovacího jazyka je kódovací nebo také značkovací jazyk. Ten má na starost samotné vykreslení stránky. Nejrozšířenějšími a nejužívanějšími na webu jsou HyperText Markup Language (HTML) a Cascading Style Sheets (CSS). Až spojením všech těchto tří skupin jazyků, kde se každý stará o jinou vrstvu aplikace, jsme schopni takovéto aplikace tvořit.

[11] ANNON. MySQL Workbench 6.0: Visual Database Design. [online]. [cit. 2014-01-21]. Dostupné z: http://www.mysql.com/products/workbench/design/

### 4.2.1 HyperText Markup Language - HTML

Hlavním autorem HTML byl vědec ze švýcarského výzkumného institutu CERN Tim Berners-Lee a měl primárně sloužit ke sdílení informací mezi vědci.[12]

Jedná se o značkovací jazyk, a tak jsou jeho základem značky (tagy). Ty bývají většinou párové, i když to není pravidlem. Tyto značky pak umí webový prohlížeč zobrazit jako různé objekty a doplnit k nim obsah, který je uveden mezi těmito značkami, pokud jsou párové, nebo přímo z hodnoty atributu značky, pokud je nepárová.

V současnosti se používá HTML ve verzi 5, který přidává nové grafické možnosti a komponenty.

### 4.2.2 PHP

Jedná se o interpretovaný programovací jazyk, který vytvořil v roce 1994 Rasmus Lerdorf a původně se jednalo jen o sadu skriptů, která sloužila pro sledování přístupu na jeho osobní webovou stránku. Tuto sadu pojmenoval Personal Home Page Tools, z čehož vznikl název PHP.[13]

Tento jazyk funguje tak, že zdrojový kód se vykoná přímo na serveru, na kterém aplikace běží. Poté z něj vygeneruje HTML kód, který odešle do uživatelova prohlížeče, kde se HTML kód vykreslí. Ve chvíli, kdy se vygeneruje, jsou všechny proměnné a data, kromě superglobálních proměnných SESSION a COOKIE, na straně serveru zahozeny, a pokud je třeba s těmito daty dále pracovat, je třeba tato data uložit do databáze, do superglobální proměnné nebo odeslat zpět do uživatelova prohlížeče. Je také nutné, aby server, na kterém je aplikace provozována, podporoval PHP ve verzi, pro kterou je napsána. Díky tomu, že kód se vykoná na výkonném serveru a uživatel přijme jen dokument v HTML, neovlivňuje jeho rychlost hardware uživatele.

PHP bývá často programátory kritizováno za svoji nejednoznačnou syntaxi, protože na rozdíl od většiny ostatních jazyků bylo vyvíjeno dobrovolníky organicky. Nebyla jasně

[12] ANNON. The history of HTML. [online]. [cit. 2014-02-8]. Dostupné z: http://inventors.about.com/od/computersoftware/a/html.htm

[13] ANNON. History of PHP. [online]. [cit. 2014-02-8]. Dostupné z: http://www.php.net/manual/en/history.php.php

stanovená pravidla a omezení. Také nemá žádné vhodné vývojové prostředí, kde by se daly odstraňovat chyby v kódu.

### 4.2.3 JavaScript

JavaScript byl vytvořen v roce 1995 zaměstnancem firmy Netscape Brendanem Eichem. Původní název byl Mocha, ale po získání licence od Sun se, především z marketingových důvodů, přejmenoval na JavaScript[14]. Jazyk má podobnou syntaxi jako Java nebo jazyk C.

Jedná se o objektově orientovaný, interpretovaný jazyk, který se na rozdíl od jazyka PHP vykonává přímo v prohlížeči na straně klienta. To přímo ovlivňuje rychlost vykonávaného programu v závislosti na hardwarovém vybavení uživatele. Kód se zapisuje přímo do HTML stránky, většinou mezi párové tagy <script></script>, a funguje, pouze pokud je v prohlížeči povolen.

### 4.2.4 SQL

SQL (Structured Query Language) bylo původně vyvinuto firmou IBM pod názvem SEQUEL (Structured English Query Language) na základě modelu relačních databází Dr. E. F. Codda v 70. letech 20. století. V roce 1979 společnost Relational Software (nyní Oracle) uvedla první komerční implementaci jazyka SQL.[15]

Je to dotazovací jazyk uzpůsobený pro práci s relačními databázemi a jeho syntaxe vychází z klasické angličtiny, aby byl jazyk co nejsrozumitelnější pro uživatele.

## 4.3 UML

Unified Modeling Language (UML) je jediným standardem v oblasti objektového modelování. Jedná se o grafický jazyk, který obsahuje standardizované metody pro návrh,

[14] ANNON. A short history of Javascript. [online]. [cit. 2014-02-8]. Dostupné z: http://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

[15] ANNON. History of SQL. [online]. [cit. 2014-02-8]. Dostupné z: http://www.stanford.edu/dept/itss/docs/oracle/10g/server.101/b10759/intro001.htm

analýzu a implementaci v softwarovém inženýrství. Obsahuje různé diagramy pro usnadnění těchto činností během vývoje.[16]

## 4.4 Frameworky

Při standardním vytváření webové aplikace jsou činnosti, které vývojář vykonává opakovaně, a nepřinášejí téměř žádnou přidanou hodnotu. Nejlepším příkladem takové činnosti je tvorba formulářů. Základní vytvoření formuláře, třeba na vytvoření registrace, která bude mít jen pár standardních vstupů a tlačítko odeslání zabere pár řádků kódu a minimum času. Ke každému vstupu je však nutné připojit také kód, který ošetřuje validaci formuláře a zabrání uživateli odeslat špatná data, kterými může být jen špatně zadaný email bez zavináče, ale také naformátovaný SQL dotaz, který může poškodit nebo získat citlivá data z databáze. Tato validace se musí vykonat jak na straně uživatele pomocí JavaScriptu, pokud je povolen, tak také na straně serveru, třeba v PHP. Takováto validace může obsahovat i několik desítek či stovek řádků kódu, a i tak se může stát, že vývojář nějakou chybu opomene ošetřit. Přitom takovéto validace jsou třeba provádět na každém jednotlivém formuláři.

Právě z tohoto důvodu vznikly frameworky. Dobré frameworky obsahují běžné funkce, které je třeba vykonávat několikanásobně v celém programu, jako je třeba právě zabezpečení aplikace, a tím značně usnadňují vývojáři práci a zvyšují jeho produktivitu.

### 4.4.1 Nette Framework

Jedná se o český PHP framework vyvíjený skupinou Nette Foundation, založenou v roce 2009 Davidem Grudlem za účelem vývoje tohoto frameworku. Mezi jeho hlavní přednosti patří bezpečnost, rychlost a vlastní debugovací nástroj (laděnka). Je postaven na modelu Model-View-Present (MVP), který odděluje model pro práci s daty, view pro vykreslení a presenter, který dvě předchozí části spojuje, protože model a view o sobě vzájemně ani nevědí, a stará se o aplikační logiku. Má také vlastní šablonovací systém Latte, který značné zjednodušuje práci pomocí latte maker. Další velkou výhodou je spolupráce

[16] ZÁDOVÁ, Vladimíra. Objektový návrh IS. [online]. [cit. 2014-01-21]. Dostupné z: http://multiedu.tul.cz/~vladimira.zadova/multiedu/informacni_systemy_III/IS3__10_11__objekt__pristup.pdf

s knihovnou pro práci s databázemi dibi, která je také vyvíjena Nette Foundation. O to, aby byl web dynamický, se starají snippety, které umožňují překreslení jen určité části stránky pomocí technologie AJAX, která je v Nette výborně implementována a jednoduše se s ní pracuje.[17]

Nette má velice aktivní českou komunitu a i sami tvůrci frameworku se často zapojují do řešení problému na oficiálním fóru a IRC kanálu. Největší nevýhodou frameworku se tak stává nekompletní dokumentace. V aplikaci je použita verze Nette 2.0.12.

## 4.5 JavaScriptové knihovny

Stejně jako frameworky, tak také knihovny mají za úkol především usnadnit programátorovy práci. Pomocí JavaScriptu je možné na stránkách vykreslit takřka cokoliv, ale občas je samotné programování časové velice náročné, protože elementární syntaxe samotného jazyka je velice zdlouhavá a nepraktická. Proto existují různé knihovny například na vykreslení grafů nebo zpracování HTML tabulek.

### 4.5.1 jQuery

Jedná se o knihovnu, která usnadňuje práci s HTML prvky, jako jsou různá textová pole a tlačítka a především zjednodušuje syntaxi. Povoluje jejich dynamickou změnu za běhu aplikace, stará se o události (eventy) spojené s jednotlivými prvky a dále rozšiřuje možnosti využití AJAXu v programu. Umožňuje také využití jQuery UI pro tvorbu visuálních komponent, které HTML neumí. Jedná se například o modální okna nebo slidery. EstiCon využívá verzi v2.0.3.

### 4.5.2 DataTables

Tato knihovna slouží pro práci s tabulkami. K obyčejné tabulce v HTML přidá možnost třídění podle zvolených sloupečků nebo full textové vyhledávání v datech tabulky. Umožňuje také jednoduší formátování tabulky a data se do ní dají přidávat i dynamicky, bez nutnosti znovu načíst celou stránku. Podporuje také serverside skripty pro extrémně

[17] GRUDL, David. Ajax & snippety. [online]. [cit. 2014-01-21]. Dostupné z: http://doc.nette.org/cs/2.1/ajax

velké tabulky, které zpracují požadavek na serveru a do prohlížeče pošlou jen tolik řádků, kolik je tabulka schopna zobrazit najednou. Aplikace používá verzi 1.9.4.

### 4.5.3 Highcharts

Knihovna Highcharts umožňuje vývojáři tvořit grafy pomocí JavaScriptu z několika zdrojů dat. Podporuje velké množství různých typů grafů od koláčových přes sloupcové až po bodové a nabízí vysokou úroveň jejich modifikace a to vše velice jednoduše a rychle. V současné době funguje na EstiConu verze v3.0.9.

# 5 Tvorba aplikace EstiCon

V této části se zaměřím na tvorbu stavebního softwaru EstiCon. Nejprve bude popsána veškerá požadovaná funkčnost, která bude posléze analyzována. Z této analýzy bude poté vytvořen návrh samotné aplikace.

## 5.1 Požadavky na aplikaci

EstiCon bude komerčním softwarem, který bude rozdělen na dvě hlavní části. První z nich jsou stavební ceníky, které vydává firma IBR Consulting. Ta se specializuje právě na vydávání vlastních třídníků a cenových normativů. Druhou částí je investorské oceňování projektu, které má pomoci investorovi s odhadem stavebních nákladů.

### 5.1.1 Ceníky

Stavební ceníky jsou rozděleny do čtyř stupňů, které vystihují jednotlivé fáze projektu a v každém tomto stupni může být několik datových základen, například silniční nebo železniční, které obsahují různé položky. Každá datová základna má hodnoty v několika cenových úrovních, které jsou dány rokem a regionem.

První stupeň je záměr projektu (ZP). V tomto stupni jsou položky jednotlivými objekty stavby, například rychlostní silnice, protože v této fázi je podrobnější členění projektu nemožné. Tyto objekty jsou dále ještě rozděleny na tři typy. Objekty typu A, což jsou standardní objekty, mimoúrovňové křižovatky a objekty typu B, které nemají v ceníku cenu, ale procento, protože se počítají procentuelně z cen objektů typu A na konkrétním projektu. Jedná se třeba o inženýrské sítě a pomocné práce.

Druhým stupněm je dokumentace pro územní rozhodnutí (DÚR). Zde jsou položky také objekty stavby, ale jejich ceny už jsou mnohem přesnější a rozlišení objektů podrobnější.

Třetí stupeň je dokumentace pro stavební povolení (DSP). Položky jsou zde již jednotlivé části objektů, které jsou však stále logicky členěny v závislosti na jednotlivých objektech. Objektová skladba tak zůstává stejná jako ve stupni DÚR.

Posledním stupněm je zadávací dokumentace stavby (ZDS). Položky jsou zde již velice podrobné, protože vyplývají přímo ze zadání stavby zhotoviteli. Nejedná se už o objekty, ale pouze o jejich malé části. Pro lepší orientaci jsou položky součástí víceúrovňového stromu.

Ve všech stupních musí mít ještě každá položka detail, v kterém bude vidět vývoj cen v různých letech a regionech a atributy položky, které zpřesňují výslednou cenu. Ve stupni ZDS mají položky ještě technickou specifikaci a kalkulace, které uživateli ukazují z jakých materiálů a prací se cena položky vypočítala.

Registrovaný uživatel se zaplaceným přístupem k ceníkům si tak bude moci zobrazit v každém stupni ceník položek v libovolném roce a regionu a u každé položky si zobrazit jednoduše její detail.

### 5.1.2 Ocenění

V této části je třeba uživateli povolit založení stavby, pod kterou si bude moci zakládat jednotlivé varianty této stavby ve stupních ZP, DÚR, DSP a ZDS. Samotné ocenění bude ještě rozděleno na investiční, rizikové a pro některé stupně i celoživotní ocenění.

#### 5.1.2.1 Investiční ocenění

V investičním ocenění se bude u každé varianty definovat cenová úroveň, v které se varianta oceňuje a při její změně se bude muset celá varianta přecenit. V prvních třech stupních si pak bude uživatel zakládat jednotlivé části přidáváním položek z příslušných ceníků, jejichž cenu bude možno zpřesnit pomocí atributů a expertních úprav. Ve stupni ZP se musí automaticky generovat objekty typu B a mimoúrovňové křižovatky pro každou variantu. Ve stupni ZDS si nebude uživatel variantu tvořit pomocí položek z ceníků, ale bude si ji celou importovat ve formátu XC4 a po importu bude následně oceněna podle zvoleného roku a datové základny. Toto automatické zatřídění poté půjde změnit v detailu položky. V tomto stupni také nejde upřesňovat cenu pomocí expertních úprav, ale pouze pomocí atributů.

Výsledkem investičního ocenění je stavební náklad, který nezohledňuje žádna rizika ani možné pokuty.

#### 5.1.2.2 Rizikové ocenění

V rizikovém ocenění si bude uživatel procentuelně stanovovat hodnotu jednotlivých typů rizik na variantě a jednotlivých objektech varianty. Jedná se třeba o environmentální nebo geologická rizika. Některá rizika budou pro celou variantu stejná, a tak nepůjdou na úrovni objektů měnit.

Cílem toho ocenění je stanovit maximální hodnotu rizikových faktorů. Uživatel si poté na variantě zvolí procentuelní část rizikových nákladů, která se nakonec přičte ke stavebnímu nákladu a stanoví tak výslednou plánovací cenu projektu.

#### 5.1.2.3 Celoživotní ocenění

Celoživotní ocenění se bude vztahovat na stupeň DÚR, protože zde je ještě možné změnit veškeré objekty stavby a rozhodnout se, která varianta je nejvýhodnější.

Na stavbě si uživatel zvolí počet let, na který bude chtít vypočítat celoživotní náklady. Na již existující objekty z investičního ocenění, si pak bude moci navázat konstrukční části a elementy, které si bude moci vybrat z předem připravených šablon nebo si je bude moci vytvořit úplně sám. Každý element bude mít určitou životnost, po které se bude muset vyměnit, cenu této výměny za měrnou jednotku a množství. Půjde také tvořit logické vazby mezi elementy, kterými je možno definovat jejich vzájemné závislosti. Pokud tak třeba uživatel vytvoří závislost mostního zábradlí na mostní římse, bude to znamenat, že pokud se vyměňuje římsa, je nutné vyměnit také zábradlí. Každý element bude mít na sobě definované také různé údržby, například natírání nebo čištění.

Na každém objektu bude možno definovat jeho rozměry, které se využijí pro stanovení množství elementu, pokud se uživatel rozhodne využít našich předpřipravených šablon. Na každém objektu půjde také definovat cenu jeho provozu za měsíc a stejně jako na objektech i údržby.

Samotné údržby budou obsahovat informace o jejich intervalech provádění, ceně jedné údržby a také první a poslední rok, kdy se bude údržba provádět.

V rámci jedné varianty bude nutné zobrazit uživateli srozumitelný graf nákladů za výměny, provoz a údržby jednotlivých objektů varianty, jejich součet za celou variantu a také harmonogram výměn jednotlivých elementů s jejich zbývající životností při výměně,

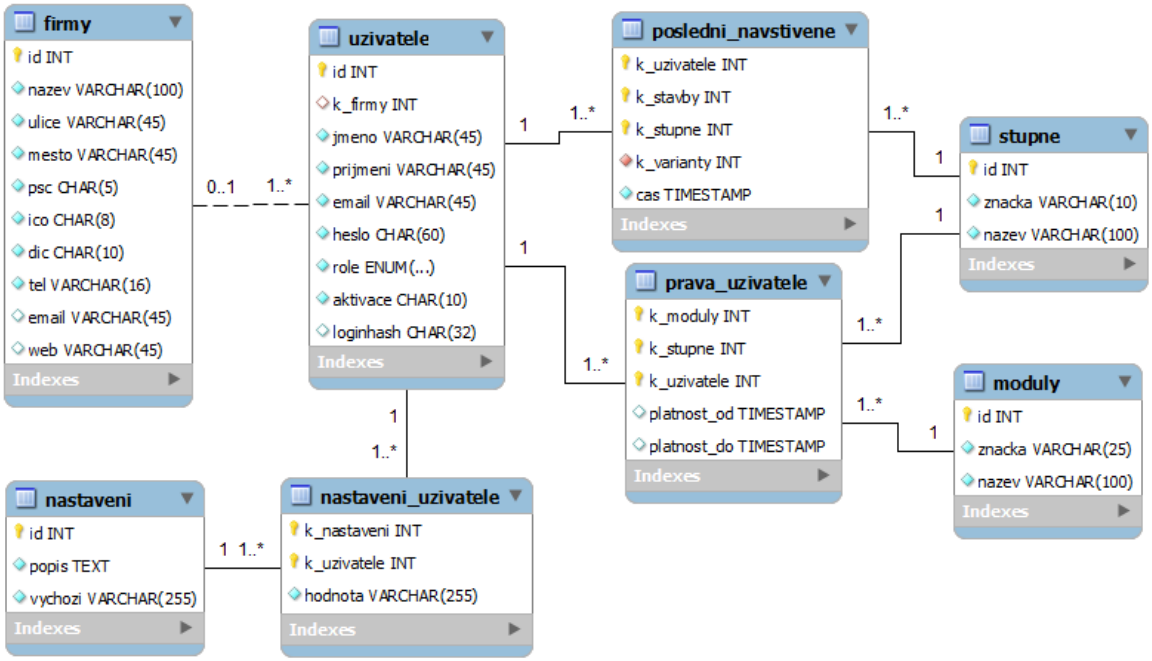
aby byl uživatel schopen optimalizovat životní cyklus celé varianty a ve výsledku tak snížit celoživotní náklady.

Pro uživatele bude však z celého celoživotního ocenění nejdůležitější samotné porovnání variant mezi sebou. Pro toto porovnání bude sloužit graf, v kterém budou zahrnuty celoživotní náklady vybraných variant.

## 5.2 Databáze

V následující kapitole bude rozebrána tvorba databáze pomocí ERA diagramů. Celá databáze je příliš rozsáhlá na to, aby se dala popsat najednou, proto bude popisována po jednotlivých logických celcích. Schémata databáze jsou vytvořena pomocí programu MySQL workbench, protože umí snadno vygenerovat skript na založení databáze na serveru. Databáze je relační, a proto se vytvoří ve formátu InnoDB. Pro porovnání různých hodnot používá kódování utf8_general_ci.

### 5.2.1 Uživatel



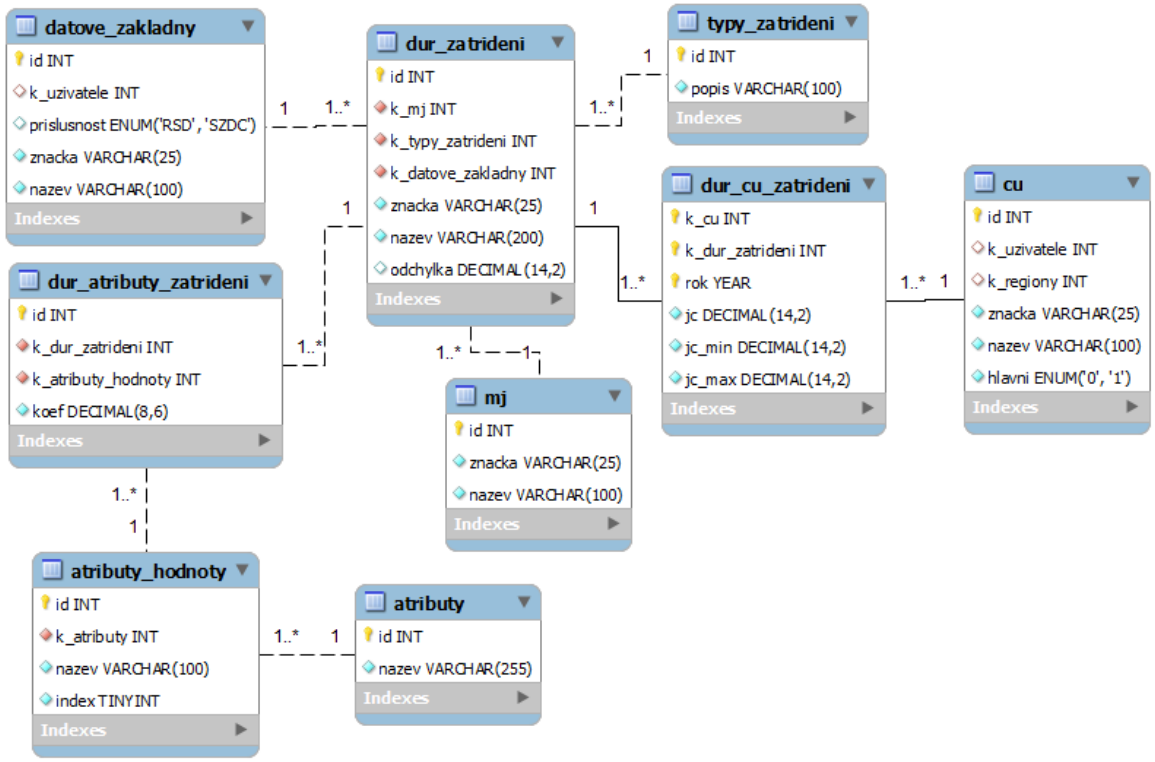

*Obrázek 1: Schéma uživatele*

Zdroj: vlastní

Na obrázku 1 můžeme vidět, jak je v databázi reprezentován uživatel. V tabulce *uzivatele* jsou uloženy konkrétní údaje o uživateli, jako jméno, příjmení nebo hashované heslo. Každý uživatel má také svojí roli, které jsou supervisor, administrator, uživatel nebo demo. Tyto role slouží k povolení přístupu k některým sekcím programu. Každý uživatel také může být zaměstnancem nějaké firmy, což se využije především při používání podnikové multilicence. V tabulce *prava_uzivatele* je uloženo, ke kterým modulům v konkrétních stupních si uživatel zaplatil licenci a do kdy je platná. V *posledni_navstivene* jsou uloženy údaje o posledních navštívených variantách v závislosti na stavbě a stupni. V *nastaveni_uzivatele* jsou pak uloženy údaje jako výchozí oceňovací rok nebo datová základna.

### 5.2.2 Ceníky

V této části bude popsán pouze ceník stupně DÚR, protože všechny jsou si kromě ZDS značně podobné.



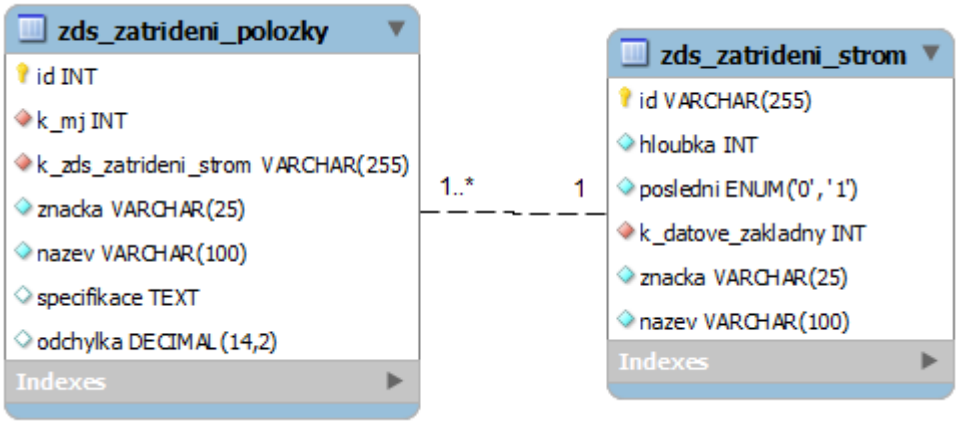

*Obrázek 2: Schéma databáze ceníku DÚR*

Zdroj: vlastní

Jak už bylo řečeno dříve, každá položka ceníku se váže na datovou základnu, která je reprezentována v tabulce *datove_zakladny*. Samotné položky jsou v tabulce *dur_zatrideni* a každá z nich má určený typ zatřídění z tabulky *typy_zatrideni* a měrnou jednotku z *mj*. Některé položky mohou mít také své atributy. Každý atribut má své hodnoty a ty se pak přes asociační tabulku k této položce přidají. Cena, která je závislá na cenové úrovni z tabulky *cu* a roku je uložena v asociační tabulce *dur_cu_zatrideni*.

### 5.2.3 Ceník ZDS

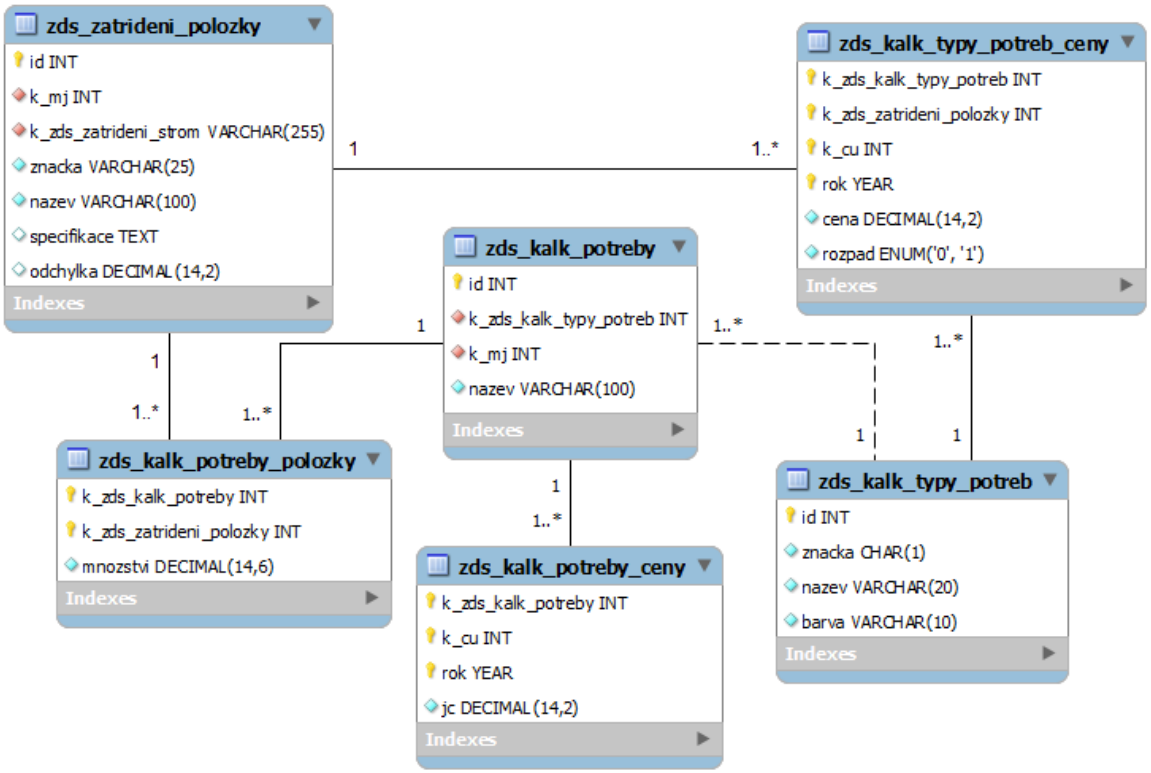
Ceník ZDS se od všech ostatních podstatně liší tím, že je uložen ve stromové struktuře pro snazší orientaci a vyhledávání. Dalším specifikem jsou také kalkulace, které ukazují rozpad ceny položky.




*Obrázek 3:Schéma stromu ZDS*

Zdroj: vlastní

Na obrázku 3 je vidět, jak je navržena struktura ukládání do stromové struktury, která je v tabulce *zds_zatrideni_strom*. Nejedná se o klasický naivní strom, kdy každá položka odkazuje na svého předka, ale o strom se strukturovaným ID, kde jednotlivé úrovně jsou odděleny pomlčkou, a tak v jsou v ID každé větve uloženy informace o všech předcích (např. 1-1-1-1-1). To velice usnadňuje a zrychluje procházení stromu, protože není třeba používat rekurzivní funkce.




*Obrázek 4: Schéma kalkulace ZDS*

Zdroj: vlastní

Kalkulace na stupni ZDS uživateli pomohou pochopit tvorbu expertní ceny, která z těchto kalkulací vychází. Každá položka se totiž skládá z typů potřeb, kterými jsou hmoty, mzdy, stroje, ostatní a neurčeno. Tyto typy dohromady tvoří nákladovou cenu a z ní se pak vypočítává režie a zisk. Každá položka má jinou cenu těchto typů potřeb v závislosti na roce a cenové úrovni. Některé typy potřeb pak mají ještě rozpad na potřeby, které mají také své ceny v tabulce *zds_kalk_potreby_ceny*. Položka, která obsahuje beton, bude mít typ potřeb hmoty a tyto hmoty se dále rozdělí na jednotlivé potřeby jako je písek a cement. Množství těchto potřeb se samozřejmě může lišit na různých položkách a je ukládáno do tabulky *zds_kalk_potreby_polozky*.

### 5.2.4 Stavba

Stavba je v databázi reprezentována pouze jednou tabulkou, v které jsou uloženy základní údaje, délka predikce pro celoživotní náklady, metodika ocenění a cizí klíč uživatele, který si stavbu založil.

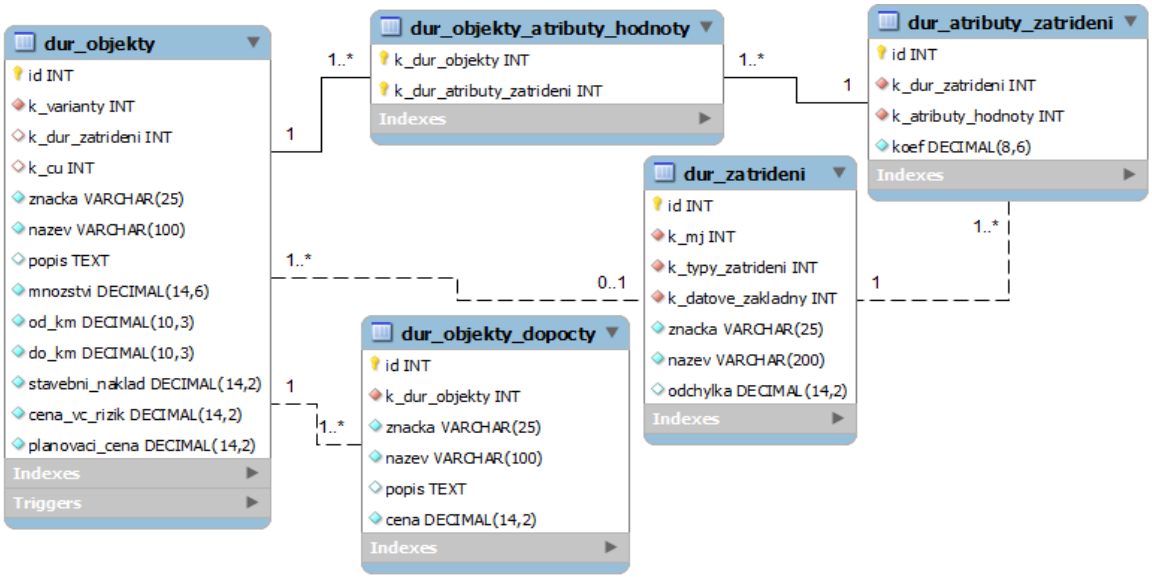
### 5.2.5 Varianta a rizika

Varianta je stejně jako stavba reprezentována jednou tabulkou, která obsahuje základní údaje, cizí klíč priority datových základen a cenových úrovní pro ocenění a cizí klíč stavby a stupně, pod které spadá. Na variantu se ještě vážou rizika, která jsou společná pro všechny objekty varianty.

### 5.2.6 Objekty

Protože objekty mají ve všech stupních různý význam, mají o sobě v databázi uložené i různé informace, a tak bylo nutné je rozdělit do čtyř oddělených a na sobě nezávislých větví. I když jsou takto rozděleny, základní logika je u všech objektů stejná, a proto se tato část zaměřuje pouze na stupeň DÚR, protože v sobě zahrnuje vše, co aplikace EstiCon nabízí. Jsou zde atributy, expertní úpravy, rizika i celoživotní náklady.

#### 5.2.6.1 Objekty DÚR



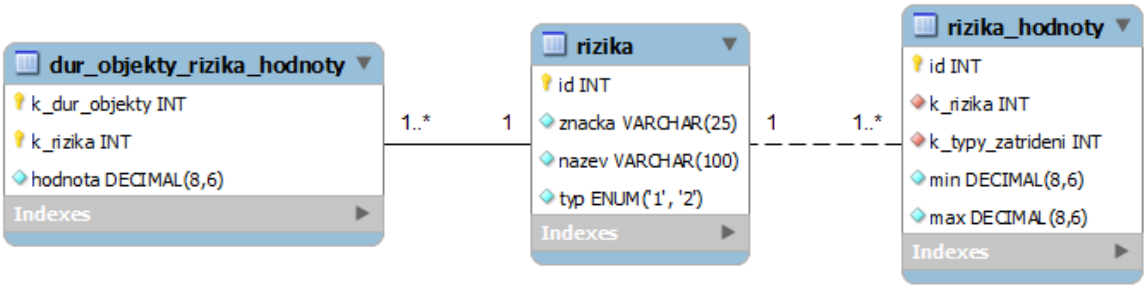

*Obrázek 5: Schéma objekty DÚR*

Zdroj: vlastní

Na obrázku 5 je vidět hlavní tabulku *dur_objekty*, ve které jsou uloženy základní údaje o objektech. Je zde také povinný cizí klíč na variantu, aby bylo jasné, pod kterou variantu objekt spadá. Dále je zde cizí klíč zatřídění a cenové úrovně, který je nepovinný z důvodu

možnosti importu z Excelu, kdy nemusí být specifikované zatřídění. Samotné zatřídění je navázáno přes tabulku *dur_zatrideni* a atributy jsou na objekt propojeny přes asociační tabulku *dur_objekty_atributy_hodnoty*. Expertní úpravy se ukládají do tabulky *dur_objekty_dopocty*.

#### 5.2.6.2 Rizika DÚR




*Obrázek 6: Schéma rizika DÚR*

Zdroj: vlastní

Rizika jsou reprezentována v databázi identicky pro všechny stupně i varianty a jejich konkrétní hodnoty pro každý objekt nebo variantu jsou uloženy v asociační tabulce. Ve stupni DÚR se jedná o tabulku *dur_objekty_rizika_hodnoty*. V tabulce *rizika_hodnoty* jsou pak uloženy doporučené hodnoty pro každý typ zatřídění. Samotná rizika mají také dva typy, které odlišují, zda se dané riziko dá měnit na úrovni objektů nebo je stejné pro celou variantu.

#### 5.2.6.3 Celoživotní DÚR

Celoživotní výpočty budou rozděleny na čtyři samostatné větve, jak je patrné z obrázku 21 v příloze B. Každý objekt bude mít provozní náklady, údržby a konstrukční části, které přímo ovlivňují výsledné celoživotní náklady. Poslední větví jsou rozměry objektu, které budou závislé na zatřídění a budou se používat pro výpočet doporučeného množství jednotlivých elementů. Elementy spadají pod konstrukční části a každý element má specifikovanou životnost, po které musí být vyměněn. Cena této výměny, která je rozdělena na jednotlivé zpracovatelské náklady, je taktéž uvedena na elementu. Tyto elementy je možno vybírat z předem připravených šablon nebo si je bude moci uživatel tvořit sám podle sebe. Každý element má také své údržby, které jsou nezávislé na údržbách

objektu. Uživatel také může definovat vazby mezi jednotlivými elementy a to jak v rámci jednoho objektu, tak také mezi objekty. Vazby mezi objekty si však musí povolit na variantě.

## 5.3 Aplikace

Samotná aplikace je v Nette rozdělena na tři vrstvy, protože, jak již bylo zmíněno, jedná se o framework postavený na modelu MVP, který vychází z konceptu Model-View-Controler (MVC). V této části práce bude zběžně popsána každá vrstva tohoto modelu v souvislosti s aplikací EstiCon a dále bude podrobněji popsán celý stupeň DÚR, od založení varianty přes tvorbu rizik až po celoživotní ocenění včetně ukázek zdrojového kódu.

### 5.3.1 Modely

Pro práci s daty a databází slouží v EstiConu modely, tak jak je to doporučeno i v Nette frameworku a MVP architektuře. Jedná se o objektové třídy, které obsahují funkce pro zpracování dat a jejich následné uložení do databáze, funkce pro načtení dat z databáze a jejich úpravu pro využití v samotné aplikaci nebo jen funkce pro zpracování přijatých dat a jejich následný návrat v upravené podobě. Některé modely také zpracovávají data z tabulkových procesorů a ta poté ukládají do databáze nebo vrací v asociačním poli.

Modely v EstiConu využívají dědičnosti a interfaců stejně jako v běžném objektovém programování. Každý model má minimálně jednoho předka, kterým je základní třída *Nette\Object*. Modely jsou logicky členěny podle toho, co reprezentují pro lepší orientaci.

Právě v modelech se pro práci s databází využívá výše zmiňovaná knihovna dibi, která umožňuje snadné přidání PHP proměnných do SQL dotazů a automaticky je escapuje podle datového typu, čímž je zajištěna bezpečnost dotazu. Samotné připojení k databázi je také objekt, který má v sobě nastavené parametry pro připojení do databáze. Instance toho objektu se tvoří v nastavení samotného Nette v souboru *config.neon*.

Většina modelů tak obsahuje podobné funkce, které vkládají, upravují nebo vybírají různá data. Většina dotazů vkládajících do databáze nová data vrací ID vloženého řádku což je v AJAXové aplikaci, jako je EstiCon velice důležité, protože toto ID se musí zpětně

injektovat do uloženého formuláře. Změny většinou nic nevrací, protože změněná data již jsou v odeslaném formuláři. Všechny funkce měnící data v databázi jsou uvnitř TRY-CATCH bloku, který je díky dalším dibi funkcím schopný při výskytu chyby vrátit databázi do původního stavu. Nette zároveň tyto chyby zachytává a umožňuje jejich zápis do logů, což je velice praktické při odlaďování chyb, protože jinak by vývojář musel pracně vyhledávat, kde k chybě došlo.

Funkce pro výběr dat pak vrací data v požadovaných formátech, čehož se docílí pomocí funkcí *fetch*. Samotný *fetch()* vrací asociační pole, pokud chceme pouze jeden řádek z databáze. Pro návrat více řádků slouží funkce *fetchAll()*, která vrací pole asociačních polí. Pokud je třeba vrátit pouze jednu hodnotu, využije se funkce *fetchSingle()*. Někdy je žádoucí vrátit páry klíčů a k nim přiřazených hodnot, například pro *SelectBoxy*. K tomu se využije funkce *fetchPairs('klíč', 'hodnota')*.

### 5.3.2 Šablony (View)

Pro zobrazení veškerého obsahu uživateli se využívají šablony. Pro tvorbu těchto šablon se využívá šablonovací systém Latte. Kromě tradičních HTML prvků a JavaScriptu obsahují šablony také latte makra.

Latte makra umožňují snadnou tvorbu všech cyklů nebo práci s PHP proměnnými přímo v HTML nebo JavaScriptu a v neposlední řadě také načtení části stránky z jiné šablony. Označují se složenými závorkami a aby se v JavaScriptu nepletlo označení maker se samotnými funkcemi JavaScriptu, musí samotné makro následovat hned po první složené závorce, jinak šablonovací systém bude chápat složenou závorku jako JavaScript nebo oznámí chybu HTML.

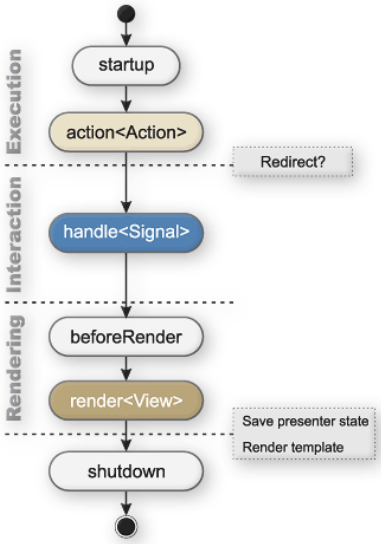
V šablonách se také používjaí výše zmiňované JavaScriptové knihovny jQuery, DataTables a HighCharts.

### 5.3.3 Presentery

Samotné presentery jsou z hlediska aplikace nejdůležitější vrstvou, protože propojují šablony a modely. Bez presenterů by nebylo možné uložit ani načíst žádná data a ani se přesměrovat na jinou stránku.

Každý presenter je stejně jako model objektová třída podporující dědičnosti. Všechny rozšiřují základní abstraktní třídu BasePresenter, která obsahuje základní funkce jako je *startup*, *actionDefault* nebo *beforeRender*. Tato třída je pak přímým potomkem *Nette\Application\UI\Presenter*.

Presentery obsahují několik různých typů metod a pro vývojáře je stěžejní vědět, kdy se který typ metod vykonává z pohledu životního cyklu presenteru. Tento životní cyklus můžeme vidět na obrazku 7.




*Obrázek 7: Životní cyklus presenteru*

Zdroj: http://doc.nette.org/cs/2.1/presenters

První z těchto metod je *startup(),* který inicializuje proměnné. Po ní následují *actionJmeno()*. Ta slouží především k tomu, aby provedla nějakou akci a poté přesměrovala na jinou šablonu, aniž by při tom něco vykreslovala. Nejdůležitější z hlediska AJAXu jsou pak handlery. Tyto metody zpracovávají libovolné uživatelské akce, umí přijímat i nastavovat proměnné a poté třeba překreslit část stránky. *BeforeRender()* slouží k nastavení proměnných, které mohou být společné pro více šablon nebo nastavení šablony. Metodou, která se v presenterech EstiConu vykonává naposledy je pak *renderJmeno()*. Ta pošle veškerá potřebná data pro vykreslení do šablony. Aby se spustila, musí mít stejné jméno jako action. Pokud action neexistuje, volá se automaticky *renderDefault()*. Metoda *shutdown()* v jednotlivých presenterech není,protože je zděděna ze zakládního Nette presenteru a vykoná se automaticky při ukončení presenteru.

Na začátku každého presenteru jsou také deklarované proměnné a objekty modelů, které během svého životního cyklu využívá.

### 5.3.4 Objekty DÚR

V této kapitole bude podrobněji rozebrán zdrojový kód aplikace EstiCon, tvorba komponent, formulářů, využití knihoven a ukázky latte maker ve stupni DÚR.

#### 5.3.4.1 Ocenění DÚR

Před tím, než je možné na nějaký objekt aplikovat rizika a celoživotní náklady, je třeba tento objekt nejprve založit a ocenit. Tyto dvě funkce v sobě spojuje presenter *ObjektyDURPresenter*, který k tomu využívá několika modelů, komponent a šablon. Pro otevření formuláře na založení objektu slouží hyperlink tag, který je vidět na obrázku 8. Obsahuje makro *n:href* s odkazem na handler presenteru *getObjektyDURForm*, do kterého posílá *id_varianty* a na stránce je representován ikonou. Má také nastavenu třídu (atribut class) *ajax*, z čehož Nette samo pozná, že po kliknutí na ikonu nemá uživatele přesměrovat na novou stránku, ale zpracovat AJAXový požadavek, což je zajištěno JavaScriptovou knihovnou *nette.ajax.js*, která ale není v základní instalaci Nette, ale musí se dodatečně doplnit.

```
<a n:href="getObjektyDURForm!, 'id_varianty' => $id_varianty"
    class="ajax ikona ikona_pridat"
    title="Přidat">
</a>
```

*Obrázek 8: Odkaz na otevření formuláře*

Zdroj: vlastní

V metodě presenteru, která se musí jmenovat *handleGetObjektyDURForm()*, aby ji bylo schopno Nette samo najít, jsou nejdůležitější tři řádky, které vidíme na obrázku 9. První příkaz pošle do šablony PHP proměnnou, druhý pak objekt typu JSON, který knihovna *nette.ajax.js* automaticky zpracuje a vytvoří z něj JavaScriptové proměnné a třetí dá příkaz k překreslení části stránky, která je uvnitř snippetu *objektyDURForm*.

```
$this->template->showObjektyDURForm = true;
$this->payload->showObjektyDURForm = true;
$this->invalidateControl('objektyDURForm');
```

*Obrázek 9: Invalidace formuláře*

Zdroj: vlastní

Protože snippety pouze označují část stránky k překreslení, ale vykreslují se i při načtení stránky samotné, je třeba obsah snippetu, který se nemá hned zobrazit obalit podmínkou if, stejně jako na obrázku 10. Z presenteru byl dán příkaz k překreslení a zároveň byla poslána proměnná, která splňuje podmínku, a tak se může vykreslit obsah snippetu. Další latte makro poté opět zavolá presenter, tentokrát s žádostí o vytvoření formulářové komponenty objektyDURForm, která bude opět AJAXová.

```
{snippet objektyDURForm }
    {if isset($showObjektyDURForm)}
        <div id="modal-objektyDURForm">
            {form objektyDURForm class => ajax}
        </div>
    {/if}
{/snippet}
```

*Obrázek 10: Ukázka snippetu*

Zdroj: vlastní

Formuláře se v Nette tvoří velice snadno, jak je vidět na obrázku 11. Framework totiž obsahuje třídu *FrontModule\Nette\Form*, do které pak velice snadno přidáme formulářové prvky. Můžeme jim také nastavit validační pravidla, CSS třídy nebo výchozí hodnoty. V každém formuláři se také definuje metoda, která se má zavolat po úspěšném odeslání formuláře a která při chybě. Poté se formulář vykreslí do šablony. Nette umí vykreslit formulář jak automaticky nebo umožňuje programátorovi ruční vykreslení. Jednotlivé formulářové prvky a jejich popisky se pak do HTML zakomponují pomocí latte maker *{input znacka}* a *{label znacka /}*.

```php
protected function createComponentObjektyDURForm() {

    $form = new Form();

    $form->addText('znacka', 'Objekt:', 10, 25)
        ->addRule(Form::FILLED, 'Vyplňte značku')
        ->getControlPrototype()->setClass('znacka');

    $form->onSuccess[] = callback($this, 'objektyDURFormSubmitted');
    $form->onError[] = callback($this, 'onFormError');
    return $form;
}
```

*Obrázek 11: Tvorba formuláře*

Zdroj: vlastní

Aby bylo ovládání co nejpříjemnější pro uživatele, je velice nepraktické, aby se tento formulář vykreslil přímo do stránky, jak by to automaticky udělal framework, ale je lepší ho otevřít v modálním okně. Toho se docílí pomocí JavaScriptové knihovny jQuery, nastavby jQueryUI a knihovny nette.ajax. Tato knihovna slouží k zachytávání proměnných z presenteru, které jsou posílány do proměnné payload, jak bylo ukázáno na obrázku 9. Na obrázku 12 je pak vidět, že pokud je zachycena proměnná JSON objektu zaslaného v presenteru, z HTML tagu div je vytvořeno modální okno, které obsahuje vytvořený formulář.

```javascript
$.nette.ext('EstiCon', {
    complete: function(data) {

        if (data.showObjektyDURForm) {
            modalForm({
                form_id: 'frm-objektyDURForm',
                div_id: 'modal-objektyDURForm',
            });
        }
```

*Obrázek 12: Vytvoření modálního okna*

Zdroj: vlastní

Poté co se otevře formulář a uživatel vyplní základní údaje, je třeba, aby k objektu také přiřadil nějaké zatřídění z oficiálního ceníku, jinak by daný objekt neměl žádnou cenu. Jelikož zatřídění vypadá ve všech stupních takřka identicky, pouze se načítají jiné ceníky,

byla k tomuto účelu vytvořena samostatná komponenta. Samotný požadavek na vytvoření komponenty probíhá v podstatě stejně jako u formuláře. Na stisknutí tlačítka se zavolá příslušný handler, který zinvaliduje část stránky (snippet), kde se zavolá vytvoření komponenty. Ovšem vytvoření komponenty v presenteru již vypadá jinak, jak je vidět na obrázku 13. Do konstruktoru komponenty *VyberZatrideni* se předají 3 parametry, které komponenta pro svou funkci vyžaduje. V tomto případě se jedná o model *cenikDUR*, model *varianty* a proměnnou *id_varianty*.

```
public function createComponentVyberZatrideni() {
    return new \VyberZatrideni($this->cenikDUR, $this->varianty, $this->id_varianty);
}
```

*Obrázek 13: Vytvoření komponenty zatřídění v presenteru*

Zdroj: vlastní

Poté co konstruktor nastaví všechny proměnné, je třeba získat samotný ceník z databáze. K tomu slouží funkce *getAll()* z modelu *CenikyDUR*, která je na obrázku 14. Využívá knihovny dibi, díky které se dá využít MySQL syntaxe s přiřazením proměnných získaných v konstruktoru pomocí modifikátorů. Data získaná tímto dotazem jsou do komponenty vrácena jako pole asociačních polí, která jsou následně odeslána do šablony komponenty v metodě *render()*.

```
public function getAll($k_cu, $k_dz, $rok) {
    return $this->db->query('
        SELECT `id`, `znacka`, `nazev`, `mj`, `jc`, `jc_min`, `jc_max`
        FROM `v_dur_zatrideni`
        WHERE `k_cu` = %i
        AND `rok` = %i
        AND `k_datove_zakladny` = %i', $k_cu, $rok, $k_dz
    )->fetchAll();
}
```

*Obrázek 14: Funkce getAll*

Zdroj: vlastní

V šabloně se data vykreslí do tabulky pomocí makra *n:foreach* jak je vidět na obrázku 15, které prochází celé pole *tableData* po jednotlivých asociačních polích, kdy jedno toto pole odpovídá jednomu řádku v tabulce a do každého sloupce vypíše určité hodnoty.

```
<tr class="point" n:foreach="$tableData as $row">
    <td>{$row->znacka}</td>
    <td>{$row->nazev}</td>
    <td>{$row->mj}</td>
</tr>
```

*Obrázek 15: Vykreslení tabulky*

Zdroj: vlastní

Protože tabulka bude obsahovat mnoho záznamů, je pro uživatele praktické, aby se dala třídit nebo aby se v ní dalo snadno vyhledávat. K tomu slouží JavaScriptová knihovna DataTables a její využití je velice snadné, jak dokládá obrázek 16. Stačí pouze napsat jméno tabulky a nastavit jednotlivé sloupce, filtrování nebo označení. Na obrázku je také vidět, že tabulka je typu *estiTable*, což je rozšířeni základní knihovny pro potřeby aplikace EstiCon.

```
$(function(){
oTableVyberZatrideni = $('#tbl_vyber_zatrideni').estiTable({
    "height": 300,
    "columns": [
        { "sType": "czech", "sWidth": "60px" }, // znacka
        { "sType": "czech" }, // nazev
        { "sType": "czech", "sWidth": "35px" } // mj
    ],
    "sorting": [[ 0, "asc" ]],
    "multiselect": false
});
```

*Obrázek 16: Vykreslení pomocí estiTable*

Zdroj: vlastní

Poté co si uživatel vybere dvojklikem zatřídění, které se pomocí JavaScriptu uloží do formuláře objektu, zpřístupní se mu také atributy. Ty jsou stejně jako zatřídění řešeny pomocí komponenty, protože jsou vázány pouze na zatřídění a ne na stupeň ocenění. Pokud uživatel vyplní všechny nezbytnosti a nastaví atributy ke své spokojenosti, může objekt uložit. Poté co uživatel potvrdí formulář, zavolá se automaticky funkce *ObjektyDURFormSubmitted()*. Ta je na obrázku 17. Hodnoty z formuláře se získají pomocí funkce *getValues()*. Atributy jsou uloženy ve skrytém elementu jako řetězec hodnot oddělených středníkem a proto je třeba dostat tyto hodnoty z řetězce do pole. K tomu se využije PHP funkce *explode*(). Veškerá práce s databází, která by mohla poškodit data v ní,

je uvnitř TRY-CATCH. Nejprve se ověří, zda má uživatel právo na založení objektu. Pokud má, zavolá se funkce *insert()* z modelu *ObjektyDUR*, kam se pošle pole *$values* a druhé pole *$atributy*. Tato funkce uloží hodnoty a atributy do databáze a vrátí ID nového záznamu. To se pak injectuje pomocí JavaScriptu zpět do formuláře, pokud ho uživatel odeslal pomocí tlačítka uložit a ne OK. Na uložení se také povolí expertní úpravy objektu a překreslí se tabulka objektů, ve které je nově uložený záznam barevně označen.

```
$values = $form->getValues();
$atributy = $values['atributy'];
$atributy = !empty($atributy) ? explode(';', $atributy) : null;
try {
        if ($this->user->canCreate()) {
            $id_objektu = $this->objektyDUR->insert($values, $atributy);
            $this->flashMessage('Objekt byl založen.', 'success');
            $this->payload->id_objektu = $id_objektu;
            $this->handleGetDopocty($id_objektu);
        } else {
            $this->flashMessage('Nemáte právo na založení objektu.', 'error');
        }
    $this->invalidateControl('objektyDURTable');
} catch (\EsticonException $ex) {
    $this->flashMessage($ex->getMessage(), 'error');
}
```

*Obrázek 17: Uložení formuláře*

Zdroj: vlastní

Pokud má uživatel oceněny všechny objekty, může přejít k rizikovému ocenění.

#### 5.3.4.2 Rizika DÚR

Jak již bylo zmíněno, rizika jsou na všech stupních vázány na objekty, a tak jsou vlastně na stupni nezávislé, protože objektovou skladbu má každý z nich. Některé stupně mají poté ještě podrobnější členění, to ovšem není pro rizikové ocenění podstatné. Proto byly vytvořeny tři komponenty, které se postarají o celé rizikové ocenění. Od tabulky, kde jsou vypsány objekty, které byly vytvořeny v ocenění a na něž se rizika váží, přes formulář, kde se nastaví konkrétní rizika až po vykreslení grafu rizik pomocí JavaScriptové knihovny Highcharts.

#### 5.3.4.3 Celoživotní DÚR

Když má uživatel stavbu oceněnu a nastavená příslušná rizika, dostane se do nejzajímavější části celé aplikace, kterou jsou celoživotní náklady.

Stejně jako předchozí dvě části stupně DÚR využívá značného množství různých komponent. Je zde komponenta na zakládání konstrukčních částí pod konkrétní objekty, které jsou převzaty z oceňovací části. Další slouží k zakládání elementů pod konstrukční části nebo k nastavení provozních nákladů objektu. Pro stanovení údržby na celém objektu ale i na elementech je také samostatná komponenta. Pro tvorbu vazeb mezi elementy byla vytvořena Drag and Drop komponenta, která nejen že umožňuje rychlé vytvoření těchto vazeb, ale také je ihned vykreslí včetně harmonogramu výměny s údajem o zbývající životnosti každého elementu v čase jeho nahrazení.

Kromě různých komponent se na celoživotním ocenění podílí i několik modelů. Nejzajímavějším z nich jsou *CelozivotniVypocty*. V tomto modelu, jak již název napovídá, probíhají veškeré výpočty celoživotních nákladů. Obsahuje několik pomocných private funkcí, které jsou stejné pro výpočty nákladů na všech úrovních od elementu až po variantu a nejsou přístupné mimo tuto třídu. Jedná se třeba o funkci *agregateArray*, která přijme jako parametr nějaké pole hodnot a to pak postupně agreguje tak, že každý další prvek je součtem všech předchozích nebo funkce *escalateArray* a *diskontArray*, které celé pole upraví podle nastavené eskalace nebo úrokové míry, aby bylo celé pole převedeno na čistou současnou hodnotu.

Tou nejdůležitější funkcí celého modelu je rekurzivní funkce *getRokyPredikce*. Tato funkce přijme *id_elementu* a zjistí, zda má nebo nemá nějaký nadřazený element. Pokud má nadřazený element, tak se zavolá znovu s ID tohoto elementu a roky výměn tohoto nadřazeného elementu přidá k rokům výměn podřízeného elementu. Pokud je životnost nadřazeného elementu nižší, tak se celé pole vymaže a vytvoří se znovu pouze z životnosti nadřazeného elementu. Toto se opakuje tak dlouho, dokud existuje nějaký nadřazený element a poté funkce vrátí pole všech let, kdy se bude původní element měnit. Tyto roky jsou velice důležité pro stanovení výsledných nákladů, protože element se v těchto letech nejen mění, ale také se musí od každého roku načítat údržby znovu od nuly.

Kromě těchto výpočtových funkcí obsahuje model veřejné funkce, které je možno volat z příslušného presenteru a komponent, které potřebují data pro celoživotní náklady. Tyto funkce přijímají několik logických parametrů a podle nich rozhodnou co se má spočítat, zavolají příslušné private funkce a vrátí pole spočítaných hodnot.

Zpracování výstupů těchto funkcí pak může být třeba harmonogram výměn, který je vidět v příloze A na obrázku 18 nebo graf celoživotních nákladů s rozpadem na výměny, údržby a provoz v příloze A na obrázku 19.

## 5.4 Ekonomické vyhodnocení

Cílem této kapitoly je provést ekonomické vyhodnocení projektu. Protože spuštění aplikace EstiCon se plánuje až na konec června 2014 a s ohledem na to, že firma vytváří takovouto aplikaci jako první na českém trhu, je téměř nemožné stanovit ziskovost tohoto projektu.

Dají se pouze vyčíslit dosavadní náklady na vývoj ve spojitosti s časovou náročností projektu a průměrnou hodinovou mzdou praktikanta bez vysokoškolského vzdělání.

Průměrný nástupní plat takového praktikanta je 15 000 Kč hrubého za měsíc na plný úvazek (z nejmenovaného zdroje). Při takovémto nástupním platu tak stojí zaměstnanec zaměstnavatele 20 100 Kč měsíčně za přibližně 160 hodin, což dělá náklad na hodinu vývoje přibližně 126 Kč. Doba strávená učením se nových technologií, stanovením a analýzou požadavku a tvorbou samotné aplikace je přibližně 1 800 hodin za celý vývojový tým. Cena vývoje celé webové aplikace se tak pohybuje přibližně kolem čtvrt milionu korun.

Po prozkoumání dotazníku pro PHP vývojáře[18] je patrné, že hrubá hodinová mzda úplně nezkušeného vývojáře na hlavní pracovní poměr je 150 Kč což činí náklady na hodinu vývoje přibližně 201 Kč. Nedá se předpokládat, že nezkušený vývojář by byl schopen takovouto aplikaci výrazně rychleji, a proto by celkový vývoj vyšel asi o 135 000 Kč dráže. Pokud by si firma IBR Constulting najala zkušeného vývojáře, cena za hodinu

[18] ŠTEKL, Martin. Výsledky dotazníku pro PHP vývojáře [online]. [cit. 2014-02-8]. Dostupné z: http://www.steky.cz/2013/03/24/vysledky-dotazniku-pro-php-vyvojare/

vývoje by se tak v průměru zvedla až na 536 Kč. Zkušený vývojář by mohl délku vývoje zkrátit až o polovinu, ale i tak by vyšel vývoj na 482 400 Kč.

Žádná jiná dostupná řešení není třeba zohledňovat, protože firma chce provozovat aplikaci ke komerčním účelům a jiné aplikace, které by se zabývaly investorským oceněním na českém trhu, nejsou známy.

Z analýzy cen na trhu práce tak vychází, že společnost si vybrala nejlevnější možné řešení vývoje aplikace EstiCon.

# 6 Závěr

Prvním cílem této práce bylo zhodnotit využitelnost webových aplikací jako nástroje informačních služeb. Jak bylo popsáno v první části, webové aplikace jsou díky moderním technologiím velice vhodné pro informační služby, protože nabízejí vyšší uživatelský komfort i dostupnost než aplikace nainstalované na konkrétních zařízeních.

Druhým cílem, kterým byla tvorba webové aplikace za použití moderních technologií, a hlavním přínosem je webová aplikace EstiCon, která umožňuje uživateli oceňovat pomocí oficiálních třídníků a stanovit investiční, rizikové a celoživotní náklady přímo ve webovém prohlížeči bez potřeby jakéhokoliv pevného klienta nebo jiných programů. Do aplikace lze též importovat předpřipravené projekty z Excelu, vytvářet vlastní položky ceníku ZDS, exportovat projekty do formátu pdf a to vše ve velice přátelském uživatelském prostředí. Aplikace také umí vytvořit grafy pro přehlednější získání informací o stavbě, nebo pro porovnání různých technických zpracování stejné stavby a všechna data jsou zobrazena v tabulkách, které umožňují třídění a vyhledávání.

Téměř veškerá práce s daty a změna obsahu stránky probíhá AJAXově pro ještě větší uživatelské pohodlí a rychlost aplikace.

Aplikace v této době běží pouze na vývojovém serveru a její nasazení pro testování se plánuje na polovinu června 2014. Spuštění aplikace je naplánováno na začátek července 2014.

Zatím ještě nebyly stanoveny ceny, za které se bude přístup do aplikace povolovat, pouze to, že se bude platit ve formě měsíčních poplatků a program bude rozdělen na jednotlivé moduly, které budou zpoplatněny zvlášť. Proto nelze stanovit ani hrubý odhad ekonomického přínosu pro firmu IBR Consulting, s.r.o. Pokud by se cena licence na rok stanovila přibližně na 20 000 Kč, jako tomu bylo u EstiRoadu, muselo by se prodat nejméně 13 kompletních ročních licencí, aby se pokryl náklad na dosavadní vývoj. Ten činil přibližně čtvrt milionu korun na celou aplikaci.

Největším společenským přínosem je samotné investorské ocenění, které umožní velkým investorům, jako je například Ředitelství silnic a dálnic (ŘSD), předběžný odhad

stavebních nákladů již před vypsáním výběrového řízení a může tak lépe odhalit dumpingové ceny.

Možná další rozšíření jsou celoživotní náklady na dalších stupních ocenění, automatická optimalizace délek životností jednotlivých elementů pro snížení celoživotních nákladů nebo peněžní ohodnocení přínosů plynoucích z realizace stavby. Také se dá stále pracovat na ještě přívětivějším ovládání a dalších uživatelských funkcích.

# Seznam použité literatury

## Citace


ANNON. *Buildpass – obnova a údržba objektů.* [online]. [cit. 2014-02-8]. Dostupné z: http://www.buildpass.eu/

ANNON. *MySQL Workbench 6.0: Visual Database Design.* [online]. [cit. 2014-01-21]. Dostupné z: http://www.mysql.com/products/workbench/design/

ANNON. *The history of HTML*. [online]. [cit. 2014-02-8]. Dostupné z: http://inventors.about.com/od/computersoftware/a/html.htm

ANNON. *History of PHP*. [online]. [cit. 2014-02-8]. Dostupné z: http://www.php.net/manual/en/history.php.php

ANNON. *A short history of Javascript.* [online]. [cit. 2014-02-8]. Dostupné z: http://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript

ANNON. *History of SQL*. [online]. [cit. 2014-02-8]. Dostupné z: http://www.stanford.edu/dept/itss/docs/oracle/10g/server.101/b10759/intro001.htm

BECK, Kent aj. *Manifesto for Agile Software Development*. [online]. [cit. 2014-01-21]. Dostupné z: http://agilemanifesto.org/

GRUDL, David. *Ajax & snippety*. [online]. [cit. 2014-01-21]. Dostupné z: http://doc.nette.org/cs/2.1/ajax

HO, Don. *Notepad++ Author*. [online]. [cit. 2014-01-21]. Dostupné z: http://notepad-plus-plus.org/contributors/author.html

KNESL, Jiří. *Agilní vývoj: Úvod*. [online]. [cit. 2014-01-21]. Dostupné z: http://www.zdrojak.cz/clanky/agilni-vyvoj-uvod/

LOUDON, Kyle. *Developing large web applications*. 1st ed. Sebastopol, Calif.: O'Reilly/Yahoo! Press, 2010. ISBN 05-968-0302-8.

MACCAW, Alex. *JavaScript web applications*. 1st ed. Sebastopol, CA: O'Reilly, 2011. ISBN 978-144-9303-518.

STROBL, Roman. *Happy birthday NetBeans – Interview with Jaroslav "Yarda" Tulach*. [online]. [cit. 2014-01-21]. Dostupné z: https://netbeans.org/community/articles/interviews/yarda-tulach.html

ŠTEKL, Martin. *Výsledky dotazníku pro PHP vývojáře*. [online]. [cit. 2014-02-8]. Dostupné z: http://www.steky.cz/2013/03/24/vysledky-dotazniku-pro-php-vyvojare/

ZÁDOVÁ, Vladimíra. *Konceptuální modelování v návrhu IS/ICT*. [online]. [cit. 2014-01-21]. Dostupné z: http://multiedu.tul.cz/~vladimira.zadova/multiedu/DATABAZOVE_SYSTEMY/DBS_koncept_do_PDF

ZÁDOVÁ, Vladimíra. Návrh *IS: Strukturovaný přístup*. [online]. [cit. 2014-01-21]. Dostupné z: http://multiedu.tul.cz/~vladimira.zadova/multiedu/informacni_systemy_III/IS3__09_10___struktur__pristup.pdf

ZÁDOVÁ, Vladimíra. *Objektový návrh IS.* [online]. [cit. 2014-01-21]. Dostupné z: http://multiedu.tul.cz/~vladimira.zadova/multiedu/informacni_systemy_III/IS3__10_11__objekt__pristup.pdf

# Bibliografie

DARIE, Cristian, Bogdan BRINZAREA, Filip CHERECHES-TOSA a Mihai BUCICA. *AJAX a PHP: tvoříme interaktivní webové aplikace profesionálně.* Vyd. 1. Brno: Zoner Press, 2006, 320 s. ISBN 80-868-1547-1.

DUCKETT, Jon. *Beginning HTML, XHTML, CSS, and JavaScript*. Wrox, 2010. ISBN 978-1-4571-0728-3.

GUTMANS, Andi, Stig Sæther BAKKEN a Derick RETHANS. *Mistrovství v PHP 5: tvoříme interaktivní webové aplikace profesionálně.* Vyd. 1. Překlad Bogdan Kiszka. Brno: CP Books, 2005, 655 s. ISBN 80-251-0799-X.

KOFLER, Michael a Bernd ÖGGL. *PHP 5 a MySQL 5: průvodce webového programátora.* Vyd. 1. Brno: Computer Press, 2007, 607 s. ISBN 978-80-251-1813-9.

LAVIN, Peter. *PHP - objektově orientované: koncepty, techniky a kód.* 1. vyd. Praha: Grada, 2009, 211 s. ISBN 978-80-247-2137-8.

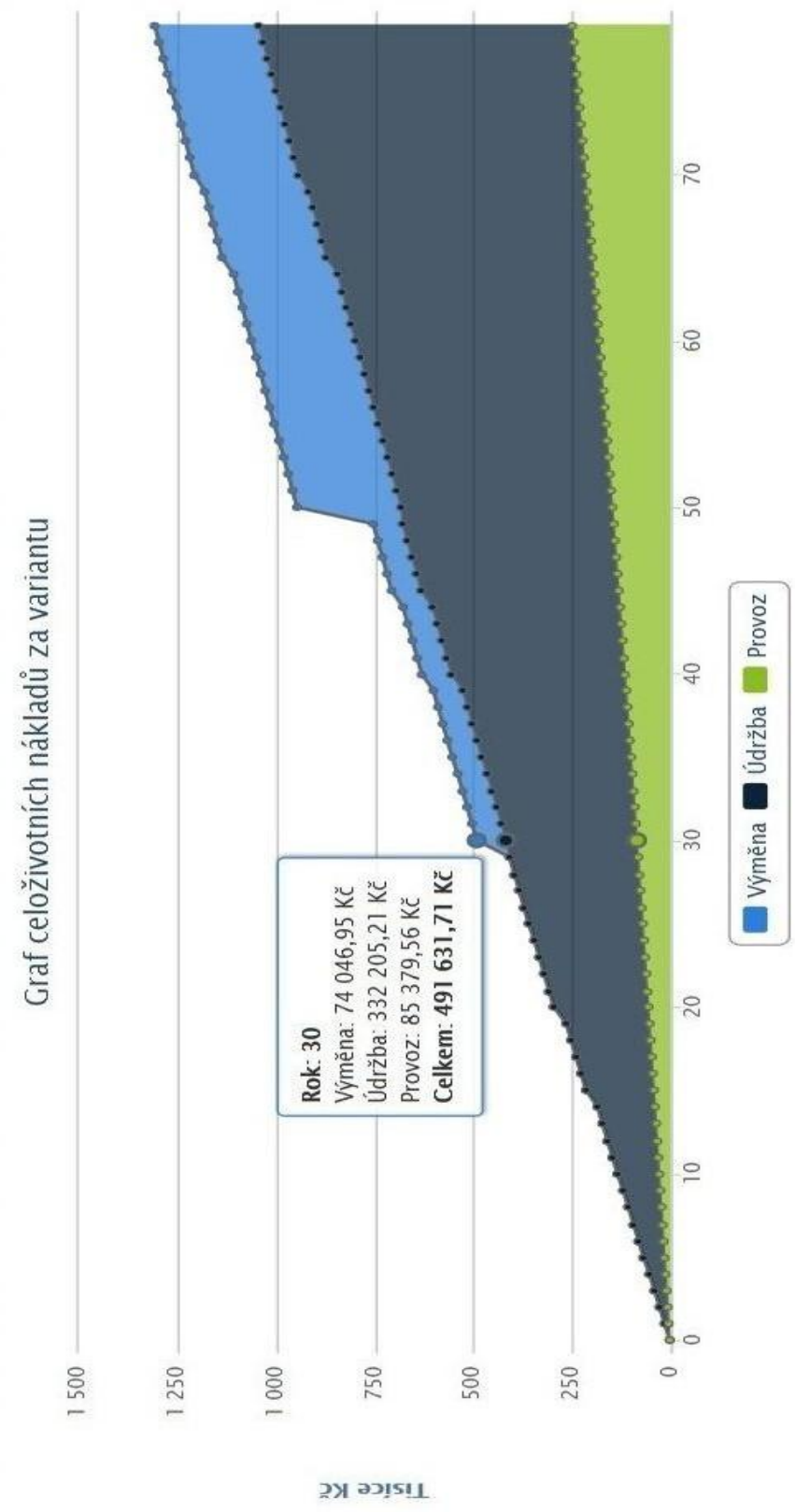
LURIG, Mario. *PHP reference: beginner to intermediate PHP5.* 1st ed. Raleigh, N.C., 2008. ISBN 978-143-5715-905.

# Seznam příloh

**Příloha A** – Vzhled aplikace

**Příloha B** – Schémata databáze

**Příloha A – Vzhled aplikace**




*Obrázek 18: Graf celoživotních nákladů*

Zdroj: Vlastní

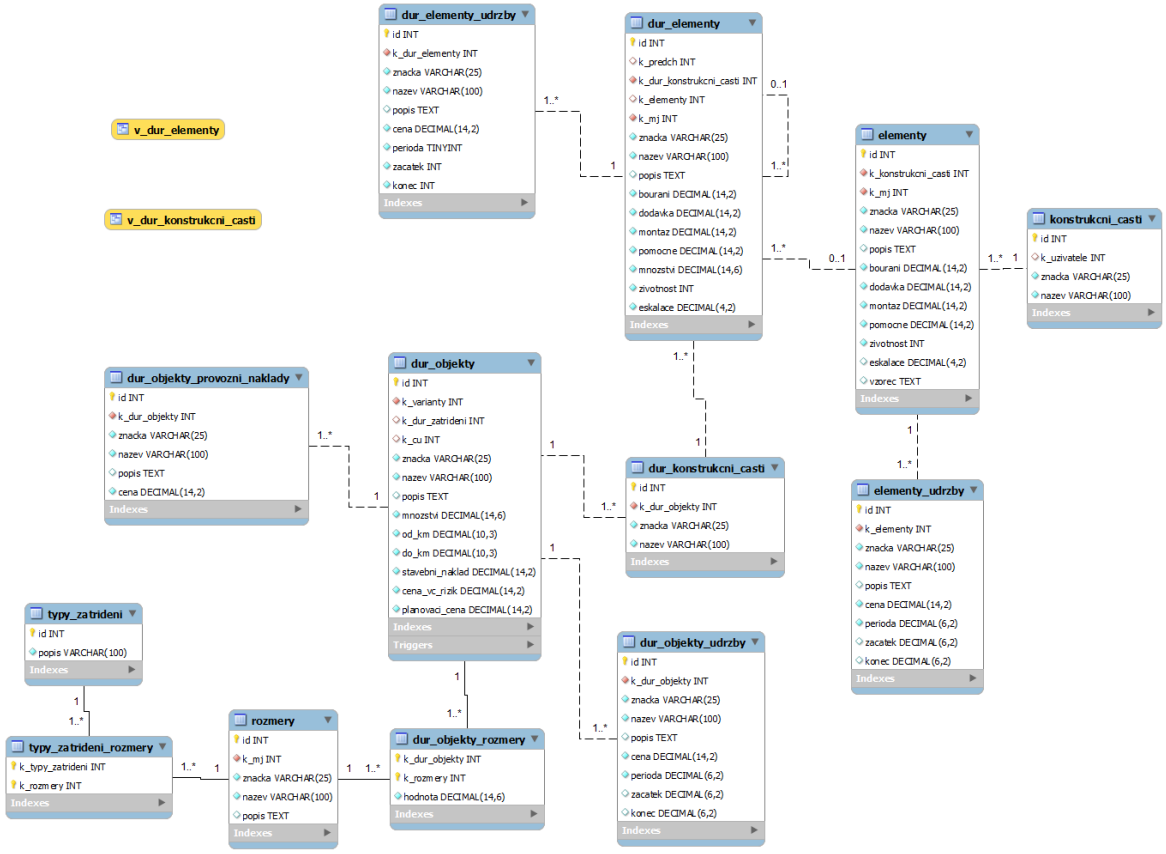

*Obrázek 19: Harmonogram výměn*

Zdroj: Vlastní

*Obrázek 20: Vzhled aplikace*

Zdroj: vlastní

**Příloha B – Schémata databáze**




*Obrázek 21: Schéma celoživotních nákladů DÚR*

Zdroj: Vlastní