Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Agregátor nabídek hospodyň

Adam Kořenek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačního inženýrství

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Adam Kořenek

Informatika

Název práce

Agregátor nabídek hospodyň

Název anglicky

Aggregator of housekeepers

Cíle práce

Cílém mé práce bude vytvořit informační systém, který bude zprostředkovávat nabídky hospodyň a agentůr zprostředkovávající domácí práce jako je například úklid, žehlení, praní apod. Stejnou funkci bude IS nabízet i na straně poptávky po těchto službách. K tvorbě funkčních částí IS se využívá programovací jazyk PHP, framework Nette a pro uchování dat se využívá databáze MySQL. Pro vývoj UI se využívá HTML 5, CSS, Java Script, knihoven Jquery a frontend framework Bootstrap.

Metodika

První fáze začíná analýzou problému, je potřeba porozumět problému a získat základní představu o funkcích webového informačního systému jako celku. Dále je potřeba rozpracovat funkcionality z hlediska přístupu uživatelů k IS.

Druhá fáze navazuje na funkcionality z analýzy a převádí je do logického designu. Tím vzniká tzv. wireframe uživatelského prostředí.

Ve třetí fázi se logický design přepracuje do grafické podoby a z ní již vzniká funkční šablona uživatelské prostředí vytvořené pomocí HTML, CSS, Java Scriptu, Jquery a Bootstrap.

V čtvrté fázi na základě analýzy, logického designu a šablony jsme schopni navrhnout strukturu databáze MySQL.

Následuje fáze pátá samotný vývoj funkční části aplikace, tedy naprogramování funkčních částí aplikace v PHP s využitím frameworku Nette.

V závěrečné fázi se otestují všechny části portálu jako celek a odladí se případné chyby.

Celý proces není zcela lineární a na závěr každé jednotlivé fáze je potřeba se zamyslet nad správností řešení a případně současné řešení upravit.

Doporučený rozsah práce

IS naprogramovaný v PHP, dokumentace k IS a popis jeho funkcí

Klíčová slova

webový portál, nabídkový/poptávkový agregátor, PHP, Nette, Jquery, Bootstrap

---

Doporučené zdroje informací

Dokumentace Nette frameworku: http://doc.nette.org/cs/2.1/
Mistrovství v jQuery (2013), ISBN: 978-80-251-4103-8
Návrhové vzory v PHP (2012), ISBN: 978-80-251-3338-5
Programujeme vlastní sociální síť v PHP 5 (2013), ISBN: 978-80-251-4103-8
Velká kniha PHP5 a MySQL (2011), ISBN: 978-80-7413-163-9

Předběžný termín obhajoby

2015/06 (červen)

Vedoucí práce

Ing. Marek Pícka, Ph.D.

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 15. 03. 2015

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Agregátor nabídek hospodyň" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 16.3.2015 _____

Poděkování

Na tomto místě bych rád poděkoval Ing. Marku Píckovi, Ph.D. za cenné připomínky a odborné rady, kterými přispěl k vypracování této bakalářské práce. Dále děkuji firmě WebAZ, s.r.o. za poskytnuté informace, konzultace a praxi.

Agregátor nabídek hospodyň

Aggregator of housekeepers

Souhrn

Cílem mé práce bude vytvořit informační systém na webu, který bude zprostředkovávat nabídky hospodyň a agentur, zprostředkovávající domácí práce, jako je například úklid, žehlení, praní apod. Stejnou funkci bude IS nabízet i na straně poptávky po těchto službách. K tvorbě funkčních částí IS se využívá programovací jazyk PHP, Framework Nette a pro uchování dat se využívá databáze MySQL. Pro vývoj UI se využívá HTML 5, CSS, Java Script, knihoven jQuery a frontend framework Bootstrap.

Summary

The goal of the bachelor‘s final project: “Aggregator of housekeepers” is to mediate offer of housekeepers and agencies offer housework for example cleaning, ironing, washing ect. The same function IS will provide for demand for this services. To develop function part of IS is used programming language PHP, framework Nette and to store data is using database MySQL. Technologies for creating UI are HTML 5, CSS, Java Script, library jQuery and frontend framework Bootstrap.

Klíčová slova

Informační systém, webový portál, nabídkový/poptávkový agregátor, PHP, Nette, MySQL, HTML, CSS, jQuery, Bootstrap

Keywords

Information system, web portal, offers/demands aggregator, PHP, Nette, MySQL, HTML, CSS, Jquery, Bootstrap

OBSAH

1 ÚVOD

S postupným rozvojem internetu a jeho dostupnosti se rozvíjejí i portály, agregátory a informační systémy na webu. Jejich výhod je nespočet, uveďme si ty nejzákladnější: snadná dostupnost z celého světa, nezávislost na zařízení (můžeme k webovému portálu přistupovat ze stolního počítače, notebooku, tabletu, či chytrého telefonu) s tím souvisí nezávislost na operačním systému (je jedno zda používáme Windows, Android, Linux, či Mac OS), *open source* vývojové prostředí pro webové aplikace (většina technologií a softwaru pro vývoj internetových aplikací není zpoplatněna a je distribuována pod *open source* licencí), současné prohlížeče podporují široké spektrum možností, jak za pomocí front-end frameworků a AJAX technologií vytvářet uživatelsky přívětivé prostředí. Vzhledem ke zmíněným výhodám jsem si zvolil informační systém na webu.

Při výběru konkrétního informačního systému jsem v první řadě hledal téma, které bude v budoucnu použitelné v praxi, a bude lidem usnadňovat práci. Na internetu mě zaujal článek o velké poptávce po hospodyních v domácnosti.

O hospodyně je v Česku stále větší zájem. Eva Kopečná z pražské společnosti Domestica, která se výchově hospodyň věnuje, je přesvědčena, že tato služba u nás má budoucnost. Poptávka totiž podle ní převyšuje nabídku. Najít šikovnou paní, které můžete svěřit klíče, je terno. [1]

To byl jasný impulz k vytvoření tzv. agregátoru poptávek a nabídek hospodyněk okořeněný sociálními funkcemi, který v době mého rozhodnutí nemá přímou konkurenci.

2 CÍL A METODIKA

2.1 Cíl

Cílem této bakalářské práce je vytvořit návrh informačního systému, který zprostředkuje nabídku a poptávku po hospodyních, případně agenturách zprostředkovávajících tyto služby, a jeho následná realizace. Tento agregátor bude disponovat třemi uživatelskými rolemi: běžný uživatel (hledá hospodyni nebo agenturu), hospodyně (nabízí své služby), agentura (nabízí služby hospodyň). Vyhledávání hospodyň bude fungovat na základě zadání lokality, kde chce klient využívat služby hospodyň. Vzhledem k využitelnosti tohoto portálu na tabletech a chytrých telefonech bude front-end navržen responzivně.

2.2 Metodika

Metody a postup zpracování se odvíjí od stanovených cílů, které byly popsány výše. Z počátku je nutné vybrat vhodné technologie pro realizaci webového portálu. Tyto technologie budu blíže specifikovány v teoretické části mé práce. Ve chvíli, kdy jsou stanoveny konkrétní technologie, může se začít s vlastní tvorbou informačního systému. V první řadě se provádí analýza problému vzhledem k stanoveným cílům. Na základě analýzy se pak pokračuje logickým designem, který nastíní budoucí vzhled a logickou funkčnost. V této fázi je možné odhalit případné nedostatky ještě dříve, než se začne s implementací systému. Nyní je nashromážděno potřebné množství informací k návrhu datového modelu. Následuje samotná tvorba UI a funkčních celků, tedy naprogramování celého informačního systému. Již hotový systém je potřeba otestovat, aby se odhalily případné chyby, které se následně opraví. V závěru práce se provede zhodnocení výsledků, dosažení stanovených cílů a zamyšlení nad případným rozšířením portálu do budoucna.

3 TEORETICKÁ VÝCHODISKA

3.1 Apache server

Server Apache vznikl v roce 1993 na Illinoiské univerzitě, původně pod názvem NCSA HTTPd. Později byl podle anglického „a patchy server“ přejmenován na Apache. [2]

Apache je softwarový server, tedy program, který běží na hardwarovém stroji připojeném do internetu a zajišťuje obsluhu prohlížečů jednotlivých návštěvníků (posílá jim jednotlivé stránky). Mezi výhody Apache patří zejména dostupnost pro všechny hlavní platformy (Windows, Linux…) a také fakt, že Apache je vyvíjen jako open source (a je tedy k dispozici zdarma). [2]

Společně s PHP (serverové skriptování) a MySQL (databáze) patří Apache k tzv. triádě, trojici programů nejčastěji používaných k vytváření dynamických internetových stránek. Svou roli v tom hraje i poměrně jednoduchá instalace Apache na domácích počítačích, čímž lze zdarma vytvořit velmi kvalitní vývojové prostředí. Nepřekvapí tedy, že podle statistik z roku 2005 slouží Apache na 69 % všech internetových serverů. [2]

Apache je možné nainstalovat i na běžný počítač a snadno vyvíjet webové aplikace bez připojení k internetu. K tomu slouží program XAMPP, který lze jednoduše nainstalovat a vytvořit tím z počítače plnohodnotný server. XAMPP kromě PHP zvládá i MySQL databáze včetně phpMyAdmin, Filezilla FTP server, Mercury, Tomcat a další. Jediný problém při provozování serveru na běžném PC může být konflikt s aplikací Skype, ta totiž využívá port 80, který pro svůj chod potřebuje i XAMPP. Pro souběžný chod těchto aplikací je potřeba změnit aplikaci Skype port na jiný než 80. [3]

Obrázek 1- Apache server na webu [4]

3.2 HTML

Hyper Text Markup Language je značkovací jazyk, který tvoří základ pro veškeré webové stránky internetu. HTML se používá pro značení textu, ostatního obsahu a určuje celkovou strukturu stránky. [5]

Webová stránka je tvořena nejen textem, ale i obrázky, videi atd. Každá část tohoto obsahu je značena HTML značkami (prakticky se jedná o kolekci symbolů a slov, které jsou srozumitelné pro počítač). HTML se používá k definování struktury stránky, určuje jednotlivé části jako například (hlavička, tělo dokumentu a patička). [5]

HTML se používá k definování struktury dokumentu, nicméně neříká nic o tom, jak stránka vypadá, například (barva, ohraničení, pozice prvků). O vzhled výsledné stránky se stará Cascading Style Sheets, zkráceně CSS. [5]

HTML definice vznikla v roce 1991 a od té doby se neustále vyvíjí. Nejnovější verzí HTML je HTML5, které bylo oficiálně dokončeno v roce 2014. První návrh HTML5 byl zveřejněn již v roce 2008, ale až v roce 2014 byla dokončena finální verze. [5]

Oficiální specifikace HTML5 přinesla mnoho nových možností, které umožňují vývojářům vyvíjet weby rychleji a chytřeji, než tomu bylo se staršími verzemi HTML. Tyto nové možnosti obsahují například Local storage (umožňuje ukládat data na uživatelově PC) a HTML5 Video (umožňuje přehrávat videa v prohlížeči bez nutnosti plug-inu jako například Flash), dále také formulářové prvky pro výběr data či posuvníky. [5]

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>

<body>
Content of the document......
```

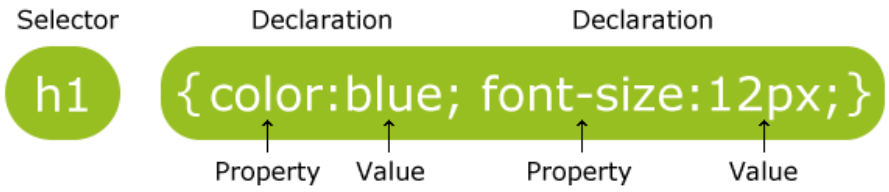
Obrázek 2- Ukázka syntaxe HTML 5 [6]

Nicméně ne všechny nové prvky HTML5 se dají bez problémů používat, stále spousta uživatelů používá starší verze prohlížečů, které tento standard nepodporují, například uživatelé Windows XP nemohou aktualizovat Internet Explorer na vyšší verzi než 8.

3.3 Cascading Style Sheets

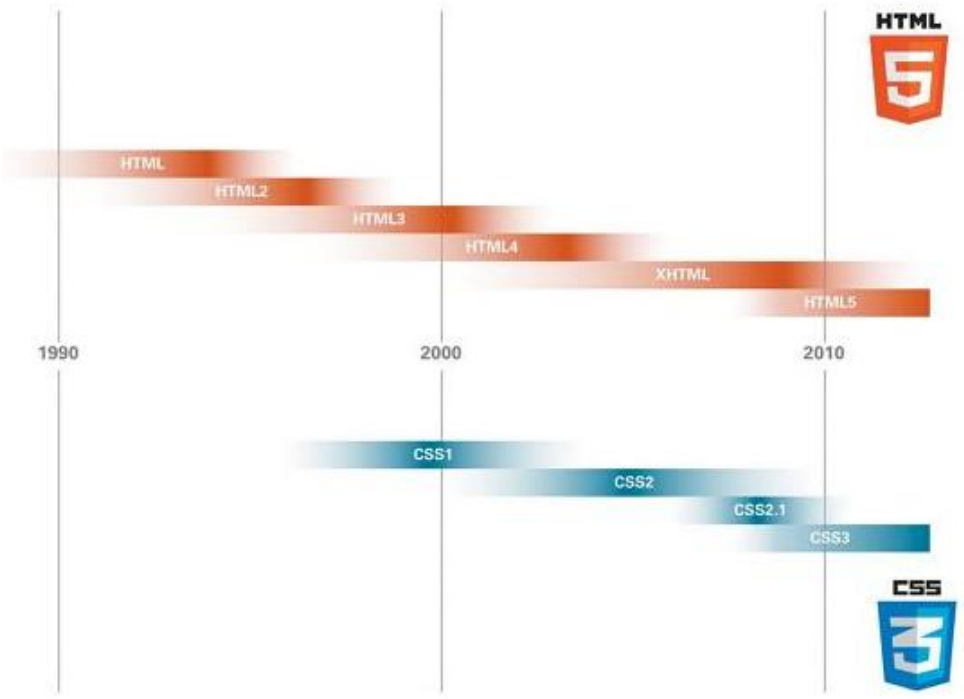
Cascading Style Sheets (dále jen CSS) byly navržen standardizační organizací W3C v roce 1996, kdy v čele vývoje prvotního návrhu stál Håkon Wium Lie. Hlavní myšlenkou je oddělení webové struktury od popisu vzhledu výsledné stránky, což mělo původně umět HTML, ale vzhledem k nedostatečné standardizaci a konkurenčnímu boji mezi prohlížeči se tomu tak nestalo. [7]

CSS je jazyk, který definuje, jak má prohlížeč interpretovat HTML elementy a jejich obsah. Pomocí CSS lze snadno definovat kompletní vzhled výsledného webu, včetně pozice elementů, barev elementů, vlastností obrázků, ohraničení, typů písma, a mnoho dalších vizuálních vlastností. [7]



Obrázek 3- ukázka syntaxe CSS [8]

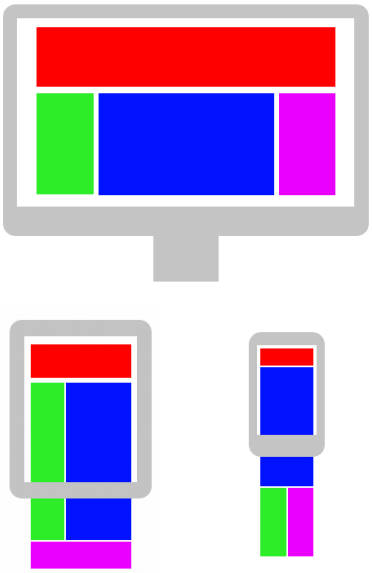
Stejně jako se neustále vyvíjí HTM, tak i CSS nezůstává pozadu a stále přináší nové možnosti. Nejnovější specifikací CSS je CSS3, které jde ruku v ruce s HTML5. [7]



Obrázek 4- Vývoj HTML a CSS [7]

3.4 Bootstrap

Bootstrap vyvinul Mark Otto a Jacob Thornton pro potřeby Twitteru a podporu konzistence mezi interními nástroji. Jedná se o open-source Framework pro rychlejší vývoj front-endu webových aplikací s ohledem na multiplatformní vývoj. Jinými slovy je to kolekce nástrojů, které ulehčují vývojářům vytváření moderních webových aplikací, které fungují responzivně. Rozmístění jednotlivých bloků webové stránky se flexibilně mění podle velikosti displeje. Obsahuje základní HTML a CSS šablony, typografie písma, formulářů, tlačítek, navigací, mřížku layoutu a dalších komponent rozhraní, dále také možnost JavaScriptového rozšíření jako například jQuery. [9]



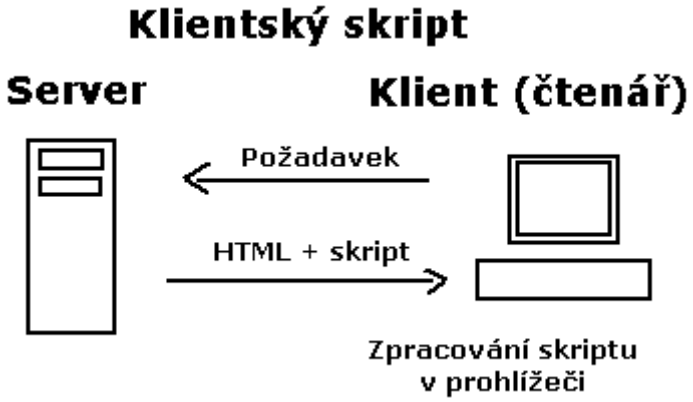
Obrázek 5- Responzivní design [10]

Základem pro tvorbu respozivních aplikací pomocí Bootstrap je mřížka tvořena 12 sloupci. Tyto sloupce se pak přeskládají sami podle velikosti obrazovky jak je vidět na obrázku výše. Velikost obrazovky pro sloupce v mřížce se určuje pomocí 4 tříd [9]:

- xs (pro telefon) <768px
- sm (pro tablet) >=768px
- md (pro notebook) >=992px
- lg (pro velký monitor) >=1200px

3.5 JavaScript

JavaScript je dynamický, objektově orientovaný, interpretační, multiplatformní programovací jazyk vyvinutý firmou Netscape v roce 1995. Jeho syntaxe patří do rodiny jazyků C/C++/Java. Obecně je programování kratších programů ve skriptovacím jazyce jednodušší a rychlejší než v kompilovaném jazyce jako je třeba C a C++. JavaScript se primárně interpretuje v prohlížeči, ale je možné ho umístit i přímo na server, například node.js složený z knihoven od Googlu. [11]

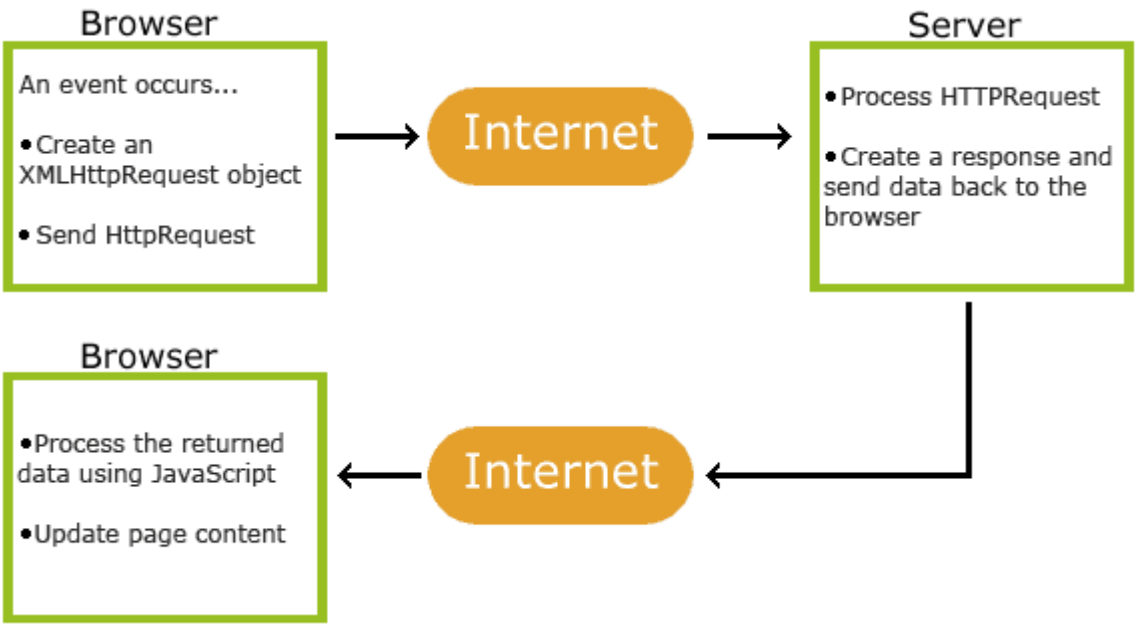


Obrázek 6 - Princip JavaScriptu [12]

JavaScript je možné zapisovat přímo do hlavičky resp. těla HTML dokumentu mezi párovou značku <script>. Nicméně pro rozsáhlejší skripty je vhodné uložit skript do separátního souboru a ten do HTML dokumentu importovat pomocí značky <script src="myScript.js"></script>. Umístění skriptů na webové stránce je volitelné, ale z hlediska rychlosti načítání stránky je vhodné veškeré JavaScripty umístit do těla dokumentu až úplně na konec pod veškerý obsah. [11]

3.6 AJAX

AJAX (Asynchronous JavaScript and XML) je označení pro technologii sloužící pro vývoj moderních interaktivních webových aplikací, které mění obsah stránky bez toho, aby jí bylo nutné znovu načíst. Ajaxové aplikace pro zobrazování obsahu využívají HTML, CSS a technologii JavaScript pro zobrazování dynamických změn. Pro výměnu informací se serverem se využívá formát údajů XML. Z toho vyplývá, že AJAX vlastně není konkrétní technologie, ale pojem pro společné využívání několika dosud známých a zavedených technologií. Záměrem AJAXu je vytvoření interaktivního uživatelského rozhraní webových a internetových aplikací. Aplikace AJAX nejčastěji využívají asynchronní komunikaci webového prohlížeče se serverem pomocí objektu XMLHttpRequest, nicméně není to vždy podmínkou. [13]

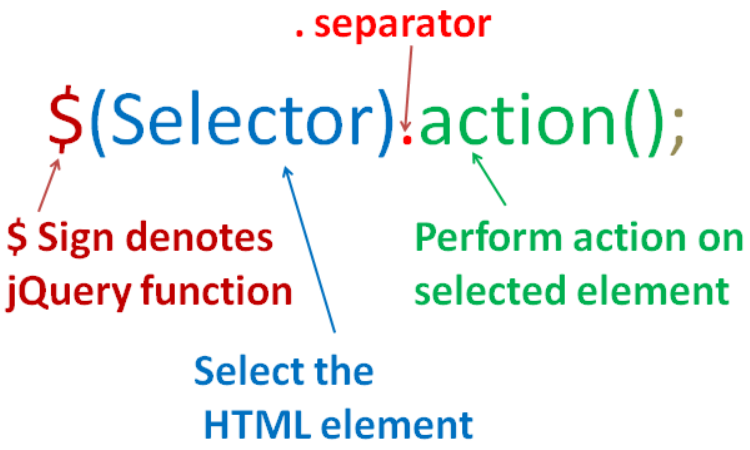


Obrázek 7- Princip AJAXu [14]

3.7 jQuery

jQuery je komplexní JavaScriptová knihovna, která zjednodušuje vývoj interaktivních webových aplikací s podporou nejpoužívanějších webových prohlížečů. Umožňuje snadnou práci s elementy v HTML stránce, zpracování událostí, manipulaci s CSS, efekty, animace a v neposlední řadě také AJAX. Díky jednoduchosti jQuery není potřeba mít k používání tohoto frameworku nijak rozsáhlé znalosti JavaScriptu. Další výhodou je nesrovnatelně kratší zápis výsledného kódu, než kdybychom stejnou funkcionalitu zapisovali v čistém JavaScriptu. První verze jQuery byla představena v roce 2006, ihned po jejím představení došlo k prudkému rozšíření na mnoho populárních webových stránkách. [15]

Velkým přínosem jQuery je oddělení funkční části front-endu od HTML struktury, což zvyšuje přehlednost zdrojového kódu samotné stránky a umožňuje snazší znovu použitelnost JavaScriptového kódu. Tomuto oddělení chování od struktury se také říká princip nevtíravého JavaScriptu. [16]



Obrázek 8- jQuery syntaxe [16]

3.8 MySQL

MySQL bylo vytvořeno švédskou firmou MySQL AB v roce 1995, nyní patří společností Sun Microsystems a dceřinou společností je Oracle Corporation. Jeho hlavními autory jsou Michael Widenius a David Axmark. K dispozici jak pod bezplatnou licencí GPL, tak i pod komerční placenou licencí, čímž je považován za průkopníka dvojího licencování.

MySQL je open-source relační databázový systém (RDBMS), který běží na serveru a umožňuje multi-uživatelský přístup k mnoha databázím.

MySQL používá velmi dobře známý standard SQL jazyka.

MySQL funguje na mnoha operačních systémech a programovacích jazycích například: PHP, PERL, C, C++, JAVA etc.

MySQL podporuje velké databáze - až 50 miliónů řádků v jedné tabulce a více. Standardní limit velikosti databáze je 4GB, ale pokud to operační systém umožňuje, lze tento limit navýšit až na 8 miliónů TB.

3.9 Návrh MySQL databáze

Tato část byla zpracována podle [17]

3.9.1 Postup návrhu „shoda dolů“

Při návrhu databáze jsou nejprve specifikovány struktury entitních množin a následně funkční závislosti mezi nimi.

Jednotlivé kroky modelování:

- Specifikování entitních množin – verbální popis problémové domény, specifikovaným objektům se přidělí jednoznačné, vhodné pojmenování.
- Specifikace vztahů:
    - Kardinalita vztahu (1:1, 1:N, M:N)
    - Povinnost vztahu je v grafickém návrhu určena pomocí spojnice. Přerušované spojnice znamenají nepovinnost vztahu a plná spojnice znamená povinnost vztahu.
    - Transformace modelu do logické struktury – konceptuální datový model je rozpracován do logické struktury „předběžných“ relací. Tyto relace jsou ovlivněny kardinalitou vztahů:
        - Vztah 1:1
        - Vztah 1:N – nejčastější, řešení minimálně 2 relacemi
        - Vztah M:N – 3 relace

3.9.2 Prověření modelu z hlediska normalizace

Účelem datové normalizace je prověření strukturální správnosti a prověření konzistence datového modelu.

Cílem normalizace modelu je možnost reprezentace každé relace v databázi, optimalizace algoritmů vyhledávání, zjednodušení aktualizace a rušení vět. Postupným reverzibilní procesem nahradíme dané množiny relací relacemi, které mají jednodušší strukturu.

Mezi klíčové vlastnosti datové normalizace patří možnost ověření správnosti navrženého modelu. Slouží k dekompozičnímu návrhu tabulek s minimální redundancí dat. Hlavní přínosy

datové normalizace jsou konzistentní neduplicitní data, úspora kapacity média, zjednodušené aktualizace a vyhledávání dat. Výsledné dekomponované relace je možné opět funkčně spojit.

3.9.3 Jednotlivé normální formy

1. NF relace nesmí obsahovat násobná data (např. opakující se jméno – vertikálně).

2. NF všechna neklíčová data relace musí funkčně záviset na celém primárním klíči.

3. NF všechna neklíčová data musí záviset pouze na klíčových hodnotách a nikoliv mezi sebou.

3.9.4 Přiřazení domény jednotlivým atributům

V rámci přiřazení domény jednotlivým atributům je třeba stanovit charakteristiky jednotlivým atributům. Doména nepokrývá případné vstupní kontroly, ty je potřeba ošetřit na úrovni formulářů. Atributům nastavujeme následující domény:

- Datový typ
- Délka
- Rozsah
- Přípustné hodnoty
- Formát (maska)
- Přípustnost „NULL“ hodnoty

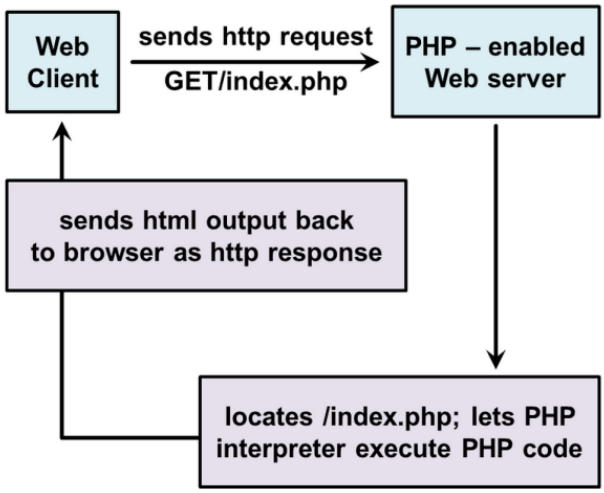
3.9.5 MySQL Workbench

MySQL Workbench zjednodušuje návrh a údržbu databáze, automatizuje časově náročné a k chybám náchylné úkoly. Zlepšuje komunikaci mezi DBA a vývojovými týmy. To umožňuje datovým architektům vizualizovat požadavky, komunikovat se zúčastněnými stranami a řešit problémy. Umožňuje designovat model databáze, což je nejúčinnější metoda pro vytváření platné a dobře fungující databáze, a zároveň poskytuje flexibilitu reagovat na vyvíjející se obchodní požadavky. Model a Schema Validation pomůcky umožňují dodržování praktických pravidel pro modelování dat. [18]

3.10 PHP

PHP je objektově orientovaný, skriptovací programovací jazyk šířený pod otevřenou PHP licencí, navržený pro tvorbu dynamického HTML obsahu. Původně vznikl v roce 1995, kdy zkratka znamenala Personal Home Page, od té doby prošlo PHP dlouhým vývojem a nyní tato zkratka znamená rekursivně Hypertext Preprocessor. [6]

Standardní činnost PHP na webu je následující. Uživatel zadá ve svém prohlížeči URL, jinými slovy odešle HTTP požadavek na server. Veškeré PHP skripty jsou uloženy na serveru a na serveru se interpretují na HTML stránku, kterou server vrací při úspěšné interpretaci prohlížeči. Prohlížeč zobrazí uživateli webovou stránku. [6]



Obrázek 9- PHP klient-server [19]

PHP je nezávislé na operačním systému a takřka většina skriptů se dá bez problému přenášet z Linuxu na Windows. Občas může být jen problém s knihovnami napsanými přímo pro danou platformu. I přes tuto skutečnost je PHP nejčastěji používané na Linuxu. Díky jeho otevřené licenci, jednoduchosti použití, velkou řadou propracovaných funkcí a integraci knihoven pro práci s MySQL databázemi se PHP stalo nejpoužívanějším programovacím jazykem pro tvorbu webových aplikací (v roce 2014 to bylo 82 %). PHP je vhodné jak pro menší webové stránky, tak i pro velké webové systémy jako třeba Wikipedia, Wordpress nebo částečně i Facebook a Yahoo (než bylo převedeno na node.js). Často se balík pro tvorbu webových aplikací označuje jako LAMP, tedy Linux, Apache server, MySQL a PHP. [6]

Syntaxe jazyka PHP původně vycházela z několika programovacích jazyků: C, Java, Perl a Pascal. Základem syntaxe je uvedení PHP skriptu pomocí <?php a uzavření ?>. Koncovka souboru je logicky podle názvu *.php. PHP soubory mohou obsahovat základní HTML strukturu, nicméně mohou být i samotné jako například PHP třída. [6]

```
<!DOCTYPE html>
<html>
    <head>
        <title>PHP Test</title>
    </head>
    <body>
        <?php echo '<p>Hello World</p>'; ?>
    </body>
</html>
```

Obrázek 10- Syntaxe PHP [6]

Od verze PHP 4, kdy byly do jazyka doplněny objektově orientované prvky, začaly vznikat nejrůznější frameworky například Zend, Symfony, CakePHP nebo v Čechách velmi rozšířené Nette. [6]

V dnešní době se prakticky jinak než objektově orientovaně neprogramuje. Od verze PHP 5 se v PHP mnohem lépe pracuje s objekty, podobně jako tomu je v Javě. Objektově orientované programování je mnohem bližší reálnému světu objektů, umožňuje lepší znovu použitelnost kódů (není zde potřeba znovu objevovat Ameriku), výsledný kód je lépe strukturovaný a tím pádem i přehlednější. Tento způsob také umožňuje větší míru abstrakce a modularity, než procedurální způsob programování. [6]

Vzhledem k přehlednosti by každá PHP třída měla být uložena ve speciálním souboru určeném právě pro danou třídu. Základní struktura třídy se skládá z názvu třídy, atributů, případně getteru a setteru atributů, konstruktoru a metod. U každého atributu a metody je možné nastavovat viditelnost pomocí klíčových slov [6]:

- *Public* - znamená viditelnost všem uživatelům třídy.
- *Protected* - znamená přístupná pouze uživatelům třídy a jejím potomkům.
- *Private* - znamená přístupná pouze uvnitř třídy.

Na obrázku níže je zobrazena základní syntaxe třídy v PHP a následné vytvoření objektu a práce s objektem. [6]

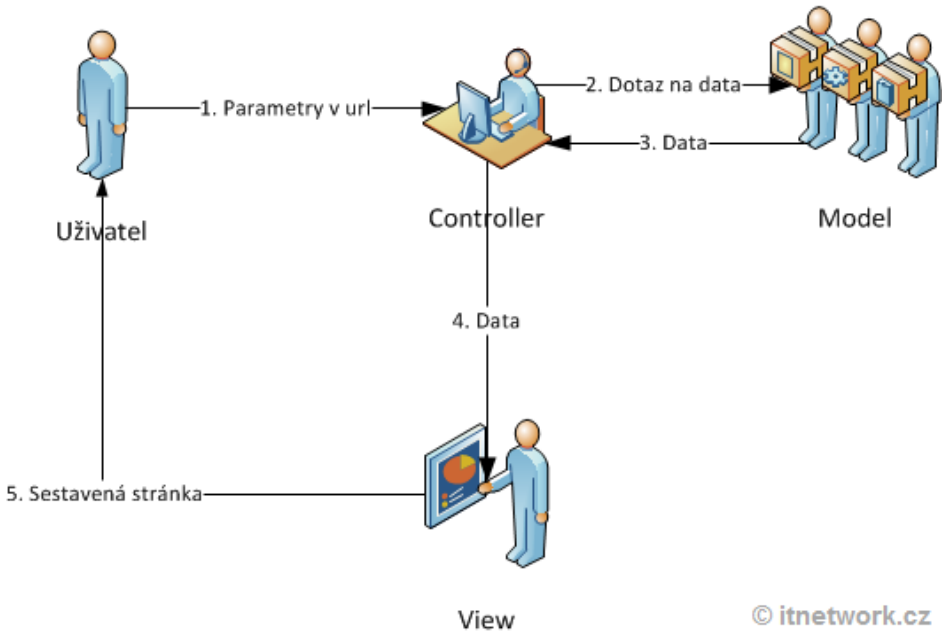
```php
class Person
{
    public $firstName;
    public $lastName;

    public function __construct($firstName, $lastName = '') { // optional second argument
        $this->firstName = $firstName;
        $this->lastName  = $lastName;
    }

    public function greet() {
        return 'Hello, my name is ' . $this->firstName .
               (($this->lastName != '') ? (' ' . $this->lastName) : '') . '.';
    }

    public static function staticGreet($firstName, $lastName) {
        return 'Hello, my name is ' . $firstName . ' ' . $lastName . '.';
    }
}

$he    = new Person('John', 'Smith');
$she   = new Person('Sally', 'Davis');
$other = new Person('iAmine');

echo $he->greet(); // prints "Hello, my name is John Smith."
echo '<br />';

echo $she->greet(); // prints "Hello, my name is Sally Davis."
echo '<br />';

echo $other->greet(); // prints "Hello, my name is iAmine."
echo '<br />';

echo Person::staticGreet('Jane', 'Doe'); // prints "Hello, my name is Jane Doe."
```

Obrázek 11- Syntaxe OOP v PHP [6]

3.11 MVC architektura

MVC je velmi oblíbený architektonický vzor, který se uchytil zejména na webu, ačkoli původně vznikl na desktopech. Je součástí populárních webových frameworků, jakými jsou např. Zend nebo Nette pro PHP, Ruby On Rail pro Ruby nebo MVC pro ASP .NET. [20]

Architektura MVC dělí aplikaci na 3 logické části tak, aby je šlo upravovat samostatně, a dopad změn byl na ostatní části co nejmenší. Tyto tři části jsou Model, View a Controller. Model reprezentuje data a business logiku aplikace, View zobrazuje uživatelské rozhraní a Controller má na starosti tok událostí v aplikaci a obecně aplikační logiku. [21]



Obrázek 12- MVC architektura [20]

3.12 Framework Nette

Nette Framework je stavěný tak, aby byl co nejpoužitelnější a nejvstřícnější. Jde o framework, s nímž je nejen snadné, ale i zábavné pracovat. Dává vám srozumitelnou a úspornou syntaxi, vychází vám vstříc při programování a debugování, nechává vás soustředit se na kreativní stránku vývoje. Eliminuje bezpečnostní rizika. Je vhodný pro e-shopy, wiki, blogy, CMS… [22]

Klíčové vlastnosti [22]:

- excelentní šablonovací systém
- bezkonkurenční ladící nástroje
- neobyčejně efektivní databázovou vrstvu
- důmyslné zabezpečení před zranitelnostmi
- moderní framework s podporou HTML5, AJAX nebo SEO
- s kvalitní dokumentací a nejaktivnější komunitou v ČR
- s vyzrálým a čistým objektovým návrhem
- vedoucím k dobrým návykům a dávajícím dostatek volnost

3.12.1 Sandbox – kostra aplikace

```
www/                      ← kořenový adresář webu
└── nette-blog/
    ├── app/              ← adresář webové aplikace
    │   ├── config/       ← konfigurační soubory
    │   ├── model/        ← třídy modelové vrstvy
    │   ├── presenters/   ← třídy presenterů
    │   ├── router/       ← konfigurace URL adres
    │   ├── templates/    ← šablony
    │   └── bootstrap.php ← spouštěcí soubor
    │
    ├── vendor/           ← knihovny pro vaší aplikaci
    │   └── nette/        ← jako například váš oblíbený framework :-)
    │
    ├── log/              ← zde naleznete chybové logy
    ├── temp/             ← místo pro dočasné soubory (cache, sessiony, atd.)
    ├── test/             ← adresář pro unit testy
    └── www/              ← místní kořenový adresář webu - pouze tento adresář je přístupný z internetu
```

Obrázek 13- Sandbox [23]

3.12.2 MVP

Nette je postaveno na architektuře MVC respektive MVP (Model, View, Presenter), což je prakticky stejné jako MVC, jen zde funkci controlleru zastupuje presenter.

Aplikace stojí na komponentách třech typů, které se v aplikaci dělí o 3 základní úlohy - řízení, logiku a výstup. Jen takto rozdělená aplikace je totiž přehledná a rozšiřitelná. [24]

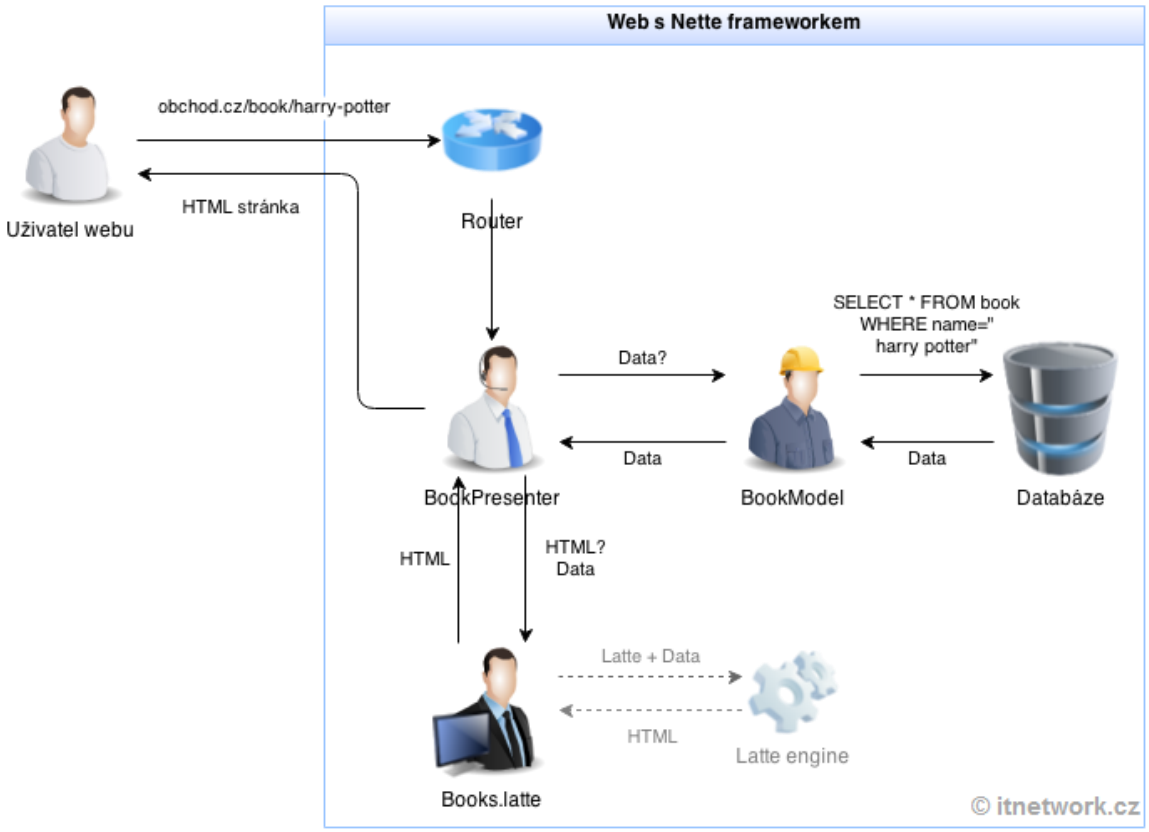
Presentery (Presenters), řízení - Presenter je komponenta, se kterou komunikuje uživatel. Předá jí parametry a ona mu vrátí HTML stránku. Presenter typicky parametry předá modelům, od kterých získá data. Tato data předá pohledům (šablonám), které data začlení do nějakého HTML kódu. Tento HTML kód pošle presenter uživateli do prohlížeče. Funguje tedy jako takový prostředník. [24]

Modely (Models), logika - Obsahují logiku aplikace, jako např. práci s databází nebo výpočty. Každá datová entita má většinou svůj model (uživatel, článek, komentář, ...).

Pohledy (V Nette Templates, česky šablony), výstup - Obsahují Latte šablony s HTML kódem. Latte je šablonovací jazyk, který do HTML šablon umožňuje vkládat data z PHP pomocí speciálních značek. [24]

Popis principu MVP na příkladu [24]:

1. Jako první se požadavek dostane k routeru. Ten podle adresy zjistí, že chceme něco s knihami a proto zavolá BookPresenter a předá mu zbytek URL.
2. BookPresenter se také podívá do parametrů, co se po něm chce a zjistí, že uživatel chce vypsat knihu Harry-Potter. Získá si tedy model BookModel, kterému sdělí, že chce tuto knihu. BookPresenter provádí pro knihy tzv. akce, v tomto případě zobrazení detailu. Stejně tak může např. knihu přidat nebo odstranit. Jednotlivé akce jsou v presenterech jednoduše reprezentované jako metody.
3. BookModel dostane v parametru název knihy, tu získá z databáze a vrátí.
4. BookPresenteru se vrátí data od modelu a tato data předá pohledu (šabloně).
5. Šablona obsahuje HTML stránku pro detail knihy a v ní Latte značky, do kterých se vloží data. Vložení těchto dat obstará automaticky Latte engine.
6. BookPresenteru přijde z šablony výsledné HTML a to pošle uživateli.
7. Uživateli se v prohlížeči zobrazí HTML stránka



Obrázek 14- Nette MVP [24]

3.13 PhpStorm

JetBrains PhpStorm je komerční, multi-platformní IDE pro jazyk PHP, vyvíjený na JetBrains' IntelliJ IDEA platformě. Díky univerzitní licenci ČZU, je možné tento komerční nástroj využít zcela zdarma. [25]

PhpStorm poskytuje rozhraní pro vývoj webových aplikací v PHP, HTML a JavaScriptu s dynamickou analýzou kódu, prevencí proti chybám a automatizovaným refaktoringem pro PHP a JavaScript. PhpStorm podporuje následující verze PHP 5.3, 5.4, 5.5 a 5.6. Dále také obsahuje generátory kódu a plnohodnotný SQL editor. [25]

Kromě PhpStorm existuje WebStorm, který není tak komplexí, a je vhodnější spíše pro vývoj front-endu. PhpStorm obsahuje vše co WebStorm + podporu PHP a databází, tudíž je vhodnější pro ucelený vývoj aplikací na webu. [25]

3.14 Google Maps API

Po úspěšném reverzivním inženýrství mashups jako chicagocrime.org a housingmaps.com, Google spustil v roce 2005 službu Google Maps API, které umožňuje vývojářům webových stránek integrovat Google Mapy do webových stránek. Tato služba je momentálně zcela zdarma i pro komerční účely s podmínkou, že nesmí překročit 25 000 přístupů za den. Google Maps API je prozatím bez reklam, nicméně Google si vyhrazuje právo do budoucna mít možnost vkládat reklamy. [26]

Pomocí Google Maps API je možné do webových stránek nejen vkládat standardní mapy, ale také je všemožně individualizovat, vkládat do nich značky, vrstvy atd. Dále je možné pomocí API vytvářet vyhledávací nástroje jako například našeptávač lokality nebo automatické doplňování formulářů na základě našeptávače. Google Maps využívají pro určování polohy souřadnice tzv. Latitude a Longitude. [26]

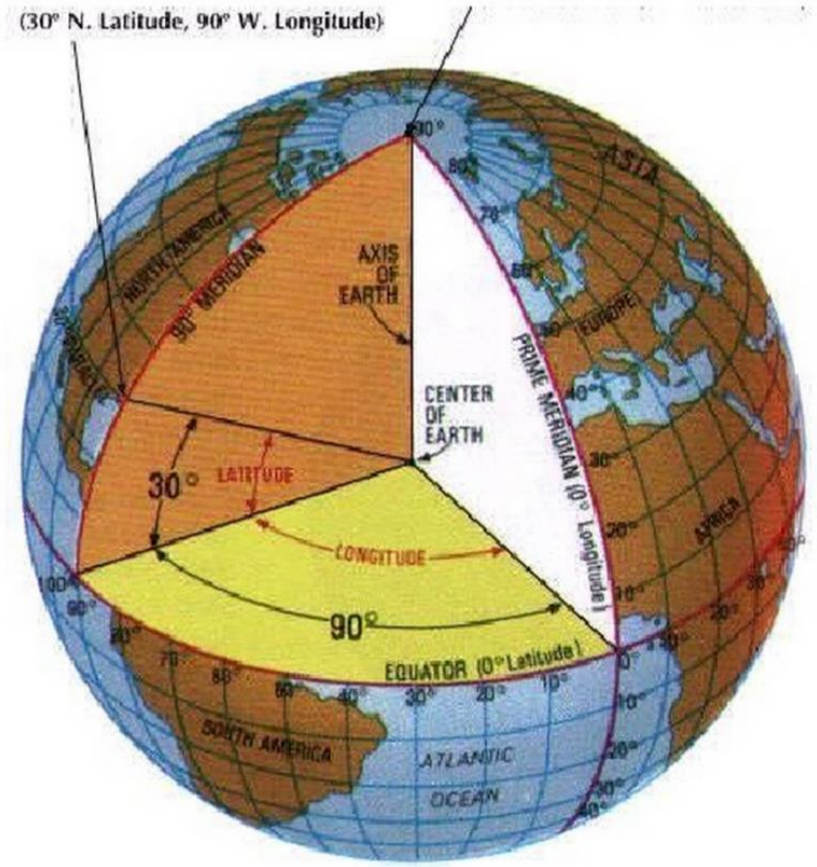
Na základě získaných souřadnic z Google Maps je možné za pomoci Haversine formuli spočítat rádius vzdálenosti mezi dvěma body na mapě, což je nejkratší vzdálenost po povrchu země. Jinými slovy získáme vzdušnou vzdálenost mezi dvěma body. [27]

Haversine formule [27]:

φ zde znamená latitude, λ znamená longitude, R znamená poloměr země (6,371km).

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$
$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$
$$d = R \cdot c$$

Obrázek 15- Haversine formula [27]



Obrázek 16- Latitude a Longitude [28]

4 VLASTNÍ PRÁCE - IMPLEMENTACE

4.1 Analýza

V první fázi je potřeba porozumět problému, zanalyzovat požadavky všech rolí uživatelů a zohlednit možné scénáře. Část portálu bude přístupná anonymním uživatelům bez přihlášení do informačního systému. Vzhledem k udržení serióznosti portálu a zajištění ochrany některých osobních údajů bude zbylá část portálu přístupná až po registraci a přihlášení do systému. Každá uživatelská role bude muset při registraci vyplnit základní přihlašovací údaje: uživatelské jméno a heslo. Jako uživatelské jméno bude sloužit e-mail, na který budou následně zasílány notifikační zprávy a obchodní sdělení.

V roli hospodyně systém vyžaduje, kromě přihlašovacích údajů, popis služeb a předností, které můžeme od hospodyně očekávat, tedy důvod proč bychom si měli jako klienti vybrat právě tuto hospodyni. Dále jsou vyžadovány kontaktní údaje, díky kterým bude klient schopen hospodyni oslovit. Kvůli větší přehlednosti a budoucí možnosti rozšíření jsou vyžadovány další údaje, tedy upřesnění jaké práce, a v jakých prostorách je hospodyně schopna vykonávat. Například zda hospodyně zvládá úklid pouze menších domácností (bytů) nebo i velkých rodinných domů včetně údržby zahrady. Krom výše zmíněných je vyžadováno rozpětí ceny hodinové sazby, časové možnosti, zkušenosti s touto prací v letech a vzdálenost, kterou je ochotna dojíždět.

V roli běžného uživatele resp. klienta se od systému očekává co nejjednodušší dosažení požadovaného cíle, tedy nalezení vhodné hospodyně. Na druhé straně musíme zajistit serióznost portálu a ochranu osobních údajů hospodyň, proto je zde nutná registrace uživatele. Na základě registrace uživatele a jeho přihlášení jsou pak klientovi zpřístupněny bližší údaje o hospodyni a v případě nenalezení vhodné kandidátky možnost zaslat soukromou zprávu.

Agentura může nabízet služby hospodyní, tedy poněkud alternativnější prezentace než je tomu zvykem.

4.2 Logický design

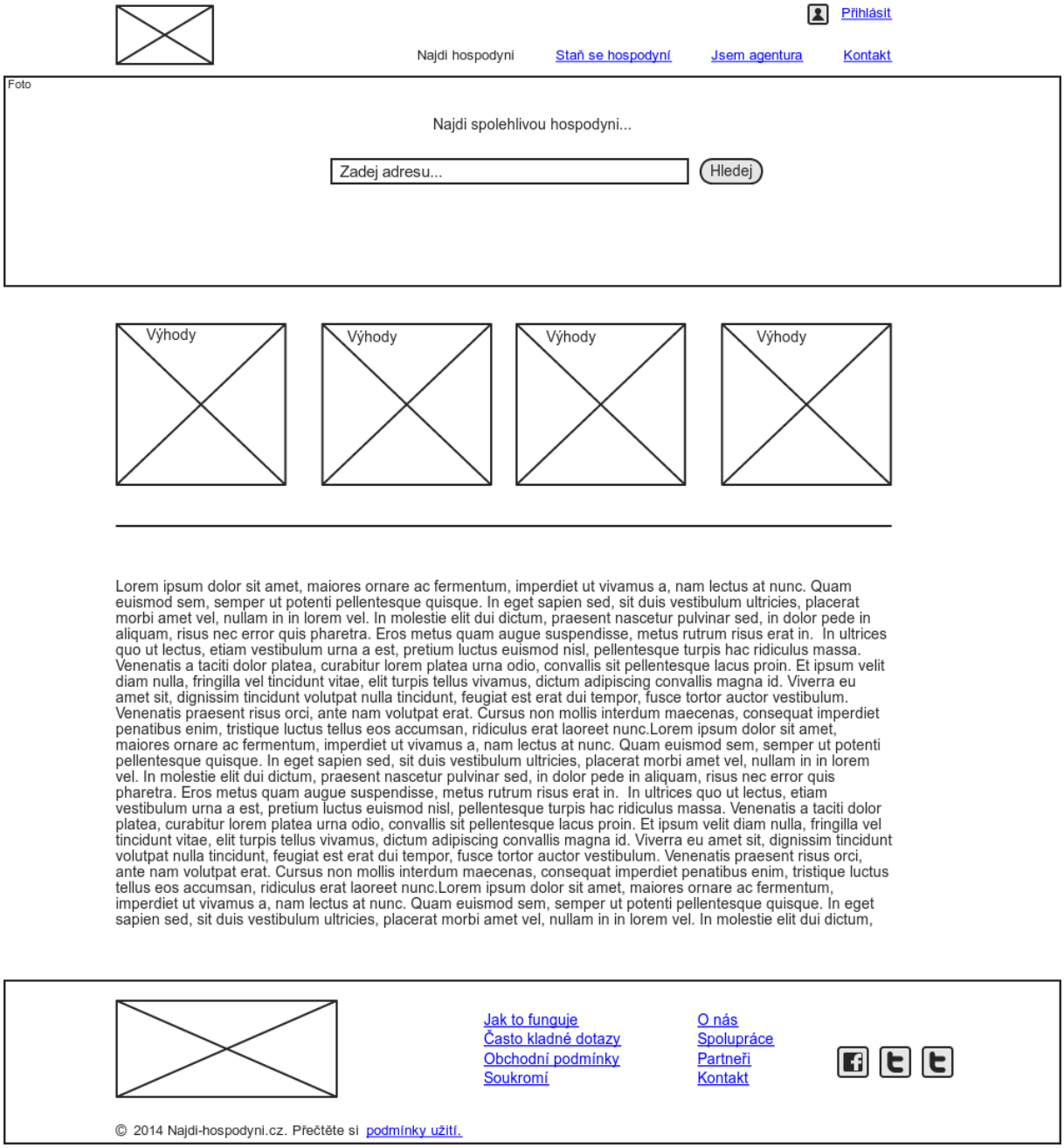
Na základě analýzy a požadavků se nejprve zpracovává logický design tzv. wireframe, v kterém se navrhuje základní rozložení jednotlivých stránek, umístění prvků a základní funkcionality. Logický design nám pomůže odhalit základní nedostatky našeho řešení ještě ve fázi, kdy není nakreslena jediná čárka designu, není navržená jediná tabulka databáze a není napsaný jediný řádek zdrojového kódu. Kdybychom některý z nedostatků odhalili až ve fázi samotné implementace jistě by to vedlo k pracnému předělávání velké části IS.

Titulní strana je v první řadě zaměřena na klienta, proto největší viditelnou část tvoří tematický obrázek související s úklidem domácnosti, aby upoutal klientovu pozornost. Přímo na úvodním obrázku je umístěn formulář pro vyhledání hospodyně na základě zadání adresy. Adresa se porovnává s API Google maps a na základě nalezení adresy se získají souřadnice polohy. Na základě těchto souřadnic se pak vyhledávají vhodné hospodyně. Přímo pod formulářem je umístěné jednoduché vysvětlení, jak to vlastně funguje, jak má klient postupovat a jaké tento portál přináší výhody.

V horní levé části se nachází logo a napravo od něj horizontální menu a tlačítko pro přihlášení nebo registraci. Po kliknutí na přihlášení se zobrazí formulář s inputy pro zadání uživatelského jména a hesla a možnost přihlášení.

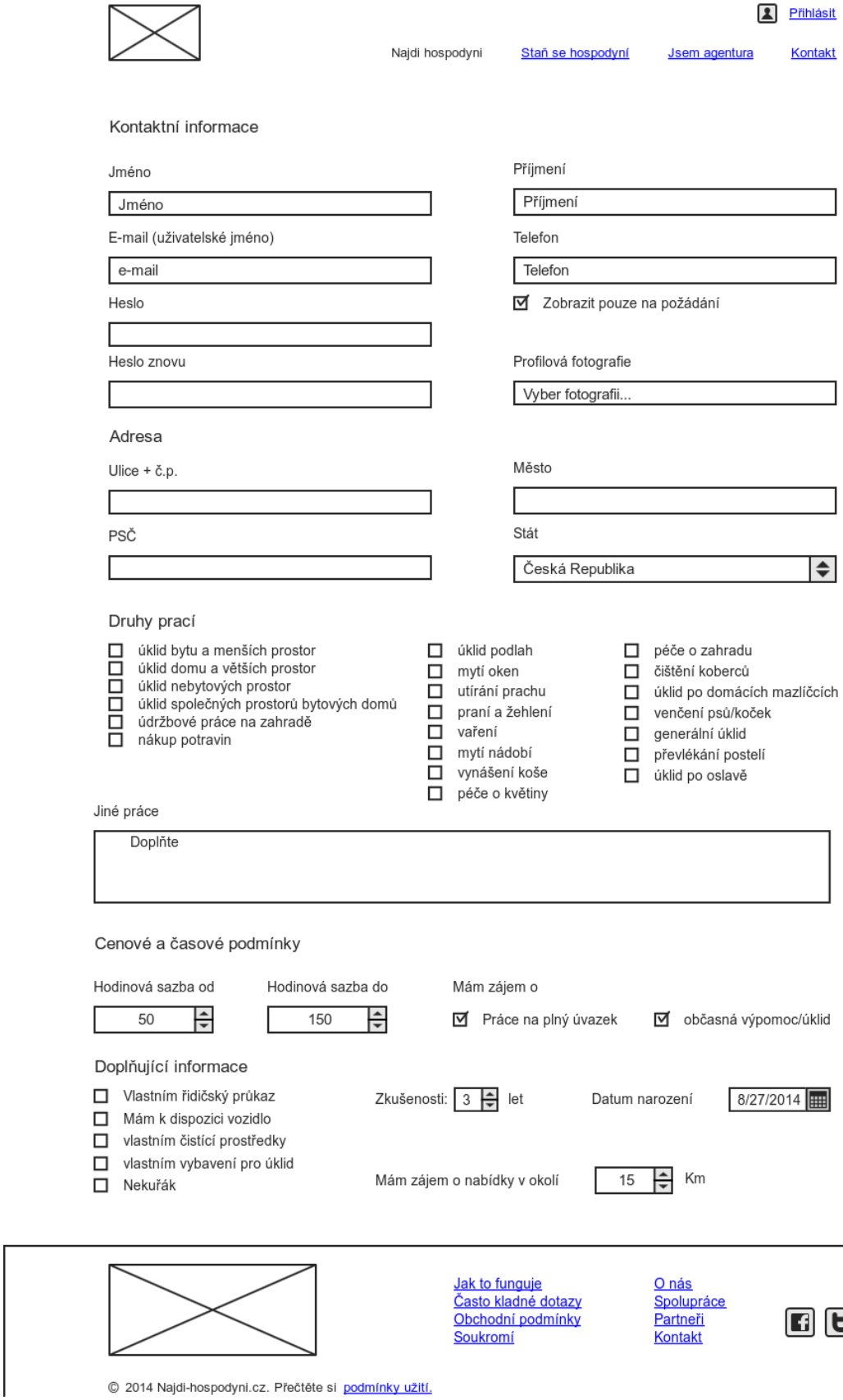
Níže na místě s nižší prioritou je umístěn krátký text s informací o portálu.

Zcela dole se nachází patička portálu s užitečnými odkazy na strany, ikony s odkazy na sociální sitě: facebook, twitter, google+ a pinterest.

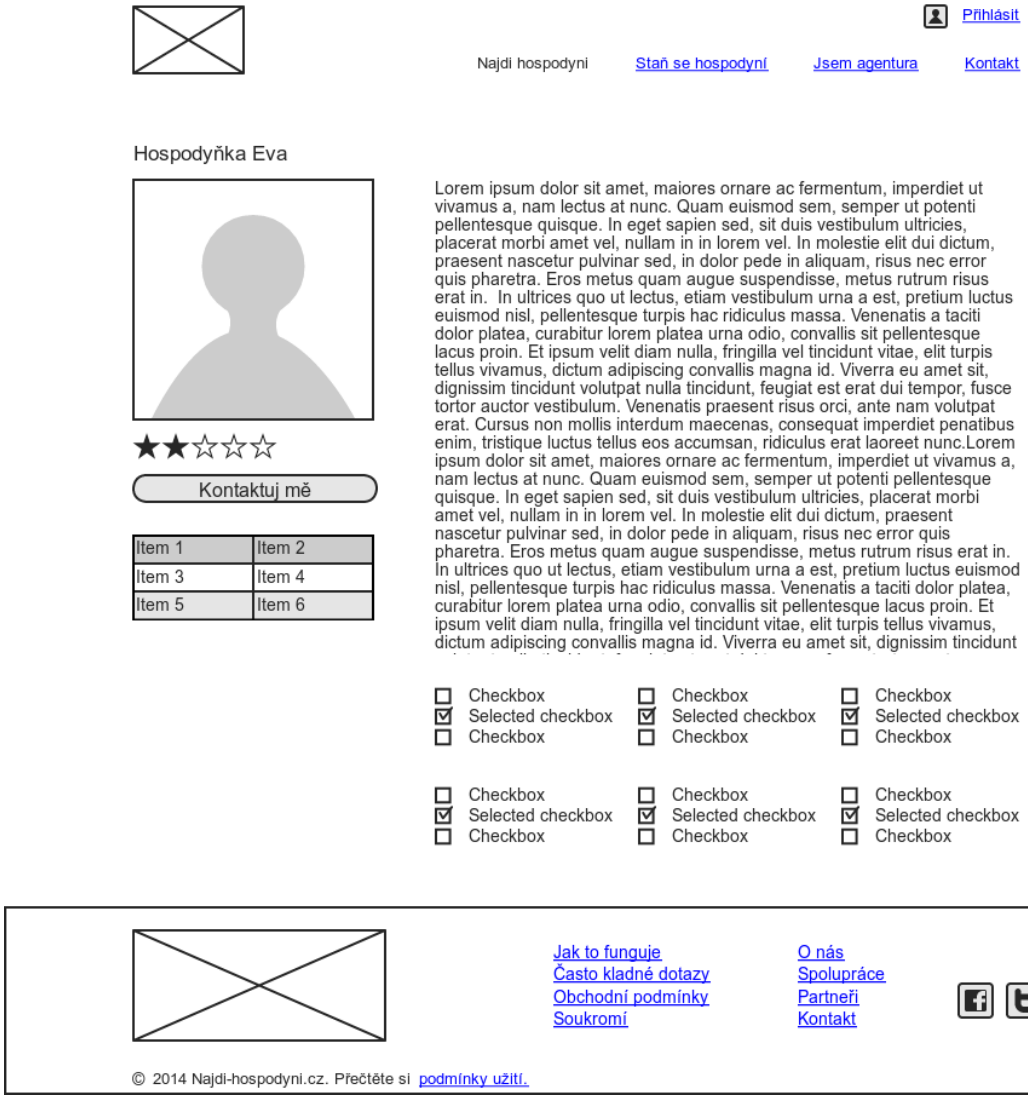


Obrázek 17- UI titulní strana (vlastní zpracování)

Obrázek 18- UI výsledek vyhledávání (vlastní zpracování)

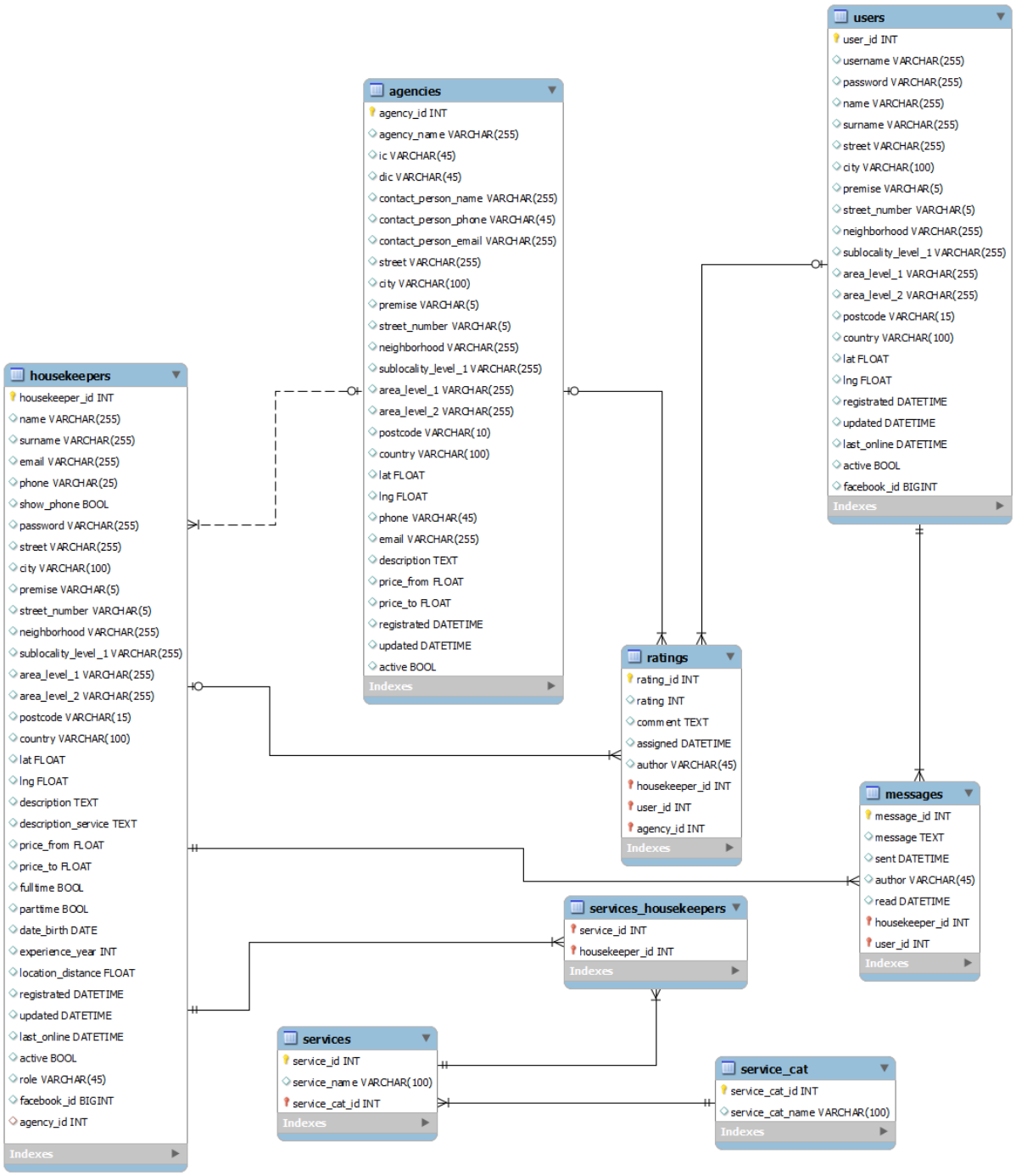


Obrázek 19- UI registrace hospodyně (vlastní zpracování)



Obrázek 20- UI profil hospodyně (vlastní zpracování)

4.3 Návrh MySQL



Obrázek 21- MySQL diagram (vlastní zpracování)

4.3.1 Popis databáze

- **Housekeepers** – informace o hospodyních, jako jsou osobní údaje, adresa, popis, stavy, ceny atd. Mohou tvořit podmnožinu agentury ve vztahu jako zaměstnanci.
- **Agencies** – obsahuje informace o dané agentuře, jako jsou kontaktní údaje, ceny a stavy.
- **Services** – obsahuje informace o nabízených službách, které daná hospodyně nabízí. Tvoří podmnožinu kategorií služeb, do kterých jsou pro lepší přehlednost řazeny.
- **Service_cat** – obsahuje seznam kategorií služeb.
- **Services_housekeepers** – vazební tabulka mezi hospodyněmi a službami, které nabízí.
- **Messages** – obsahuje komunikaci mezi hospodyněmi a uživateli. Dále obsahuje informace o datu a směru odeslání zprávy (hospodyně -> uživatel, uživatel -> hospodyně).
- **Ratings** – obsahuje hodnocení a komentáře (součástí hodnocení je i slovní komentář), informace o autorovi a datu přiřazení hodnocení. Tabulka je navržena i pro budoucí hodnocení klientů nebo agentur.

4.4 PHP – Nette

4.4.1 Struktura

Jak již bylo zmíněno v kapitole o Nette, tento framework je objektově orientovaný a podporuje MVC resp. MVP architekturu. V tomto duchu je vyvíjen i celý IS.

V modelu byla vytvořena třída Base, která je potomkem Nette\Object, a představuje základní třídu pro práci s databází. Všechny třídy, představující model, které pracují s databází, dědí metody právě od třídy Base. V třídě Base je vytvořen konstruktor pro připojení k databázi a metody pro nejčastější operace s databází.

Výpis metod třídy base:

- findAll – vrací všechny záznamy databáze
- findBy – vrací vyfiltrované záznamy podle vstupního pole
- findOneBy – vrací pouze jeden vyfiltrovaný záznam podle vstupního pole
- find – vrací záznam podle vstupního primárního klíče
- countBy – vrací počet nalezených záznamů vyfiltrovaných podle vstupního pole
- count – vrací počet všech záznamů
- update – upraví záznamy vyfiltrované podle vstupního pole
- insert – vloží záznam s daty uloženými ve vstupním poli
- delete – vymaže záznamy vyfiltrované podle vstupního pole

Díky dědění těchto metod není třeba znovu psát celé SQL dotazy nebo vytvářet metody pro obsluhu základních operací s databází. Výsledkem je velmi jednoduchý a dobře čitelný kód.

```php
/**
 * Edit params of housekeeper
 *
 * @param integer $id ID of client
 * @param array $values values from form
 */
public function editHousekeeper($id, $values)
{
    $this->tableName = 'housekeepers';

    $by['housekeeper_id'] = $id;
    $this->update($by, $values);
}
```

V konkrétní implementaci v třídě Housekeeper v metodě editHousekeeper na obrázku výše je vidět, jak snadno lze pak upravovat všechny parametry tabulky. Do metody je zasláno ID primárního klíče a pole s hodnotami z formuláře. Všechny názvy vstupních polí z formuláře korespondují s názvy sloupečků v tabulce, proto není nutné pole nijak upravovat. Zabezpečení proti SQL injection není třeba explicitně řešit, je již vyřešeno přímo v Nette databázové vrstvě.

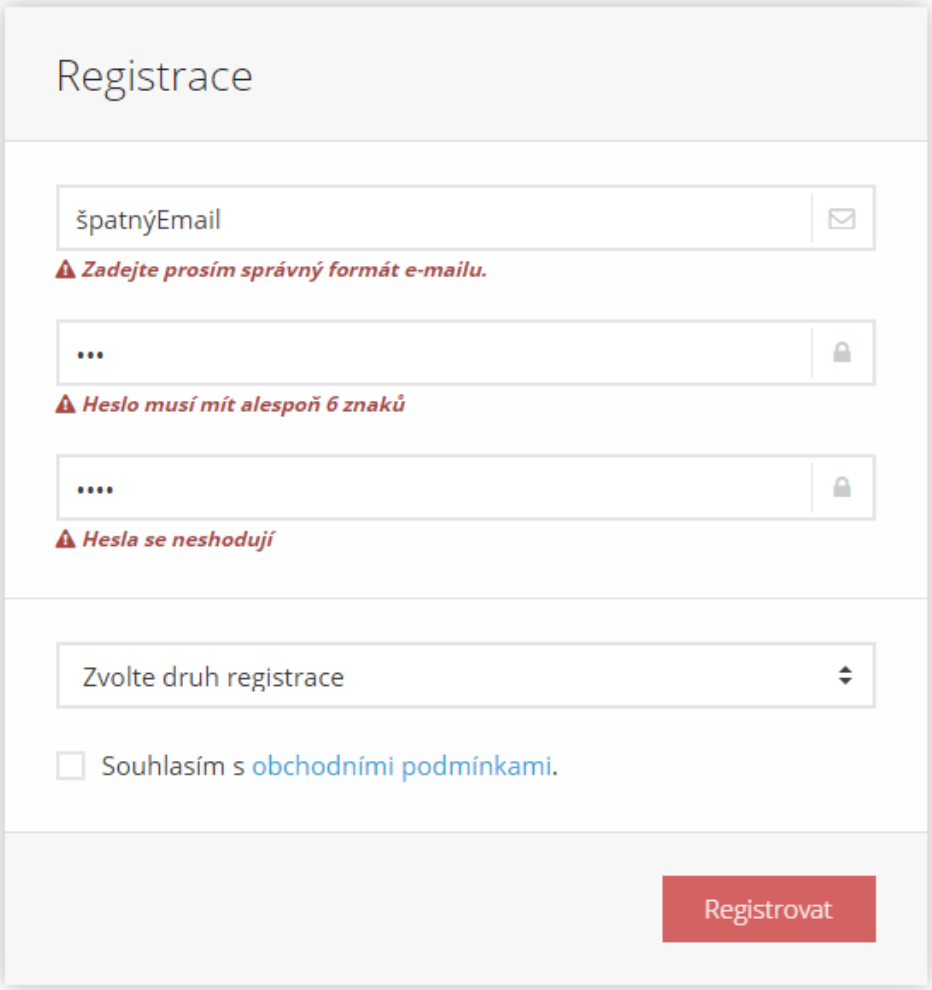
4.4.2 Formuláře

Nette podporuje velmi propracované řešení pro tvoru formulářů, které splňuje kritéria MVC architektury. Je tedy snadno použitelné v libovolných šablonách, zajišťuje bezpečnost a umožňuje velmi snadnou validaci vstupních elementů jak na straně serveru, tak na straně klienta.

Například komponenta pro registraci uživatele, dobře demonstruje sílu a jednoduchost validace pomocí Nette. Výhodou jsou již předpřipravená pravidla pro validaci e-mailu, minimální délku řetězce nebo shody hesel.

```
/**
     * Component of form for creating account
     *
     * @return Form
     */
    protected function createComponentSignUp()
{
    $optionsTypeReg = array();
    $optionsTypeReg[0] = "Zvolte druh registrace";
    $optionsTypeReg[1] = "Hledám hospodyni";
    $optionsTypeReg[2] = "Jsem hospodyně";

        if ($this->agencyEncodeId)
        {
            $optionsTypeReg = array();
            $optionsTypeReg[2] = "Jsem hospodyně";
        }

    $form = new Form;
    $form->addText('email')
            ->setRequired("Zadejte prosím váš e-mail.")
            ->addRule(Form::EMAIL, 'Zadejte prosím správný formát e-
mailu.');
    $form->addPassword('password')
            ->setRequired("Zadejte prosím heslo")
            ->addRule(Form::MIN_LENGTH, 'Heslo musí mít alespoň %d znaků',
6);
    $form->addPassword('passwordAgain')
            ->setRequired('Zadejte prosím heslo.')
            ->addRule(Form::EQUAL, 'Hesla se neshodují',
$form['password']);
    $form->addSelect('type_registration')->setItems($optionsTypeReg)
            ->setRequired('Zvolte prosím druhy registrace.');
    $form->addCheckbox('terms')
            ->setRequired("Pro registraci musíte souhlasit s podmínkami.");
        $form->addHidden('agency_id')
            ->setValue($this->agencyEncodeId);
    $form->addSubmit('send', 'Sign in');

    $form->onSuccess[] = array($this, 'signUpSucceeded');
    return $form;
}
```

Výslednou validaci bylo jen potřeba dostylovat pomocí CSS, aby výsledek dobře zapadal do designu celého webu.

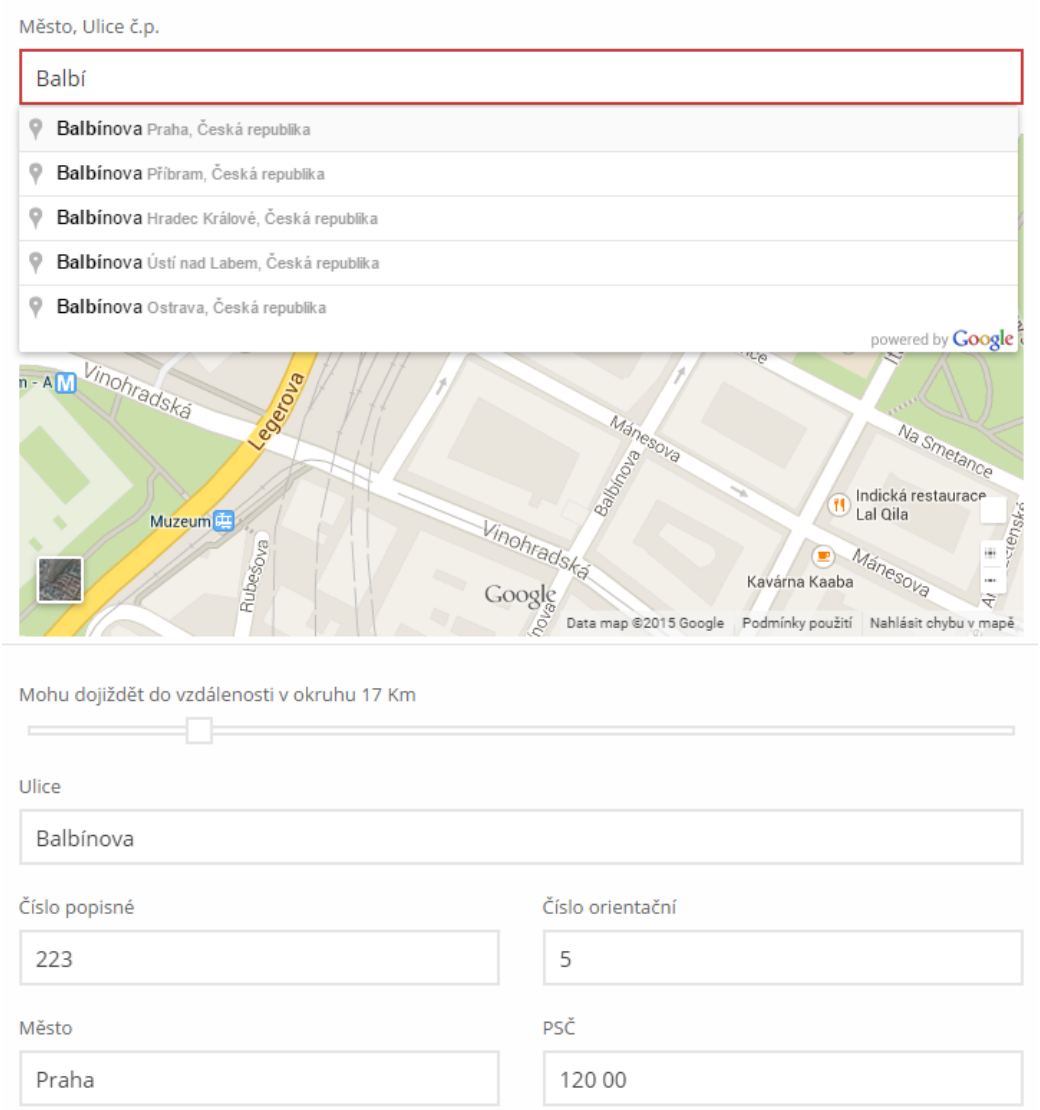


Obrázek 22- Validace registrace (vlastní zpracování)

4.4.3 Registrace hospodyň

Registrace hospodyně probíhá ve 2 krocích. V prvním kroku proběhne pouze jednoduchá registrace na základě e-mailu, hesla a volby, zda se chceme registrovat jako hospodyně, nebo zda hospodyni hledáme. Díky takto jednoduché registraci získáme první kontaktní údaje na hospodyni, a v případě, že by z nějakého důvodu hospodyně nedokončila druhý krok registrace, můžeme získat feed back, proč registraci nedokončila, a případně podle potřeby přepracovat registraci nebo business plan. Pro zajištění bezpečnosti uložených hesel v databázi, byla využitá předpřipravená hashovací funkce, založená na algoritmu bcrypt.

V druhém kroku je již hospodyně přihlášena, a vyzvána k doplnění veškerých potřebných informací. Pro zjednodušení vyplňování je zde použito Google Maps API, které na základě vyplnění části adresy (našeptávače) samo doplní potřebné údaje o adrese. Zároveň je na Google mapě automaticky označeno místo adresy pomocí tzv. markeru, se kterým lze libovolně pohybovat pomocí myši a upřesnit tím polohu hospodyně. Pod mapou se jednoduše pomocí posuvníku určuje vzdálenost, kterou je hospodyně ochotna dojíždět za svými klienty. Zmiňovaná poloha a vzdálenost je pro funkčnost celého IS klíčová, protože prakticky základním kamenem vyhledávání hospodyní je výpočet vzdálenosti mezi místem úklidu a bydlištěm hospodyně.



Obrázek 23- Registrace Google Maps API (vlastní zpracování)

4.4.4 Přihlášení

Přihlášení funguje obecně stejně jak pro hospodyně, tak pro klienty. Svou roli si uživatelé volí výběrem v select inputu. Systém pro proces přihlášení, kontrolu stavu přihlášení a nastavení délky přihlášení, byl z důvodu bezpečnosti převzat z Nette, které uchovává informace o přihlášení uživatele v Cookies.

Cookies jsou malé (Max. 4 KB) dočasné soubory, které se v případě povolení ukládají v prohlížeči. Každá Cookies má uloženy následující informace: jméno, data, doba expirace, doména, cesta a secure. Právě díky cookies je možné snadno nastavit dobu expirace přihlášení i v případě vypnutí PC na dobu v řádů dní i delší. Pokud uživateli nevypršeli Cookies a vrátí se na danou webovou stránku, tak si server přečte uživatelovi Cookies, které sám kdysi vytvořil a uživatele autentizuje.

Pro lepší informovanost o aktivitě hospodyň v IS byla vytvořena funkce, která v případě přihlášené hospodyně ukládá poslední čas její aktivity. Tuto funkcionalitu obstarává třída Housekeeper pomocí metody setLastOnline.

```
/**
 * Set last online active
 *
 * @param $id
 */
public function setLastOnline($id)
{
    $this->tableName = 'housekeepers';

    $by['housekeeper_id'] = $id;
    $values['last_online'] = date("Y-m-d H:i:s");
    $this->update($by, $values);
}
```

Pro zjištění délky neaktivity, která může být při výběru vhodné hospodyně pro některé klienty klíčová, se používá třída Housekeeper a metoda getLastOnline.

```php
/**
 * Return time last online
 *
 * @param $id Housekeeper ID
 * @return string
 */
public function getLastOnline($id)
{
    $this->tableName = 'housekeepers';

    $row = $this->findOneBy(array('housekeeper_id' => $id));
    $actDate = date('Y-m-d H:i:s');
    $lastOnline = date($row->last_online);
    $diff = $this->dateTime->timeDiff($actDate, $lastOnline);
    return $this->dateTime->getTimeInConvenientUnits($diff);
}
```

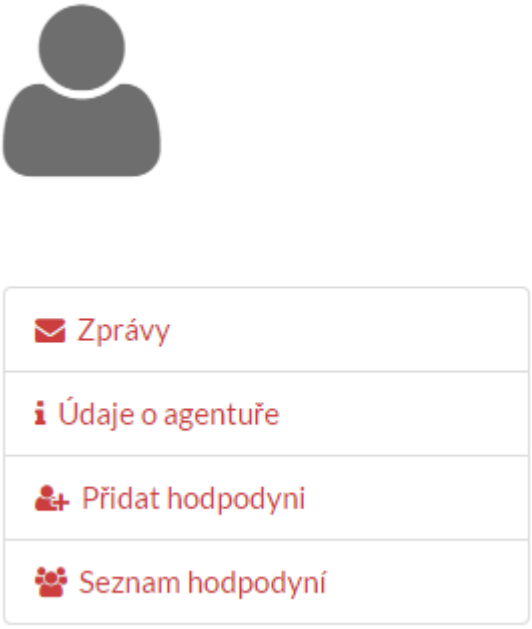
Metoda vypočítává časový rozdíl mezi datem poslední aktivity a aktuálním časem. Pro lepší čitelnost běžným lidem metoda vrací výsledek ve vhodných jednotkách, které budou člověku srozumitelné. Nebylo by vhodné zobrazovat například čas „Naposledy online před 3 569 852s.“. Za tímto účelem byla vytvořena metoda getTimeInConvenientUnits. Do metody vkládáme vypočtený rozdíl mezi datem poslední aktivity a aktuálním časem v sekundách. Metoda pak porovnává čas, a pokud je množství dané jednotky zobrazitelné v celých číslech jednotky vyšší, pak metoda použije pro zobrazení vyšší jednotky. Například pokud je výsledek 60 sekund metoda zobrazí 1 minutu.

```php
/**
 * Convert time in seconds to convenient form
 *
 * @param $seconds
 * @return string
 */
public function getTimeInConvenientUnits($seconds)
{
    if ($seconds < 60)
    {
        return $seconds." s";
    }
    elseif ($seconds >= 60 && $seconds < 3600)
    {
        $output = $this->secondsToMinutes($seconds);
        return round($output)." min.";
    }
    elseif ($seconds >= 3600 && $seconds < 86400)
    {
        $output = $this->minutesToHours($this-
>secondsToMinutes($seconds));
        return round($output)." hod.";
    }
    elseif ($seconds >= 86400 && $seconds < 2592000)
    {
        $output = $this->hoursToDays($this->minutesToHours($this-
>secondsToMinutes($seconds)));
        if (round($output) == 1)
        {
            return round($output)." dnem";
        }
        else
        {
            return round($output)." dny";
        }
    }
    else
    {
        return 'déle než měsícem';
    }
}
```

4.4.5 Registrace agentur

Vzhledem k tomu, že portál není určen pouze pro soukromé hospodyně, je zde možnost registrovat se jako agentura. První krok registrace je zcela stejný jako běžné hospodyně. V druhém kroku se již uvádějí údaje o firmě.

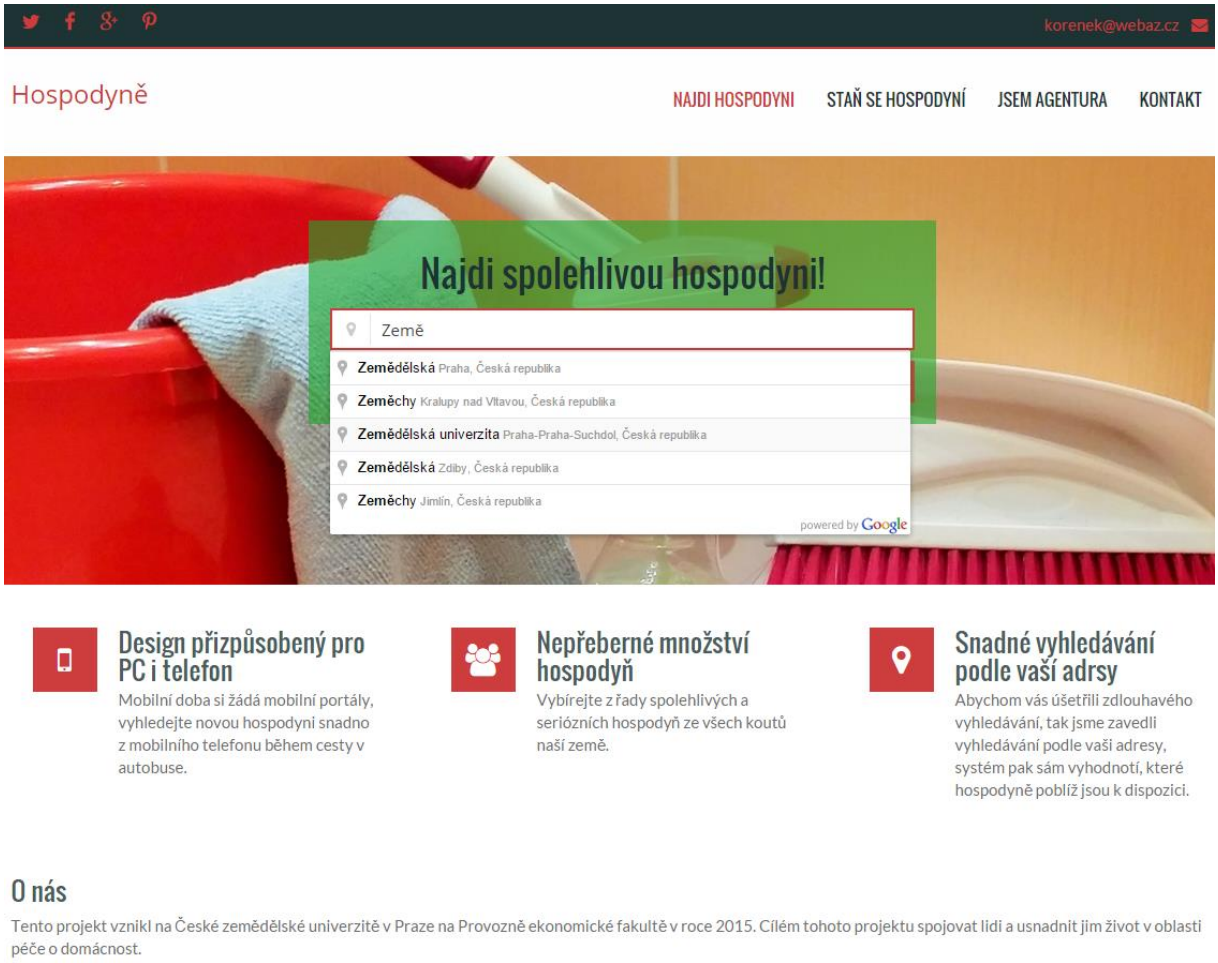
Po úspěšné registraci agentury, je možné přidávat nové hospodyně pod záštitou agentury, prohlížet seznam hospodyň nebo editovat údaje firmy.



Obrázek 24 - Menu agentury (vlastní zpracování)

4.4.6 Vyhledávání hospodyní pomocí vzdálenosti souřadnic

Jak již bylo výše zmíněno, podstatou tohoto IS je vyhledávání na základě vzdálenosti poptávajícího klienta a bydliště hospodyně. Tomuto faktu je i přizpůsobena hlavní strana, která nabádá klienta k vyplnění své adresy. Adresa se vyplňuje do vyhledávacího pole, která je pro pohodlí klienta doplněno našeptávačem adresy. Po zvolení adresy se předají souřadnice místa (latitude, longitudes) do skrytých inputů, aby mohli být po odeslání formuláře předány třídě, která výsledky vyhledávání zpracovává.



Obrázek 25 -homepage vyhledávač pomocí Google map (vlastní zpracování)

Jak bylo popsáno výše v kapitole o Google Maps API, veškerá místa na mapě se označují na základě dvou souřadnic - Latitude a Longitude. Pro určení vzdálenosti hospodyně od souřadnic adresy poptávajícího klienta je použita Haversinova formule, která je aplikována přímo do SQL dotazu. Na základě získané vzdálenosti mezi dvěma body, pak zjišťujeme, zda poptávané místo úklidu je v rádiusu vzdálenosti, kam je hospodyně ochotna dojíždět, pokud ano, tak je daná hospodyně zařazena do výsledku vyhledávání. Předpokládá se, že pro hospodyni i poptávajícího klienta je nejvýhodnější nejkratší vzdálenost, proto je výsledek řazen podle vzdálenosti od nejmenší.

```
/**
 * Return housekeepers by accepted distance from inserted lat + lng
 *
 * @param $lat
 * @param $lng
 * @return Array
 */
public function searchByGeolocation($lat, $lng)
{
    $distance = "((( 6371 * acos( cos( radians(".$lat.") ) * cos(
radians( lat ) ) * cos( radians( lng ) -
    radians(".$lng.") ) + sin( radians(".$lat.") ) * sin( radians( lat )
) ) ) ) * 1.609344 ) AS distance_between";
    $having = "HAVING distance_between < location_distance";

    $params = array();

    $results = $this->database->query("SELECT *, $distance FROM
housekeepers $having ORDER BY distance_between");
    while ($row = $results->fetch())
    {
        $params[$row['housekeeper_id']]['name'] = $row['name'];
        $params[$row['housekeeper_id']]['surname'] = $row['surname'];
        $params[$row['housekeeper_id']]['description'] =
$row['description'];
        $params[$row['housekeeper_id']]['city'] = $row['city'];
        $params[$row['housekeeper_id']]['sublocality_level_1'] =
$row['sublocality_level_1'];
        $params[$row['housekeeper_id']]['distance_between'] =
$row['distance_between'];
        $params[$row['housekeeper_id']]['price_from'] =
$row['price_from'];
        $params[$row['housekeeper_id']]['price_to'] = $row['price_to'];
        $params[$row['housekeeper_id']]['rating'] = $this-
>getRating($row['housekeeper_id'], 0);
        $params[$row['housekeeper_id']]['img_path'] = $this-
>getPathProfileImage($row['housekeeper_id']);
    }
    return $params;
}
```

4.4.7 Kontaktní formulář

Pro kontaktní stránku byl zvolen moderní design s Google maps přes celou šířku, která perfektně zapadá do konceptu celého webu. Zmiňované Google maps byly vloženy pomocí jednoduchého frameworku gmaps.js, který usnadňuje práci s Google Maps a výsledná implementace je velice elegantní.

```
<script>
    $(document).ready(function(){
        map = new GMaps({
            div: '#head',
            scrollwheel: false,
            lat: 37.530814,
            lng: -122.262318,
        });

        var marker = map.addMarker({
            lat: 37.530814,
            lng: -122.262318,
            title: 'hospodyne.cz'
        });
    });
</script>
```

Samotný kontaktní formulář je řešen pomocí Nette metod pro tvorbu formulářů a odesílání emailů. Pro odesílání emailů jsou připraveny dvě knihovny: Nette\Mail\Message a Nette\Mail\SendmailMailer.

```php
/**
 * Component of form for sending message
 *
 * @return Form
 */
protected function createComponentQuestion()
{
    $form = new Form;
    $form->addText('name')->setRequired('Zadejte prosím vaše jméno.');
    $form->addText('email')->setRequired("Zadejte prosím váš e-mail.")
        ->addRule(Form::EMAIL, 'Zadejte prosím správný formát e-
mailu.');
    $form->addTextArea('message')->setRequired('Vyplňtě prosím váš
dotaz.');
    $form->addSubmit('submit');

    $form->onSuccess[] = array($this, 'questionSucceeded');
    return $form;
}

/**
 * Send e-mail message
 *
 * @param $form
 * @param $values
 */
public function questionSucceeded($form, $values)
{
    $mail = new Message;
    $mail->setFrom($values->name.' <'.$values->email.'>')
        ->addTo('adam.korenek@gmail.com')
        ->setSubject('Dotaz - hospodyne.cz')
        ->setBody($values->message);

    $mailer = new SendmailMailer;
    $mailer->send($mail);
}
```

4.5 jQuery, AJAX a Nette

Komunikace mezi hospodyněmi a klienty je zajištěna pomocí jednoduchého formuláře, který se otevírá v modálním okně. Funkčnost modálního okna zajišťuje plugin, postavený na jQuery, nazývaný Fancybox. Díky tomuto pluginu je možné do modálního okna načítat nejen formuláře, ale i obrázky, videa (např. z youtube.com) a jiná média.

Pro kvalitní UX a pohodlné ovládání se využívá technologie AJAX popsaná výše v metodice. Principem odesílání zpráv je jQuery metoda, která vyčkává na události, kdy uživatel odešle formulář. Ve chvíli odeslání formuláře metoda zpracuje veškerá data z formuláře a na pozadí je odešle na server.

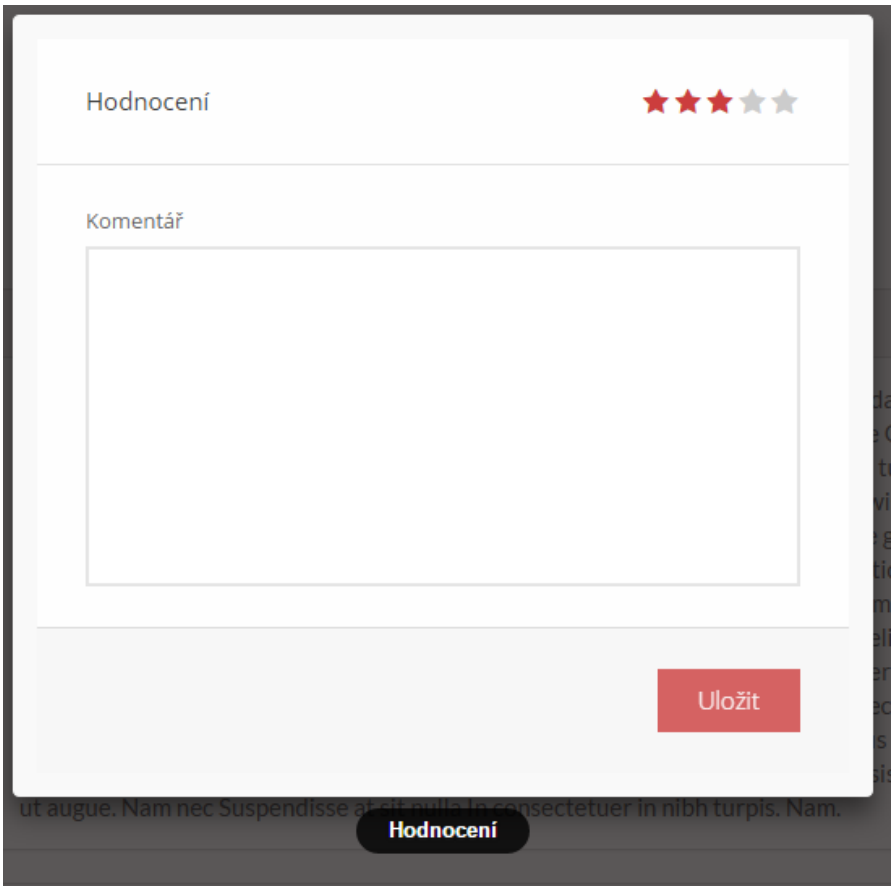
```
<script>
    $("#messageForm").submit(function () {
        $.ajax({
            type: "POST",
            url: {link sendMessage!},
            data: $("#messageForm").serialize(),
            datatype: "json",
            success: function (data) {
                if (data.success == 'Odesláno') {
                    $.fancybox.close();
                } else {
                    alert(data.error);
                }
            }
        });
        return false;
    });
</script>
```

Na serveru tyto data zpracovává handler (metoda presenteru), který prvně kontroluje, zda je uživatel přihlášen, aby mohl identifikovat odesilatele zprávy. V druhém kroku kontroluje, zda přihlášený uživatel, hospodyně nebo klient. Z obchodního hlediska je dovolená jen komunikace mezi hospodyní a klientem. Pokud je vše v pořádku, tak handler předá data modelu Messages, ten již provede uložení zprávy do databáze.

Handler vrací prohlížeči informaci o výsledku operace ve formě Json. Pokud vše proběhlo v pořádku, tak se Fancybox uzavře a uživatel bez obnovení stránky pokračuje v prohlížení. Pokud dojde k nějaké chybě během odesílání, pak je uživatel upozorněn na chybu.

```
/**
 * Handler for sending messages
 */
public function handleSendMessage()
{
    if($this->user->loggedIn)
    {
        $userData = $this->getUser()->getIdentity()->getData();
    }
    else
    {
        $loginWarning = array('error'=>'Pro zaslání zprávy se musíte
přihlásit');
        $this->sendResponse(new
Nette\Application\Responses\JsonResponse($loginWarning));
    }

    if (($_REQUEST['author'] == $this->authorMsgHousekeeper) and
($_REQUEST['housekeeper_id'] != $userData['housekeeper_id']))
    {
        $comunicationWarning = array('error'=>'Je povolena pouze
komunikace hospodyně - klient.');
        $this->sendResponse(new
Nette\Application\Responses\JsonResponse($comunicationWarning));
    }
    elseif (($_REQUEST['author'] == $this->authorMsgUser) and
($_REQUEST['user_id'] != $userData['user_id']))
    {
        $comunicationWarning = array('error'=>'Je povolena pouze
komunikace hospodyně - klient.');
        $this->sendResponse(new
Nette\Application\Responses\JsonResponse($comunicationWarning));
    }
    else
    {
        $this->context->messages
            ->sendMessage($_REQUEST['user_id'],
$_REQUEST['housekeeper_id'], $_REQUEST['author'], $_REQUEST['text-
message']);
        $success = array('success'=>'Odesláno');
        $this->sendResponse(new
Nette\Application\Responses\JsonResponse($success));
    }
}
```

Stejně jako pro zprávy i pro hodnocení hospodyň bylo využito Fancyboxu a AJAX ukládání hodnocení skrze handler. K hodnocení je možné přidat i krátký komentář pro odůvodnění zvoleného hodnocení. Aby nedocházelo k vzájemnému poškozování mezi hospodyněmi, nebo záměrnému ovlivňování výsledků, je zde povinné být přihlášen pod klientským účtem. Každý klient může hodnotit individuální hospodyni pouze jednou.



Obrázek 26 - Hodnocení hospodyň (vlastní zpracování)

Tabulka ratings byla navržena tak, aby bylo možné do budoucna udělit hodnocení mezi všemi uživateli portálu, ne jen klient hospodyni. Hodnocení hospodyň obstarává metoda setRating. Tato metoda nejprve kontroluje, zda již klient zvolenou hospodyni nehodnotil, a pokud ano, tak hodnocení přepíše, aby bylo uchováno pouze to nejaktuálnější.

```php
/**
 * Set rating of housekeeper
 *
 * @param $housekeeperId
 * @param $userId
 * @param $rating
 * @param $comment
 */
public function setRating($housekeeperId, $userId, $rating, $comment)
{
    $this->tableName = 'ratings';

    $values = array();
    $values['rating'] = $rating;
    $values['comment'] = $comment;
    $values['author'] = $this->authorUser;
    $values['housekeeper_id'] = $housekeeperId;
    $values['user_id'] = $userId;
    $values['assigned'] = date('Y-m-d H:i:s');

    if ($this->countBy(array('user_id' => $userId, 'housekeeper_id' =>
$housekeeperId)) > 0)
    {
        $data = $this->findOneBy(array('user_id' => $userId,
'housekeeper_id' => $housekeeperId));
        $this->update(array('rating_id' => $data['rating_id']),
$values);
    }
    else
    {
        $this->insert($values);
    }
}
```

Následný výpočet hodnocení se provádí pomocí metody getRating, která vrací zaokrouhlený průměr všech hodnocení.

```php
/**
 * Return rounded rating
 *
 * @param $id
 * @param int $round
 * @return bool|float|Nette\Database\Row
 */
public function getRating($id, $round = 0)
{
    $rating = $this->database
        ->query("SELECT AVG(rating) FROM ratings WHERE housekeeper_id =
$id AND author = '".$this->authorUser."'")
        ->fetch();
    $rating = round($rating[0], $round);
    return $rating;
}
```

5 ZÁVĚR

Agregátor se zdařilo zprovoznit a otestovat pro všechny role uživatelů, tedy klienty co hledají hospodyni, hospodyně i agentury. Front-end portálu je vytvořen pomocí HTML5, CSS a jQuery s využitím frameworku Bootstrap.

Funkční část agregátoru je postavena na Nette, které klade důraz na bezpečnost. Nabízí komplexní nástroj pro tvorbu webových aplikací, a díky již předpřipravené souborové a datové struktuře je vhodný jako základní kostra projektu s moderní architekturou MVC. S databázovou vrstvou založenou na NotORM frameworku, která je přímo implementovaná v Nette se pohodlně pracuje a zároveň zajišťuje ochranu proti SQL injections. Šablonovací systém Latte, který je syntaxí velice podobný Smarty, zpřehledňuje HTML kód, nabízí řadu maker pro formátování vypisovaných proměnných a zajišťuje bezpečnost aplikace.

Celkově mi Nette, v mnoha ohledech, usnadnilo práci. Například validace formulářů lze řešit pouze jednou v presenteru a Nette se samo postará o validaci na straně klienta (pomocí JavaScriptu) a zároveň na straně serveru.

Nicméně asi jako každý Framework, má i Nette svá úskalí. Konkrétně největší problém Nette je jeho rozšíření pouze v ČR. I když je Nette framework s nejsilnější českou komunitou, tak je celkem problém dohledat na internetu tutoriály vztahující se k poslední verzi, což je mimochodem problém i dokumentace. S tím souvisí i řešení problémů, na které jsem při vývoji narážel. Jen velmi těžko jsem hledal na internetu dostupné informace o řešení problému. S tímto problémem jsem se u nástrojů používaných ve světě nesetkal. Stačilo mi dát vyhledat daný problém Googlem, a z pravidla jeden z prvních třech odkazů na stackoverflow.com můj problém rychle vyřešil.

V kontaktu byla použita smyšlená adresa a ostatní kontaktní údaje.

Na tomto školním projektu, jsem si vyzkoušel návrh a realizaci IS na webu a již nyní mě napadá spoustu rozšíření nebo vylepšení, například na výsledek vyhledávání hospodyň by bylo hezké doplnit načítání dalších hospodyň pomocí „infine scroll“ nebo doplnit filtr pro filtrování konkrétních služeb.

6 SEZNAM POUŽITÝCH ZDROJŮ

x

1. KAPLANOVÁ, D. Novinky.cz. [Povolání: Hospodyně na plný úvazek] In: *Povolání: Hospodyně na plný úvazek* [online]. 2009 [cit. 2014-08-30]. Dostupné z: http://www.novinky.cz/zena/styl/168194-povolani-hospodyne-na-plny-uvazek.html
2. Adaptic. Apache Server [online]. Dostupné také z: http://www.adaptic.cz/znalosti/slovnicek/apache-server/
3. What is XAMPP? Apache *Friends* [online]. 2015 [cit. 2015-Leden-03]. Dostupné z: https:// www.apachefriends.org/index.html
4. PHP MySQL Learners. How does php *works [online]*. 2014 [cit. 2015-leden-03]. Dostupné z: https:// phpmysqllearners.wordpress.com/tag/how-does-php-works/
5. WEST, M. HTML5 Foundation. *2013. ISBN 978-1-118-43269-3.*
6. PHP [online]. San Francisco (CA): Wikimedia *Foundation,* 2001 [cit. 2015-01-20]. Dostupné z: http:// en.wikipedia.org/wiki/PHP
7. TEAGUE, J. C… In*:* CSS3: Visual QuickStart Guide [online]. Peachpit Press, 2012, s. 464 [cit. 2015-01-15]. ISBN 978-0-321-88893-8. Dostupné z: https://books.google.cz/ books?id=_xB0PyT9Y24C&printsec=frontcover&dq=css3+visual+quickstart+guide&hl=cs&sa=X&ei=NBrqVLj9J8XAPJ60gOAG&ved=0CB8Q6AEwAA#v=onepage&q&f=false
8. CSS syntax. *In: w3schools.com* [online]. Dostupné také z: http://www.w3schools.com/css/css_syntax.asp
9. Bootstrap. About [online]. [cit. 2014-prosinec-26]. Dostupné z: *http://*getbootstrap.com/about/
10. PROCHÁZKA, T… In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2012-05-02]. Dostupné z: http://upload.wikimedia.org/wikipedia/commons/thumb/0/02/ Complete.png/373px-Complete.png
11. ULLMAN, L. Modern JavaScript: Develop and Design. *Peachpit* Press, 2012. ISBN 9780321812520.
12. JANOVSKÝ, D. Jak psát web. Úvod do JavaScriptu [online]. Dostupné také z: http://www.jakpsatweb.cz/ javascript/javascript-uvod.html
13. LACKO, L. Ajax hotová řešení. Brno: Computer press, a.s. 2008. *ISBN 978-80-251-2108-5.*
14. W3Schools. AJAX Introduction [online]. Dostupné také z: http://www.w3schools.com/Ajax/ ajax_intro.asp
15. JONATHAN, C. Mistrovství v jQuery. Computer press, a.s. 2013. ISBN 978-80-251-4103-8.
16. jQuery Syntax - action on HTML elements. In: Reference Designer [online]. [cit. 2015-01-15]. Dostupné z: http://www.referencedesigner.com/tutorials/jquery/jq_4.php
17. VOSTROVSKÝ, V. Vytváření databází v Oracle. Praha: Česká zemědělská univerzita v Praze, 2009. ISBN 978-80-213-1191-6.
18. MySQL Workbench OSS. In: Softoware [online]. 15. 11. 2014. Dostupné také z: http://cs.softoware.net/ apps/download-mysql-workbench-oss-for-mac.html
19. WAGMOB. Learn PHP Programming by GoLearningBus. WAGmob, 2015.
20. ČÁPKA, D. 1. díl - Popis MVC architektury. In: *itnetwork.cz [*online]. 2014. Dostupné také z: http:// www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php-popis-architektury
21. BERNARD, B. Zdroják.cz. [Úvod do architektury MVC] In: Úvod do architektury MVC [online]. 7. 5. 2009 [cit. 2015-01-16]. Dostupné z: http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/
22. GRUDL, D. Seznámení s Nette Frameworkem. In: Nette [online]. Dostupné také z: http://doc.nette.org/cs/ 2.2/getting-started
23. GRUDL, D. Začínáme. In: Nette [online]. Dostupné také z: http://doc.nette.org/cs/2.2/quickstart/getting-started
24. ČÁPKA, D. 1. díl - Úvod do Nette frameworku pro PHP. In: inetwork.cz [online]. 2014. Dostupné také z: http://www.itnetwork.cz/uvod-do-php-frameworku-nette
25. JetBrains. PhpStorm [online]. 2015 [cit. 2015-leden-15]. Dostupné z: https://www.jetbrains.com/ phpstorm/

26. Wikipedia. Google Maps [online]. [cit. 2015-leden-22]. Dostupné z: http://en.wikipedia.org/wiki/Google_Maps
27. Calculate distance, bearing and more between Latitude/Longitude points. In: Movable Type Scripts [online]. Dostupné také z: http://www.movable-type.co.uk/scripts/latlong.html
28. KOVYRIN, O. Geo Distance Search with MySQL. In: Scribd.com [online]. 2006. Dostupné také z: http://www.scribd.com/doc/2569355/Geo-Distance-Search-with-MySQL
29. BÖHMER, M. Návrhové vzory v PHP. 2012. ISBN 978-80-251-3338-5.
30. LEON, G. gmaps.js [online]. 2012. Dostupné také z: https://hpneo.github.io/gmaps/

7 SEZNAM OBRÁZKŮ

8 SEZNAM POUŽITÝCH ZKRATEK

AJAX = Asynchronous JavaScript and XML

API = Application Programming Interface

CSS = Cascading Style Sheets

IS = Informační Systém

LAMP = Linux Apache MySQL PHP

MVC = Model View Controller

SQL = Structured Query Language

OS = Operační Systém

PHP = Hypertext Preprocessor

SEO = Search Engine Optimalization

SQL = Structured Query Language

UI = User Interface

9 SEZNAM PŘÍLOH

Příloha CD – s kompletními zdrojovými kódy a SQL dump databáze.