UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Rezervační systém pro restaurace

Karel Cerman

Bakalářská práce
2014

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Karel Cerman**
Osobní číslo:	**I11023**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Rezervační systém pro restaurace**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je vytvořit aplikaci, která umožní provádět přes internet objednávku jídla v restauracích. Při objednání se uživateli zobrazí předpokládaná doba příjezdu rozvozu. Na straně poskytovatele bude aplikace schopna mapového zobrazení požadovaných rozvozů a vhodnou metodou bude zvolena optimální trasa mezi těmito body. Výstupem tedy bude kompletní trasa pro pracovníka rozvozu. Aplikace bude využívat technologie PHP (framework Nette) spolu s databází MySQL.
V teoretické části práce bude provedeno srovnání stávajících systémů stejného zaměření a proběhne jejich srovnání s navrženým systémem. Dále bude provedena rešerše v oblasti možností přístupu k databázi MySQL z prostředí jazyka PHP (O/R mapování atd.).

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

RIORDAN, Rebecca M. Vytváříme relační databázové aplikace. Vyd. 1. Praha: Computer Press, 2000, xiv s., 280 s. ISBN 80-722-6360-9.
KOFLER, Michael. Mistrovství v MySQL 5. Vyd. 1. Překlad Jan Svoboda, Ondřej Baše, Jaroslav Černý. Brno: Computer Press, 2007, 805 s. ISBN 978-80-251-1502-2. www.oracle.com
dev.myslq.com
AJAX a PHP: tvoříme interaktivní webové aplikace profesionálně. Vyd. 1. Brno: Zoner Press, 2006, 320 s. ISBN 80-868-1547-1.
GILMORE, W. Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. Nové, 3. vyd. Překlad Jan Pokorný. Brno: Zoner Press, 2011, 736 s. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.
SVENNERBERG, Gabriel. Beginning Google Maps API 3. New York, NY: Apress, c2010, xvi, 310 p. Expert's voice in Web development. ISBN 978-143-0228-028.
CHOW, Shu-Wai. Programujeme Mashup aplikace pro Web 2.0 v PHP. Vyd. 1. Brno: Computer Press, 2008, 280 s. ISBN 978-80-251-2057-6.
https://developers.google.com/

Vedoucí bakalářské práce: **Ing. Jiří Zechmeister**
Katedra informačních technologií

Datum zadání bakalářské práce: **20. prosince 2013**
Termín odevzdání bakalářské práce: **9. května 2014**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2014

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 9. 5. 2014 Karel Cerman

Poděkování

Rád bych poděkoval Ing. Jiřímu Zechmeisterovi za poskytnutí cenných rad, věcných připomínek a vstřícný přístup při vedení mé bakalářské práce. Dále bych chtěl poděkovat rodičům a kamarádům za trpělivost a podporu během mého studia.

Anotace

Cílem práce je vytvořit aplikaci, která umožní provádět objednávku jídla v restauracích. Při objednání se uživateli zobrazí předpokládaná doba příjezdu rozvozu. Na straně poskytovatele bude aplikace schopna mapového zobrazení požadovaných rozvozů. Aplikace bude využívat technologie PHP (framework Nette) spolu s databází MySQL.

V teoretické části bude provedeno srovnání stávajících systémů stejného zaměření a proběhne jejich srovnání s navrženým systémem. Dále bude provedena rešerše v oblasti možnosti přístupu k databázi MySQL z prostředí jazyka PHP (O/R mapování atd.).

Klíčová slova

PHP, MySQL databáze, Nette Framework, problém obchodního cestujícího, restaurace, zobrazení trasy

Title

Restaurant´s reservation system.

Annotation

The aim of this bachelor thesis is to create an application that allows ordering meals in restaurants. When ordering, the user shall see the estimated delivery time. The application will enable the provider on the other side to display a map route of the required deliveries. The application will use PHP (Nette framework) technology together with MySQL database.

The theoretical part compares the existing systems with the designed system covering the same focus area. Further research will be conducted concerning the ability to access a MySQL database from PHP (O/R mapping, etc.).

Keywords

PHP, MySQL database, Nette Framework, The travelling salesman problem, restaurant, route display

Obsah

Seznam zkratek

API	Application Programming Interface
CSRF	Cross-site request forgery
CSS	Cascading Style Sheets
DB	Databáze
DOM	Document Object Model
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MySQL	My Structured Query Language
NDBT	Nette Database Table
ORM	Objektově relační mapování
PDO	PHP Data Object
PHP	Hypertext Preprocessor
RDBMS	Relational Database Management System
SQL	Structured Query Language
UML	Unified Modeling Language
URL	Uniform Resource Locator
XHTML	eXtensible HyperText Markup Language
XSS	Cross Side Scripting

Seznam obrázků

Seznam tabulek

Úvod

Cílem práce je vytvořit aplikaci rezervačního systému pro restaurace. Rezervační systém přináší výrazné zjednodušení a vyšší efektivitu práce oproti správě dat v klasické tištěné formě. Je mnohem rychlejší, pohodlnější a odolnější proti chybám. Navíc umožňuje evidovat všechny důležité údaje pro objednávky restauračního zařízení.

V teoretické části práce je popsáno obecné hledisko rezervačního systému a některá jeho specifika. Objasňuje, co má systém umožňovat a jaké funkcionality nabízet, případně konkretizovat problémy, které s tím souvisejí. Dále jsou představeny vybrané existující systémy dostupné na trhu. Jsou vyznačeny jejich klady i zápory. Teoretický rozbor pokračuje i v dalších kapitolách, kde popisuje metody přístupu k databázi MySQL z jazyka PHP. Charakterizuje implicitní způsob přístupu z frameworku Nette, nadstavbovou knihovnu Dibi a návrhové vzory pro metodu objektově relačního mapování. V poslední kapitole teoretické části jsou uváděny ORM frameworky využívající návrhových vzorů.

Praktická část je rozdělena do šesti kapitol. První kapitola se zabývá použitými technologiemi, které byly v aplikaci využity. Jednotlivým technologiím jsou věnovány samostatné podkapitoly. Druhá kapitola praktické části popisuje architekturu databázového návrhu v grafickém znázornění včetně vzájemných závislostí, splňující třetí normální formu. Grafický model je v jednotlivých částech rozpracován do podrobného textu. Obsahem třetí kapitoly je adresářová struktura, zobrazující strukturu celého projektu. Jsou popsány jednotlivé soubory a složky. Je charakterizována jejich funkcionalita a dostupnost přes http protokol. Čtvrtá kapitola je rozborem datového základu celé aplikace. Každému modulu je věnována samostatná podkapitola. Podrobně jsou definovány funkce modulu a datové položky, se kterými pracuje. Předposlední pátá kapitola představuje uživatele, jejich role a specifikuje vzájemnou návaznost na činnosti uživatelů v systému. Definuje jednotlivé vlastnosti a oprávnění pro každou roli samostatně. V poslední kapitole je znázorněna statická struktura systému pomocí UML a UseCase diagramů. Grafický vzhled vyvinuté aplikace je zobrazen v příloze.

1 Rezervační systém pro restaurace

Rezervační systém je druh informačního systému, jehož primárním účelem je přesně evidovat rezervace a dostupnost libovolných komodit v reálném čase. Komoditou může být výrobek nebo služba například jídlo, vstupenka, jízdenka, auto, ubytování, atd. Rezervační systém poskytuje informace o vytížení komodity a zabraňuje přečerpání kapacit. Hlavním nástrojem je rezervační formulář. Rezervační systémy historicky vznikaly v letecké a železniční dopravě a následně našly uplatnění v ubytovacích, restauračních a dalších službách. (2)

Pro zajímavost první rezervační systém vznikl v Americe ve společnosti American Airlines v roce 1946. První rezervační systém se vzdáleným přístupem při použití terminálu na stroji Manchester Mark 1 vynalezli vědci z Univerzity Toronto v roce 1953. Bohužel, tento systém navzdory úspěchu skončil nezdarem kvůli poruchám terminálu koncem téhož roku. Nicméně první plně funkční rezervační systém, který vydržel nasazení v komerční sféře, vznikl v roce 1962 a jmenoval se ReserVec. (3)

V rezervačním systému restaurace by se nechalo nalézt nepřeberné množství vlastností a funkcionalit. Po konzultacích v komerční sféře jsem se rozhodl, že by v systému neměly chybět tyto funkcionality:

- Správa objednávek – umožní do systému vložit novou objednávku, její editaci a zrušení. Evidenci objednávek podle jejich stavu.
- Správa položek – povolí vytvoření, editaci a mazání položek, které chce společnost nabízet. Navíc by tyto položky měly být veřejnosti dostupné.
- Výstup pro rozvážku zboží – optimalizuje pro kurýra společnosti trasu rozvážky. Vhodné pro minimalizaci nákladů na rozvoz.
- Uživatelské role a oprávnění – omezuje přístup k různým funkcionalitám pro různé osoby. Vedoucí musí mít přístup k více funkcím než zákazník.
- Model klient-server – potřeba používat aplikaci více uživateli současně na různých zařízeních.
- Bezpečnost – zabránění, aby se nepovolaná osoba dostala k informacím, ke kterým by neměla mít přístup. Ochrana heslem pro uživatelský účet. Obecně je toto požadavek na každou moderní aplikaci.

1.1 Dostupná řešení na trhu

Na trhu dnes existuje mnoho společností zabývajících se rezervačními systémy restaurací. Tyto společnosti se většinou zaměřují na modulární systémy, jako jsou online rezervace stolů, pokladní systém a e-shop. Další společnosti se zaobírají pouze katalogizací restaurací, které mají vlastní rozvoz. Restaurace v těchto katalozích poskytují rozvoz na vlastní náklady a zpravidla platí paušální poplatek za umístění na webové prezentace katalogu. Pokud nevyhovuje ani jedno z nabízených řešení, musíme se obrátit na vývojáře webových aplikací, kteří vytvoří aplikaci na míru.

V následujících podkapitolách je uvedeno několik společností a jejich produkty budou popsány. Informace o produktech jsou získané z oficiálních stránek výrobců. Je možné, že informace mohou být zkreslené a nepřesné.

1.1.1 MATCOMP s. r. o.

Společnost sídlící v Třebíči, která vytváří pouze SW řešení. Nabízí široké spektrum služeb. Jsou to například rezervační a informační systémy pro restaurace, cestovní a realitní kanceláře. Jejich restaurační systém obsahuje tyto moduly:

- Redakční systém – správa webových stránek pomocí uživatelsky přívětivého rozhraní. Stránky tak dokáže spravovat i naprostý laik bez znalosti programování. Můžete upravovat menu, články, vkládat obrázky, tabulky, odkazy a ankety.
- Správa stolů – interaktivní plánek restaurace, kde lze snadno přidávat, odebírat, přesouvat, měnit velikost a počet míst k sezení.
- Online rezervace – zákazníci mají k dispozici registrační, přihlašovací a rezervační formulář. Číslo stolu pak přidělí pověřený zaměstnanec.
- Správa databáze klientů – tato data lze využívat k provozování věrnostního či slevového systému.
- Zaměstnanecké účty s různou mírou oprávnění. (4)

1.1.2 UCS-CZ

Pražská společnost nabízející komplexní, tedy HW i SW řešení na míru pro větší restaurační společnosti. Balíček řešení R-keeper obsahuje:

- Jádro R-keeper ® - softwarový nástroj pro organizaci a kontrolu prodeje. Jeho předností je vysoká bezpečnost a variabilita použití společně se snadným ovládáním. Systém R-keeper ® je modulární a umožňuje pokrytí širokého spektra požadavků zákazníků od řešení provozu s jednou pokladnou přes náročnější provozy s větším množstvím pokladen v rámci počítačové sítě až po centralizované řízení prodeje v desítkách provozoven v různých lokalitách u českých i mezinárodních obchodních řetězců.
- VIP karty - věrnostní klientský systém s možností slev, bonusů, kreditu a debetního čerpání umožňující v případě potřeby vzájemné kombinace u individuálního klienta či skupiny.

- Store House - skladové hospodářství. Sledování stavu zásob, receptur, pohybu zboží, inventur, reporty ziskovosti, optimalizace toku zboží a peněz.
- Video surveilance - systém se záznamem videa propojený s pokladními operacemi v režimu on-line i off-line. Poskytuje maximální dostupnou kontrolu prodeje. Každá pokladní operace je zaznamenána a je spojena s obrazovým záznamem.
- Shelter - rezervační systém pro hotelové provozy.
- Pizza delivery - systém pro příjem telefonických a e-mailových objednávek k rozvozu. Transakční historie klienta.
- Monoblok Glaive – dotyková pokladna (5)

1.1.3 Wapis Group s. r. o.

Ostravská společnost poskytuje kompletní služby v oblasti webdesignu, tvorby e-shopů a SEO optimalizace. V rámci tvorby webových prezentací nabízí svým klientům zajímavé řešení pro realitní portály, restaurace, hotely i personální agentury. Dále se zabývá vytvářením intranetových systémů na míru, internetovou reklamou a komplexní realizací corporate identity firem. Konkrétní systém pro pizzerie obsahuje tyto moduly:

- Redakční systém – společnost nepublikuje vlastnosti svého CMS systému.
- Pizza – je modul sloužící pro přidávání jednotlivých druhů pizz. U každé pizzy je možné vyplnit její název, základní údaje jako je váha, cena, kód, fotografie, popis použitých surovin i přiřadit ji do vybrané kategorie. K dané pizze je možné také přidat její vlastnosti (např. ostrá, z BIO surovin).
- Atributy – slouží pro definování zvláštních vlastností pizzy a jejich zvýraznění, např. ostrá, doporučujeme, novinka apod.
- Ingredience – slouží pro sestavení seznamu veškerých ingrediencí, které jsou na pizzách použity. U konkrétní pizzy se zvolí použité ingredience. Tento modul umožňuje zákazníkům filtrovat pizzy dle zvolených přísad.
- Objednávka pizzy – modul pro vytvoření objednávky. Po odeslání objednávky je zřejmé, kdo a jakou pizzu si objednal, i kam se má doručit. (6)

1.1.4 Katalogy restaurací

Mezi nejznámější webové katalogy restaurací, které poskytují rozvoz svých jídel, patří:

- http://damejidlo.cz/ – webová stránka nabízí po zadání místa rozvozu dosažitelný typ jídla a možnost jeho objednání. Novinkou, kterou server zavádí je objednání každodenního nákupu a jeho doručení. Jistou nevýhodou je omezená časová dosažitelnost nabízených služeb – nelze objednávat non stop.
- http://www.jidloted.cz/ – server je pro zákazníka velmi přehledný a snadno ovladatelný. V nabídce jsou položky velmi dobře a přehledně řazeny. Důležitá je pro zákazníka i informace o okamžité ceně a časová dostupnost nabízených služeb. Nevýhodou je orientace pouze na zákazníky v Praze a jejím blízkém okolí.
- http://www.rozvozjidla.cz/ – portál nabízí službu pouze v Praze a velmi blízkém okolí. Jsou uvedeny obchodní podmínky pro nabízené služby. Sortiment je řazen podle národních kuchyní a také podle oblíbenosti u zákazníků. Je nutné se pro vytvoření objednávky zaregistrovat.

Společným znakem pro uvedené katalogy je snadná ovladatelnost, přehledné řazení nabízených služeb a produktů. Pro zákazníka je nezanedbatelná informace o okamžité výši ceny objednávky.

2 Metody přístupu k databázi MySQL přes PHP

V PHP existují pro práci s různými databázovými systémy specifická aplikační rozhraní, zpravidla závislá na výrobci databázového systému. Aby kód, resp. aplikace mohla být přenositelná na jinou databázi, bylo zapotřebí vytvořit jakousi abstraktní vrstvu společnou pro všechny databázové systémy. Jedním přístupem bylo používání ODBC[1], které funguje jako mezičlánek mezi aplikací a databází a zajišťuje společné aplikační rozhraní pro databázové systémy. ODBC však vyžaduje instalaci ovladače a neumožňuje vždy využít pokročilé funkce konkrétního databázového systému. Proto začaly v PHP vznikat knihovny, které nabízejí společné aplikační rozhraní pro libovolný databázový systém. Takovou knihovnou je například PHP Data Objects (PDO), která je standardní součástí PHP od verze 5. 1. PDO nabízí společné aplikační rozhraní a zároveň dovoluje využít všechny funkce databázového jazyka. To usnadňuje přenositelnost kódu, ale neřeší nezávislost na použitém databázovém systému. Databázové systémy používají různé SQL dialekty. Proto byly vyvinuty knihovny, které se snaží o maximální možnou nezávislost na databázovém systému. Hovoříme o úplné databázové abstrakci. (7)

2.1 Nette Database

Tato třída je součástí frameworku Nette a používá se pro připojení k databázi. Dle databázového serveru si vytvoří vnitřní ovladač. Podporované systémy jsou MySQL, PostgreSQL, Sqlite 3 a Oracle. (8)

Do verze Nette 2.0 je základem obálka nad PDO nazvaná Nette\Database\Connection. Jediným problémem této verze byl bug v PDO způsobující memory leak[2]. Z tohoto důvodu byla třída Connection přepsána a ve verzi 2.1 již není potomkem PDO. Ve verzi 2.2 je třída Connection přejmenována na Context. Rozdíly oproti Dibi(viz 2.2) jsou menší možnosti skládání v klauzuli WHERE a nevyužívá se zde statické obálky, kterou nahrazuje DI Container.

Ve třídě Context se využívá přístup k objektu databáze kde parametry funkce query je tradiční SQL dotaz, viz níže.

```
$this->db->query('SELECT * from ridici ORDER BY ridici_id');
```

Nette Database Table.

Další možností práce s databází ve frameworku Nette je využití NDBT, která implementuje knihovnu NotORM, jejíž autorem je Jakub Vrána. Zapojením této knihovny do Nette vyvolalo o NDBT obrovský zájem a dnes je využívanější než třída Connection. Zajímavostí je, že tato knihovna procházela mnohem rychlejším vývojovým cyklem než zbytek frameworku. Níže bude knihovna NotORM popsána.

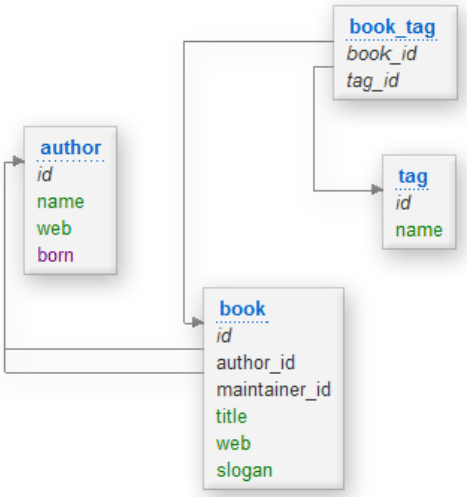
[1] Standardizované softwarové API pro přístup k databázovým systémům
[2] Chyba programu, která zabraňuje uvolnění nevyužité paměti. Může vést až k vyčerpání dostupné paměti.

NotORM se používá při práci s propojenými tabulkami v databázi. Psaní SQL dotazu spojujícího několik tabulek nemusí být vždy efektivní, protože se znovu přenáší už jednou přenesená data. Někdy je proto výhodnější, položit více jednoduchých dotazů a propojit až jejich výsledky. Elegantní a efektivní získávání dat se dosahuje pokládáním minimálního počtu dotazů. Přenesená jsou pouze potřebná data z databáze. Řešení má dva způsoby:

- Získání souvisejících záznamů se provádí najednou pro všechny řádky výsledku jedním dotazem.
- Přenášejí se jen ty sloupce, které se nakonec skutečně použijí

Pro názvy sloupců databáze se používají nejrozšířenější konvence – primární klíč je *id*, cizí klíč *table_id*, kde *table* je název odkazované tabulky.

NotORM se nestará o ukládání a validaci dat. Validaci dat je lepší řešit na úrovni databáze pomocí integritních omezení (unikátní a cizí klíče, datové typy a povinné sloupce). Jinak hrozí, že se do databáze dostanou nekonzistentní data. Ohlašování chyb je vhodné řešit v místě jejich vzniku, o to se starají v Nette formuláře. Níže je zobrazena ukázka práce s NotORM. (9)



Obrázek 1 - Příklad SQL databáze

```
// $database je objekt Nette\Database\Table
foreach ($database->table('book')->order('title')->limit(5) as $book)
{
    echo $book->title, ' (', $book->author->name, ')';
    // ekvivalentní s $book['title'], ' (', $book['author']['name'], ')';

    foreach ($book->related('book_tag') as $book_tag) {
        echo $book_tag->tag->name . ', ';
    }
}
```

2.2 Dibi

Dibi je knihovna pro skriptovací jazyk PHP, která usnadňuje a zjednodušuje práci s databází. Především pak zjednodušení zadávání SQL příkazů a ulehčení běžně používaných rutin – např. získání výsledku dotazu jako dvourozměrné pole, import/export SQL souboru atd. Mezi další přednosti patří nezávislost databázového systému, Dibi podporuje MySQL, PostgreSQL, SQLite, MS SQL a další. Programátor může díky této nezávislosti vytvořit svoji aplikaci pod jedním typem databáze a v produkčním prostředí je možné bez větších problémů využít zcela jiný databázový systém. Autorem je český programátor David Grudl. Níže budou popsány určité speciality této knihovny.

Připojení k databázi

Připojení k databázi je reprezentováno objektem DibiConnection. Ten komunikuje s databází přes ovladač (třída implementující IDibiDriver). Při vytváření tohoto objektu zvolíme, jaký databázový ovladač použít.

```
$options = array(
    'driver'   => 'mysql',
    'host'     => 'localhost',
    'username' => 'root',
    'password' => '***',
   'database' => 'table',
);

// v případě chyby vyhodí DibiException
$connection = new DibiConnection($options);
$connection->query('TRUNCATE `table`');
```

Obrázek 2 - Připojení k DB

Na obrázku 3 je zobrazen tradiční způsob připojení k databázi. Nicméně v Dibi existuje statický registr `dibi`. Ten má za úkol udržovat v globálně dostupném uložišti objekt spojení a nad ním volat potřebné funkce.

```
dibi::connect(array(
  'driver'   => 'mysql',
  'host'     => 'localhost',
  'username' => 'root',
  'password' => '***',
  'database' => 'test',
  'charset'  => 'utf8',
));
```

Obrázek 3 - Připojení k DB přes Dibi

Statická třída `dibi` má jednu obrovskou výhodu, je kdykoliv „po ruce“, když ji programátor potřebuje. Nemusíme vytvářet instanci spojení, lze jednoduše napsat `dibi::`.

Modifikátory datových typů

```
dibi::query('SELECT * FROM [table] WHERE [id] = %i', $id);
```

V ukázce je vidět, že název tabulky *table* je uzavřen do hranatých závorek. Dibi automaticky upraví syntaxi SQL dotazu pro používaný databázový systém. Pro MySQL vznikne z `[table]` text `´table´` a např pro ODBC zůstane zachováno `[table]`. Modifikátor `%i` je nahrazen proměnnou `$id`, ve které je očekáván datový typ *integer*. Dibi tedy umožňuje používání zástupných znaků pro různé datové typy. V tabulce níže zobrazuje některé zástupné řetězce.

Znak	*Datový typ*
%s	string
%b	boolean
%i %u	integer
%f	float
%sql	SQL – řetězec ponechá beze změny

Tabulka 1 - Modifikátory řetězce

Skládání dotazu

Dibi disponuje také podporou pro postupné skládání SQL dotazu. Stačí použít proměnou typu pole a přidávat do této proměnné segmenty vloženého dotazu.

```
$query[] = 'SELECT * FROM [table]';
if ($where){
    array_push($query, 'WHERE [id]=%d', $where);
}
```

Z takového dotazu nám vznikne například dotaz

```
SELECT *
FROM table
WHERE id=3;
```

Podmíněné SQL dotazy

Podmíněné SQL dotazy se ovládají pomocí tří klíčových slov `%if`, `%else` a `%end`. Závěrečné `%end` je možné vynechat. Podmínky je možné zanořovat do libovolné hloubky.

```
dibi::query('
SELECT *
FROM %if', $cond, '[one_table] %else [second_table]'
);
```

Získávání výsledků

Základní metodou je `dibi::query()`, nad kterou je možné použít klasickou iteraci. V každém průchodu dané iterace je vrácen jeden řádek tabulky. Metoda `dibi::fetchSingle()` umožňuje získání prvního řádku tabulky. Pokud chceme získat data v podobě asociativního pole klíč => hodnota použijeme metodu `dibi::fetchPairs()`. (9)

2.3 OR mapování

Při modelování a vývoji aplikací je snaha, co nejvěrněji zachytit realitu. Objekty reálného světa jsou v aplikaci reprezentovány jako entity. Zatímco je v relační databázi entita reprezentována jako řádek, resp. množina řádků v databázových tabulkách, tak v PHP je entita zpravidla reprezentována jako instance nějaké třídy.

Tato rozdílná reprezentace entit vedla ke vzniku programovací techniky, označované jako ORM. Stará se o konverzi mezi relačním a objektovým modelem, se kterými se pracuje v objektově orientovaném jazyce. Technika se tak snaží vývojáři dát unifikovaný přístup k libovolnému objektu, resp. entitě, se kterou v aplikaci pracuje. Díky tomu je vývojář při práci s daty do jisté míry odstíněn od nutnosti pracovat s SQL dotazy konkrétní relační databáze. Použití ORM usnadňuje zejména provádění běžných databázových operací jako je čtení, zápis, úprava a mazání dat. Zkráceně označované jako CRUD operace. Důležitou funkcí ORM je zajistit persistentní uchování dat. Tedy aby data, která jsou v aplikaci uložena v operační paměti, zůstala nepoškozena po plánovaném či neplánovaném ukončení běhu aplikace.

ORM se dále stará o automatickou konverzi rozdílných datových typů mezi databázovým systémem a programovacím jazykem. Pokročilé techniky ORM také řeší například možnost využití objektové dědičnosti, kterou relační databáze nepodporují.

Nicméně hlavním cílem ORM je synchronizace mezi používanými objekty v aplikaci a jejich reprezentací v databázovém systému tak, aby byla zajištěna persistence dat. Vývojář potřebuje persistentně uchovávat objekty, ale nepotřebuje se starat, jak se tato persistence provede.

Řada implementací ORM se snaží v co největší míře odstínit vývojáře od nutnosti psaní SQL dotazů. Pro selekci objektů z databáze používá raději objektový přístup. Tento postup však zpravidla umožňuje vyhledávat objekty jen podle primárního klíče, což zpravidla nestačí. Některé implementace ORM využívají pro selekci objektů objektový dotazovací jazyk. Jedna z výhod odstínění od práce s SQL může být i určitá nezávislost aplikace na konkrétním databázovém systému, resp. možnost zvolit databázový systém či jiné datové úložiště tak, aby vyhovovalo konkrétním podmínkám a požadavkům. (10)

2.3.1 Návrhové vzory

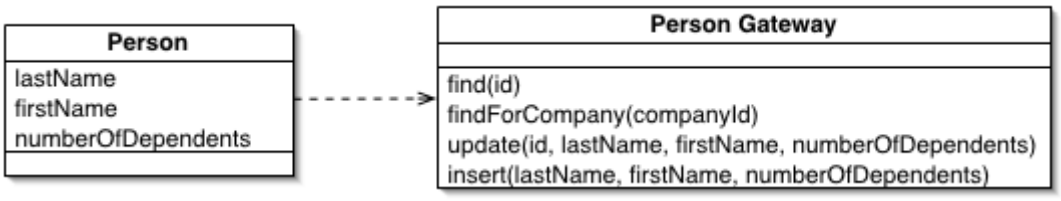
Konverze mezi relační databází a modelovými objekty není bezproblémová. Je spojena s řadou komplikací, souhrnně označovaných jako *object-relational impedance mismatch* (11) . Tyto problémy, resp. jejich možné řešení jsou důvodem pro existenci návrhových vzorů pro ORM.

Cílem a smyslem návrhového vzoru obecně je řešit nějaký často se vyskytující problém při návrhu aplikace. Návrhový vzor se nesnaží přesně popsat, jak problém implementovat, ale jen navrhnout nástin řešení a proto se mohou implementace stejného návrhového vzoru samozřejmě lišit. Zběžná znalost návrhového vzoru nám však pomůže „uchopit“ problém. Po podrobném nastudování konkrétního vzoru již není obtížné problém implementovat, resp. použít knihovnu, která již daný problém řeší.

Softwarový inženýr Martin Fowler ve své publikaci (12) definoval návrhové vzory pro práci s ORM. Některé z těchto vzorů jsou níže popsány.

Table Data Gateway

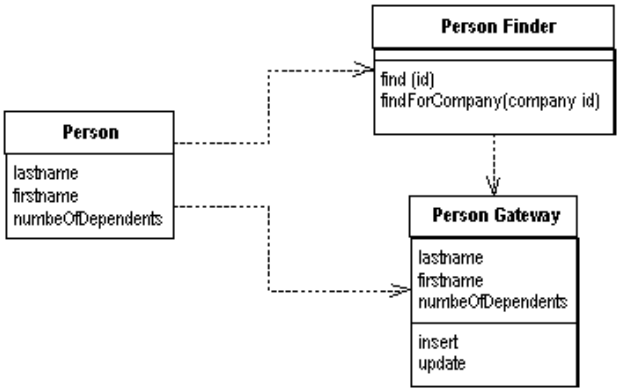
Tento návrhový vzor funguje jako brána, která zapouzdřuje databázové operace prováděné nad jednou databázovou tabulkou. Každá databázová tabulka je v aplikaci reprezentována samostatnou třídou. Tato třída pak obstarává CRUD operace s jednotlivými řádky tabulky. Metody, které v tabulce vyhledávají řádky ať už pomocí primárního klíče nebo jiných kritérií, by měly vždy vracet množinu řádků a to i v případě, že výsledkem bude jen jeden řádek. Odvozenou třídu lze doplnit o metody, které např. zjednoduší vyhledávání nad konkrétní tabulkou podle specifického kritéria.



Obrázek 4 - Table Data Gateway, zdroj (13)

Row Data Gateway

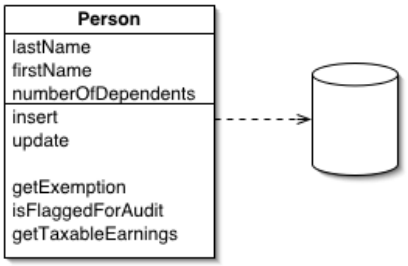
V tomto případě vzor funguje jako brána, která zapouzdřuje CRUD operace nad jedním databázovým řádkem. Každý atribut dané třídy odpovídá danému sloupci v tabulce. K atributům se zpravidla přistupuje přímo, jsou tedy veřejné, bez nutnosti používat *gettery* a *settery*. Instance řádku se vytváří nejčastěji pomocí asociativního pole, které obsahuje hodnoty všech sloupců pro daný řádek. Toto pole je předáno konstruktoru. Instanci řádku tedy nevytváří přímo uživatel, ale vytváří ji oddělená vyhledávací třída. Samotná vyhledávací třída může implementovat vzor Table Data Gateway a je i vhodné tyto vzory spolu kombinovat. Důvodem, proč je konstruktoru předáno pole hodnot a nikoliv primární klíč, je optimalizace databázového výkonu. Třída, která zajišťuje vyhledávání, provede pouze jeden databázový dotaz, který může vrátit více výsledků. Pro každý vrácený řádek vytvoří instanci bez nutnosti volání dalšího databázového dotazu.



Obrázek 5 - Row Data Gateway, zdroj: (13)

Active Record

Jedná se o jeden z nejčastěji používaných návrhových vzorů v oblasti ORM. Tento vzor je podobný jako Row Data Gateway, ale navíc implementuje vlastní business logiku. Lze tedy říci, že se jedná o doménový objekt, který obsahuje navíc CRUD operace. Vytváření instancí zajišťuje opět oddělená vyhledávací třída nebo statická metoda, která vyhledávací třídu deleguje.



Obrázek 6 - Active Record, zdroj (13)

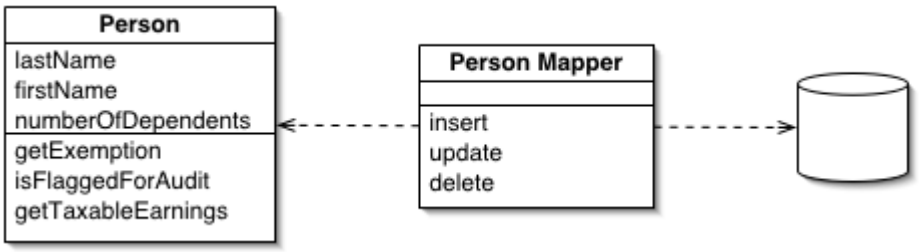
Existence metod, které zajišťují CRUD operace uvnitř doménového objektu, jako u vzoru Active Record, nemusí být vždy ideální. Porušuje tzv. Single Responsibility Principle, kdy každý objekt by měl mít nanejvýš jednu zodpovědnost za určitou činnost. Objekt implementující vzor Active Record je ale zodpovědný jak za business logiku, tak za persistentní uchování dat. S doménovým objektem se často pracuje na různých úrovních aplikace a může být proto matoucí, že obsahuje navíc metody pro uložení do databáze. Doménový objekt má reprezentovat entitu, tedy objekt reálného světa. Entita ale „neví“ o existenci databázového systému a persistentním uchování dat. Proto je z hlediska kvalitního objektového návrhu použití návrhového vzoru Active Record nežádoucí. Paradoxně je ale tento vzor mezi vývojáři značně oblíben právě z tohoto důvodu.

Další nepříjemností vzoru Active Record je, že atributy doménového objektu a sloupce databázové tabulky musí odpovídat 1:1. Podle zvolené vývojové metodiky tedy musí být diagram tříd závislý na datovém modelu, resp. obráceně. Pokud se tedy změní datový model, je potřeba současně upravit doménový objekt a obráceně. Stejný postup platí také pro výše zmíněné vzory Table Data Gateway a Row Data Gateway.

S rostoucí složitostí aplikace je také velmi obtížné zajistit, aby jeden doménový objekt byl v databázi reprezentován jen v jedné tabulce. Všechny tyto problémy se snaží řešit návrhový vzor Data Mapper.

Data Mapper

Podle tohoto vzoru neobsahuje doménový objekt žádné CRUD nebo vyhledávací operace a o vytváření, úpravu a mazání doménových objektů z databáze se stará oddělený (mapovací) objekt. Doménový objekt je tedy zcela nezávislý na databázi. Zatímco mapovací objekt má přístup jak k doménovému objektu, tak k databázovému systému. Výhodou tohoto vzoru je právě nezávislost doménového objektu na datovém modelu, kdy je veškerá zodpovědnost za persistenci přesunuta na mapovací objekt.



Obrázek 7 - Data Mapper, zdroj (13)

Pokud je ale business logika schovaná v doménových objektech, je pro jednodušší případy vhodný vzor *Active Record*. V případě složitějších objektů, nebo pokud potřebujeme nezávislost objektového a datového modelu, je potřeba použít vzor *Data Mapper*.

2.3.2 ORM knihovny v PHP

S vylepšenou podporou objektového přístupu programování v PHP 5 se začaly objevovat různé ORM knihovny. Neexistuje totiž jen jedna správná možnost, jak implementovat ORM. Existuje celá řada rozličných návrhových vzorů a vždy je potřeba zvolit přístup, který lépe odpovídá potřebám tvůrců aplikace. Podle toho různé knihovny implementují různé návrhové vzory: od minimalistického přístupu, kdy knihovna obsahuje jen abstraktní databázovou vrstvu základní CRUD operace, až po komplexní knihovnu, která řeší validaci dat a dědičnost.

Většina ORM knihoven je nezávislá na použitém databázovém systému a obsahuje oddělenou vrstvu, zpravidla založenou na PDO, pro přístup k datovým zdrojům.

Knihovna	*Návrhový vzor*	*Licence*	*Domovská stránka*
CakePHP	Active Record	MIT Licence	http://cakephp.org
Doctrine	Data Mapper	GNU LGPL	http://doctrine-project.org
phpActiveRecord	Active Record	MIT Licence	http://phpactiverecord.org
Propel	Table Data Gateway	MIT Licence	http://propelorm.org
Zend_Db_Table	Table Data Gateway	BSD Licence	http://framework.zend.com

Tabulka 2 - ORM knihovny pro PHP

3 Praktická část rezervačního systému pro restaurace

Požadavky a vlastnosti rezervačního systému pro restaurace jsou zapracovány do návrhu webové aplikace použitelné v praxi. Kompletní zdrojové kódy aplikace a obsah databáze jsou dostupné na CD, které je součástí práce. Zároveň je aplikace dostupná na adrese *http://karelcerman.php5.cz/www/*.

3.1 Použité technologie

3.1.1 PHP

PHP je zkratkou „*Hypertext Preprocessor*“, v češtině je překládán jako hypertextový preprocesor. Původní název ovšem byl „*Personal Home Page*“. Jedná se o skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek a webových aplikací obvykle ve formátu HTML a XHTML. Při použití pro dynamické stránky jsou skripty prováděny na straně serveru, k uživateli je přenášen až výsledek jejich činnosti. Dochází tedy ke skrytí kódu před uživatelem.

Syntaxe jazyka je inspirována několika programovacími jazyky, konkrétně je to Perl, C, Pascal a Java. Výhodou jazyka PHP je nezávislost na platformě. Rozdíly v různých operačních systémech se omezují na několik systémově závislých funkcí a skripty lze většinou mezi operačními systémy přenášet bez jakýchkoli úprav.

PHP je nejrozšířenějším skriptovacím jazykem pro web. Je dostupný prakticky v každé nabídce hostingových společností. Oblíbeným se stal především díky jednoduchosti použití a bohaté zásobě funkcí. V kombinaci s operačním systémem Linux, databázovým systémem (obvykle MySQL nebo PostgreSQL) a webovým serverem Apache je často využíván k tvorbě webových aplikací. Pro tuto kombinaci se vžila zkratka LAMP – tedy spojení Linux, Apache, MySQL a PHP, Perl nebo Python.

PHP podporuje mnoho knihoven pro různé účely - např. zpracování textu, grafiky, práci se soubory, přístup k většině databázových systémů (př. MySQL, Oracle, PostgreSQL, MSSQL). (14)

Aktuální stabilní verzí – ke dni 20. 4. 2014 – je PHP 5. 5. 11 nicméně aplikace byla psána na localhost serveru s verzí 5. 4. 12 a umístěná na hosting (php5.cz) s verzí 5. 5. 10.

3.1.2 JavaScript

JavaScript je další z programovacích jazyků používaných ve webových aplikacích. Podobně jako PHP se většinou umísťuje rovnou do HTML kódu. V případě rozsáhlejšího kódu se ukládá do samostatného souboru. Zpracování je ovšem rozdílné. Spuštění skriptu se realizuje přímo v prohlížeči, tedy na straně klienta, ne na serveru. To může vést k problémům a částečné nefunkčnosti aplikace, pokud je JavaScript na straně klienta nefunkční nebo zakázán. JavaScript se většinou používá pro doplňkové funkce, které nejsou nezbytně nutné pro chod stránky. Patří mezi ně různé grafické efekty, interaktivní tlačítka a podobně. Velká výhoda je asynchronní běh JavaScriptového kódu. Je možné na stránku přidávat další

objekty bez nutnosti aktualizace stránky. Toto se hodí zejména u formulářů, kde se dají podle potřeby dynamicky přidávat nová pole.

Syntaxe je odvozená od jazyků C/C++/Java. Samotný JavaScriptový kód se vloží do HTML pomocí párového tagu <script>. Poté už stačí jen vytvořit funkce, ve kterých se budou vykonávat požadované činnosti. (15)

3.1.3 MySQL

MySQL je velmi populární databázový systém, vytvořený švédskou firmou MySQL AB, v současnosti je vlastněn a vyvíjen společností Oracle Corporation. Svoji popularitu si tento databázový systém získal především díky snadné implementovatelnosti. Lze jej nainstalovat na dnes běžně používaných operačních systémech. Pro MySQL existuje grafická nadstavba pro administraci – phpMyAdmin, jenž je naprogramovaná jako webová aplikace v php. Lze v ní jednoduše pracovat s MySQL – vytvářet/ editovat/ mazat databáze, tabulky, vazby a samozřejmě i manipulovat s daty samotnými.

Syntaxe vychází ze standartu SQL s několika rozšířeními. Od počátku bylo optimalizováno především na rychlost a to i za cenu některých zjednodušení – např. podporuje pouze základní způsoby zálohování a absence pokročilejších funkcionalit jako jsou triggery a pohledy. Nicméně tyto nedostatky jsou v posledních verzích odstraňovány (16).

Aktuální stabilní verzí Community serveru – ke dni 20. 4. 2014 – je MySQL 5. 6. 18 nicméně aplikace byla psána na localhost serveru s verzí 5. 6. 12 a umístěná na hosting (php5.cz) s verzí 5. 5. 35.

3.1.4 Nette Framework

Jedná se o framework pro tvorbu webových aplikací v PHP 5. Hlavní zaměření frameworku je eliminace chyb, jednoduchost, přehlednost a znovu použitelnost zdrojového kódu. Nejdůležitějším aspektem, na který se Nette soustředí, je bezpečnost. Za programátora automaticky řeší nejčastější bezpečností hrozby, jako jsou:

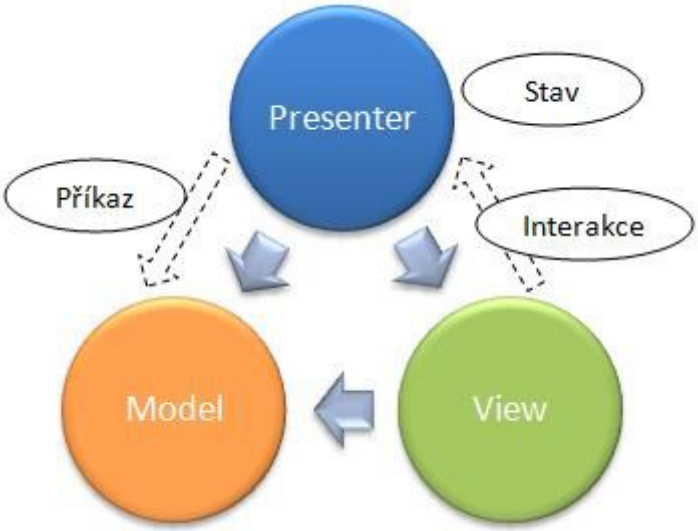
- Cross-site scripting (XSS) – metoda narušení webových stránek zneužívající neošetřených vstupů. Touto metodou je útočník schopen podstrčit vlastní škodlivý kód.
- Cross-site request forgery (CSRF) – vyvolání akce na webu podstrčením odkazu přihlášenému uživateli. Takto lze například pozměnit nebo smazat článek, aniž by si toho uživatel všiml.
- Session hijacking – útočník zcizí platnou session přihlášenému uživateli.

Framework používá architekturu MPV, kde M je modelová vrstva starající se o komunikaci s datovými úložišti, vykonání změn v datech a podobně. V je zobrazovací vrstva. Představuje konkrétní zobrazení požadované stránky včetně dat. Prezentační vrstva (P) se pak stará o výběr konkrétního pohledu zobrazovací vrstvy, zprostředkování dat z modelové vrstvy a jejich poskytnutí vybranému pohledu. Dále udržuje informace

o persistentních parametrech aplikace a především zpracovává akce uživatele. Tyto akce se dají rozdělit na tři typy:

- Změna pohledu,
- změna stavu,
- příkaz modelu.

Jednoduché znázornění tohoto modelu je na následujícím obrázku:



Obrázek 8 - Architektura MVP v Nette Frameworku

Mezi hlavní výhody Nette patří ladící nástroje a práce s formuláři.

Ladící nástroje

PHP dává vývojářům, značnou volnost ve stylu práce. Proto je v něm snadné udělat těžko odhalitelnou chybu. Nette obsahuje tzv laděnku, což je nástroj umožňující rychle odhalit chyby. Podporuje logování, vypisování proměnných a také umí změřit čas načtení stránky. Základem je „Debugger bar“, malá lišta, zobrazující se ve vývojovém režimu stránky. Obsahuje informace o využití paměti, SQL dotazech a o session přihlášeného uživatele. Existuje pro něj řada doplňků.

Obrázek 9 - Debugger bar

Vizualizaci chyb a výjimek zajišťuje „laděnka“ ve které je na první pohled jasné, kde nastala chyba.

Obrázek 10 - Ukázka „laděnky“

Na obrázku je znázorněna chyba v šabloně seznam, kde na řádku 24 je uvedeno špatné makro pro ukončení cyklu foreach.

„Laděnka“ disponuje autodetekcí prostředí. Pokud nastane chyba na produkčním serveru, místo zobrazení dojde k zalogování chyby a uživateli se zobrazí stránka s Error 500.

Formuláře

Vytvoření formuláře je v Nette velice jednoduchou činností. Stačí napsat pouze krátký kód s definicí prvků a následně se automaticky provede validace na straně serveru i na straně klienta. Vyřeší se zabezpečení a prvek se vykreslí. Podpora validace formulářových prvků je rozsáhlá. Každému prvku lze nastavit funkční podmínku, která musí být splněna, např číselný typ, minimální nebo maximální délka textu, vyplnění povinného pole.

```
$form = new Nette\Application\UI\Form;

$form->addText('jmeno', 'Jméno:', 20)
  ->addRule($form::FILLED, 'Vyplňte jméno')
  ->addCondition($form::FILLED);
$form->addText('prijmeni', 'Přijmeni:', 20)
  ->addRule($form::FILLED, 'Vyplňte přijmeni')
  ->addCondition($form::FILLED);
$form->addText('telefon', 'Telefon:')
  ->addRule($form::FILLED, 'Vyplňte Váš Telefon')
  ->addRule($form::MIN_LENGTH, 'Zadavejte pouze čísla bez +420.', 9)
  ->addRule($form::MAX_LENGTH, 'Maximálně %d znaků.', 9);
$form->onSuccess[] = callback($this, 'formNovyRidic');
return $form;
```

Obrázek 11 - Ukázka Nette formuláře

3.1.5 Google Maps API

V některých částech aplikace je používáno mapových podkladů Google Maps. Hlavním důvodem výběru této technologie je dostupnost řešení problému obchodního cestujícího přímo v mapě a to hned několika způsoby. První způsob využívá JavaScript google-maps-tsp-solver a druhá možnost je využít waypoints ve službě Directions Service. Obě možnosti jsou popsány níže. Nadále je v aplikaci využíváno služby Distance Matrix.

Google Distance Matrix API

Na základě zadaných adres startu a cíle vrátí čas jízdy vozidlem a ujetou vzdálenost. Tato hodnota se využívá při předběžném výpočtu doby doručení zakázky.

Google-maps-tsp-solver

Jedná se o skript řešící problém obchodního cestujícího v JavaScriptu, tudíž výpočet probíhá na straně klienta. Postup výpočtu trasy vypadá následovně:

- Do algoritmu vložíme **n** míst, která chceme navštívit.
- Nastavíme mód „Fastest Roundtrip“ neboli nejrychlejší okružní jízda.
- Proběhne výpočet optimální trasy.
- Pomocí metody getGDirections() dostaneme k dispozici objekt s výsledkem
- Vložíme tento objekt do mapy pro vykreslení.

```
// element DIV, do kterého bude vykreslena mapa
<div id="map-canvas" style="width: 800px; height: 400px"></div>
// načtení API
  <script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?key=
  AIzaSyAPTQVD0DekE kaixVZgIsUzElT1vbBkwk&sensor=true"></script>
  <script type="text/javascript" src="http://optimap.net/BpTspSolver.js"></script>

  <script type="text/javascript">
     // klasické načtení Google Maps
    var mapOptions = {mapTypeId: google.maps.MapTypeId.ROADMAP}
    var map = new google.maps.Map(document.getElementById("map-canvas"), mapOptions);
    // vytvoření objektu TspSolver
    tsp = new BpTspSolver(map);
    // nastavení (vyhnout se dálnicím a mód = jízda autem)
    tsp.setAvoidHighways(true);
    tsp.setTravelMode(google.maps.DirectionsTravelMode.DRIVING);

    // vložení průjezdných míst - nejprve pomocí GPS souřadnic (FEI-UPCE)
    // a následně pomocí adresy
    tsp.addWaypoint(new google.maps.LatLng(50.033941, 15.767514));
    tsp.addAddress('Univerzita Pardubice');
    tsp.addAddress('Univerzita Hradec Králové');

    // spuštění výpočtu okružní jízdy
    tsp.solveRoundTrip();

</script>
```

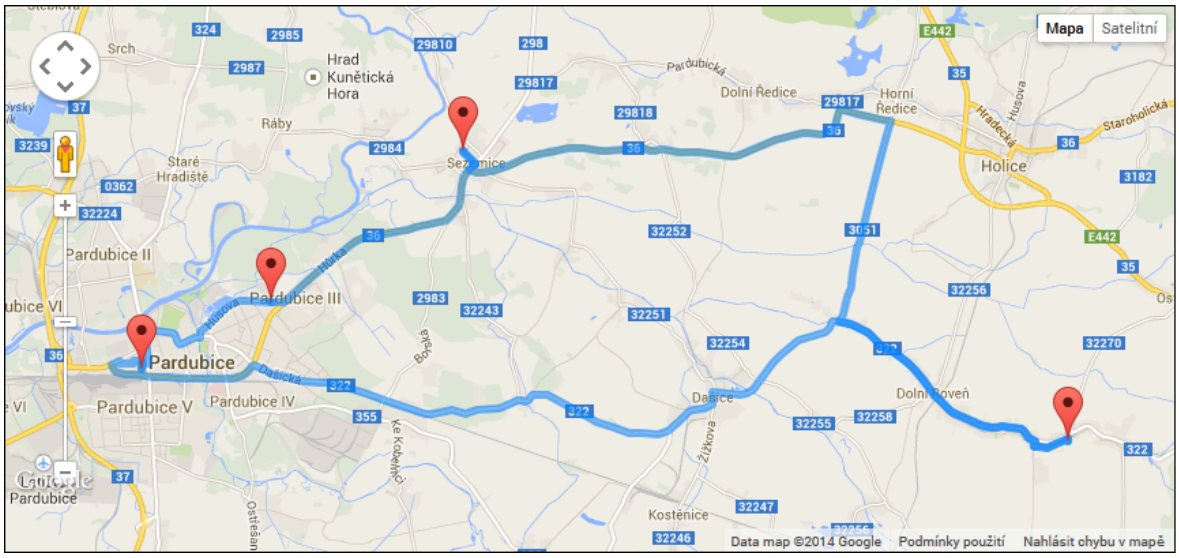
Obrázek 12 - Ukázka kódu Google-maps-tsp-solver

Samotný výpočet probíhá implicitně metodou porovnání každého s každým. Druhým typem řešení je aproximační metoda založená na optimalizaci mravenčí kolonie. Pomocí konstanty *maxTspDynamic* přímo v těle JavaScriptu můžeme nastavit maximální počet bodů, které budou počítány metodou hrubé síly. (17)

Google Directions API

Jedná se o službu, která řeší problém obchodního cestujícího prostřednictvím http požadavku, tedy na straně serveru. Je v ní možné vyhledávat cestu pro několik způsobů dopravy jako jsou například chůze, jízda na kole, jízda autem nebo plavba lodí. Body, které chceme navštívit, je možné zadávat buď adresou, nebo pomocí GPS souřadnic. Ukázkový zdrojový kód je přiložen jako Příloha B. Nejdůležitějším prvkem tohoto řešení v http požadavku je uvést *optimizeWaypoints: true*. Pokud bychom tak neučinili, k výpočtu nejkratší cesty by vůbec nedošlo. (18)

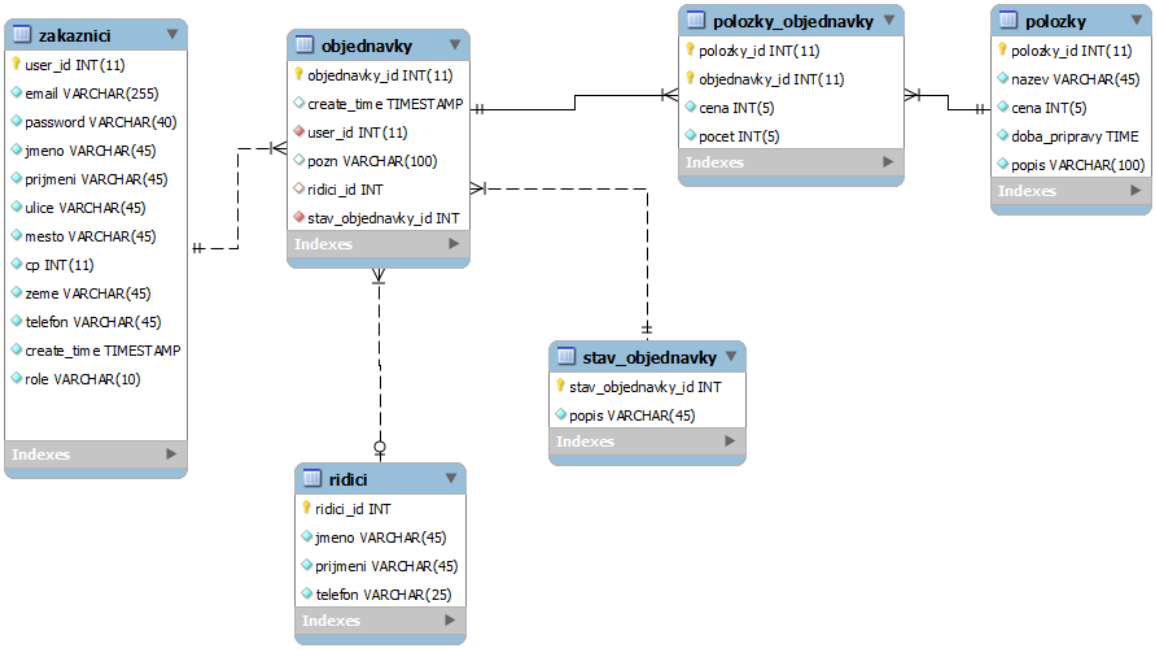
Výsledná trasa za použití obou metod zjišťování nejkratší okružní cesty je stejná, nicméně řešení pomocí Google Directions API je mnohonásobně rychlejší, protože celá problematika obchodního cestujícího se řeší na výpočetních strojích společnosti Google. Výsledku se nám tedy dostane téměř okamžitě.



Obrázek 13 - Ukázka vygenerované trasy

3.2 Databázový Návrh a popis tabulek

Databázový model obsahuje celkem 6 tabulek a je navržen ve formě fyzického modelu a splňuje třetí normální formu (3NF), což umožňuje případné rozšíření funkčnosti do budoucna.



Obrázek 14 - Databázový návrh aplikace

Zakaznici

V tabulce se nacházejí informace o uživatelích aplikace a jejich role.

- user_id INT(11), primární klíč
- email VARCHAR(255), přihlašovací údaj do systému
- password VARCHAR(32), zahešované heslo
- jmeno VARCHAR(45), jméno uživatele
- prijmeni VARCHAR(45), příjmení uživatele
- ulice VARCHAR(45), ulice uživatele
- cp INT(11), číslo popisné uživatele
- mesto VARCHAR(45), město uživatele
- zeme VARCHAR(45), země původu uživatele
- telefon VARCHAR(45), telefon na uživatele
- create_time TIMESTAMP, datum a čas vytvoření účtu, automaticky vyplňováno
- role VARCHAR(10), role uživatele, např uzivatel, ridic, admin.

Ridici

V tabulce jsou uchovávány informace o řidiči společnosti.

- ridici_id INT(11), primární klíč
- jmeno VARCHAR(45), jméno řidiče
- prijmeni VARCHAR(45), příjmení řidiče
- telefon VARCHAR(25), telefon na řidiče

Polozky

V tabulce jsou uchovávány nabízené produkty

- polozky_id INT(11), primární klíč
- nazev VARCHAR(45), název položky
- cena INT(5), cena položky
- doba_pripravy TIME, doba přípravy používaná k předběžnému určení doby příjezdu
- popis VARCHAR(100), popis položky

Stav_objednavky

Tabulka uchovává stavy objednávek, např nová, připravovaná, připravená apod.

- stav_objednavky_id INT(11), primární klíč
- popis VARCHAR(45), popis stavu objednávky

Polozky_objednavky

Spojovací tabulka pro vztah typu M:N. Obsahuje počet a cenu položek v určité objednávce.

- polozky_id INT(11), cizí primární klíč z tabulky Polozky
- objednávky_id INT(11), cizí primární klíč z tabulky Objednavky
- cena INT(5), současná cena položky v době objednání
- pocet INT(5), počet objednaných kusů určité položky

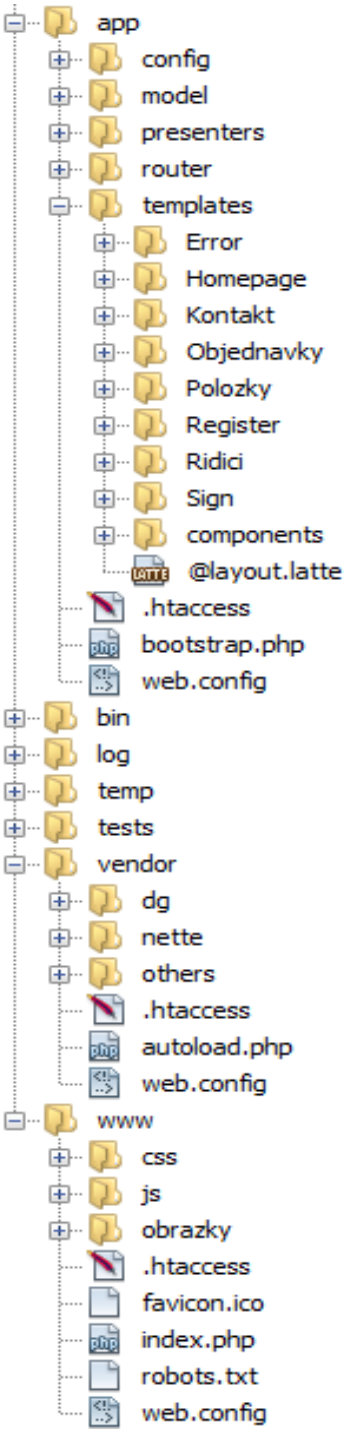
Objednavky

Nejdůležitější tabulka v aplikaci. Uchovává informace o jednotlivých objednávkách.

- objednávky_id INT(11), primární klíč
- user_id INT(11), cizí klíč z tabulky Zakaznici
- stav_objednavky_id INT(11), cizí klíč z tabulky Stav_objednavky
- ridici_id INT(11), cizí klíč z tabulky Ridici
- create_time TIMESTAMP, datum a čas vytvoření objednávky
- pozn VARCHAR(100), poznámka u objednávky

3.3 Adresářová struktura

Rozložení souborů je převážně dáno užitím frameworku Nette. Samo o sobě je velice intuitivní pro programátora, který není do vyvíjeného systému zasvěcený a nikdy nepracoval na projektu s frameworkem Nette.



V hlavním adresáři se nachází sedm adresářů – *app, bin, log, temp, tests, vendor* a *www*. Tyto adresáře nejsou skrze protokol http viditelné. Viditelnosti těchto složek ovlivňují soubory *.htacces* a *web.config*, které jsou konfiguračními soubory pro webserver Apache a Microsoft IIS. Absence těchto souborů může potenciálně vést k bezpečnostnímu riziku.

První adresářem je *app*. Jsou v něm uloženy všechny zdrojové soubory aplikace. Obsahuje následující podadresáře:

- *config* – obsahuje dva soubory *config.neon* a *config.local.neon*. Ty obsahují nastavení aplikace pro localhost a hosting, jako jsou například přihlašovací údaje k databázovému serveru, dobu expirace sessions a registrace využívaných služeb v aplikaci.
- *model* – obsahuje php soubory s modely celé aplikace. Jednotlivé modely budou popisovány později
- *presenters* – obsahuje php soubory s prezentery aplikace.
- *templates* - obsahuje šablony aplikace ve formátu Latte[3]. Nadále obsahuje další podadresáře, které jsou adekvátní k seznamu presenterů. V každém podadresáři nalezneme šablony pro danou akci. Posledním souborem v této složce je *@layout.latte,* což je hlavní šablona, do které se vkládá vyžadovaná šablona.
- soubor *bootstrap.php* – zaváděcí soubor frameworku. Načítá konfiguraci knihoven Nette a je v něm možné konfigurovat automatické načtení všech souborů z DI kontejneru.

Obrázek 15 - Adresářová struktura aplikace

[3] Formát souboru šablony ve frameworku Nette.

Dalším adresářem ve stromové struktuře je *bin.* Tato složka obsahuje soubor *create-user.php*, který slouží k vytvoření a uložení prvního uživatele aplikace do databáze.

Adresář *log* obsahuje upozornění a varování o chybách, které nastaly při práci s aplikací. *Temp* obsahuje soubory pro uchování sessions a cache frameworku. Oba adresáře musí mít nastavená práva na zápis, jinak skončí načtení aplikace takzvanou bílou stránkou.

Adresář *tests* obsahuje soubory pro testování funkčnosti aplikace umístěné na hosting. Pro funkci aplikace tato složka není důležitá, ale v případě testování nových modulů v ostrém provozu, má svůj význam.

Adresář *libs* je srdce celé aplikace. Obsahuje soubory frameworku Nette. Dále obsahuje soubor *autoload.php*, který se stará o správné načtení frameworku.

Posledním adresářem je *www*. Tento adresář je jako jediný dostupný přes http protokol. Obsahuje následující podadresáře/soubory:

- *css* – obsahuje soubory kaskádových stylů,
- *js* – obsahuje skripty v jazyce JavaScript,
- *obrazky* – obsahuje obrázky pro webovou aplikaci,
- soubor *index.php* – načten po vstupu na stránky přes http protokol. Inicializuje *bootstrap.php,*
- soubor *favicon.ico* – ikona stránky.

3.4 Moduly a funkčnost

Model je datový a zejména funkční základ celé aplikace. Je v něm obsažena aplikační logika. Jednotlivé moduly jsou popsány níže. (19)

3.4.1 Objednávky

Modul objednávek je jedním z nejdůležitějších v aplikaci. Umožňuje vytvářet, editovat a odstraňovat jednotlivé objednávky včetně obsažených položek. Pro vytvoření objednávky je nutné, aby obsahovala alespoň jednu položku. Všechny ostatní parametry jsou ukládány automaticky. Jsou to identifikační číslo uživatele, datum a čas vytvoření, celková cena objednávky a její stav. Po vytvoření objednávky se uživateli zobrazí i předpokládaná doba příjezdu kurýra. Tato hodnota je počítána ze vzdálenosti mezi restaurací a adresy zadané uživatelem při registraci, dobou přípravy jídla a skutečností, zda se kurýr momentálně nachází v restauraci, nebo je na rozvozu. Pokud není přítomen, je přičtena konstantní časová hodnota. Nadále je pro řidiče společnosti v tomto modulu generována nejkratší možná okružní cesta mezi restaurací a zákazníky.

3.4.2 Passwords

Modul spolupracující s modulem UserManager (viz 7.4.5), který se stará o vytvoření a zahešování hesla pomocí algoritmu DES a její násady salt, která je zvolená jako uživatelské jméno používané pro přihlášení. Tento modul je implicitně dodávaný s frameworkem. Jeho autorem je David Grudl. (20)

3.4.3 Položky

Modul určený pro správu jídel zobrazujících se v jídelním lístku. Umožňuje přidávat, editovat a odstraňovat položky. Nová položka je definována názvem, popisem, cenou a dobou přípravy ve formátu HH:MM.

3.4.4 Řidiči

Modul určený pro správu řidičů nebo chceme-li kurýrů restaurace. Umožňuje stejně jako modul položek přidávat, editovat a odstraňovat uživatele. Je důležité, aby se jméno a příjmení v tomto modulu shodovalo s uživatelským jménem, které má nastavenou roli „ridic“. Jinak nebude možné pro řidiče generovat okružní cestu. Nový řidič je definován jménem, příjmením a kontaktním telefonem.

3.4.5 UserManager

Modul určený pro správu uživatelů a jejich oprávnění. Umožňuje vytvářet, editovat a odstraňovat uživatele. Každý z nich je identifikován podle emailu a hesla. Pro vytvoření nového uživatele je nutné do registračního formuláře zadat email, heslo, jméno, příjmení, adresu a telefon.

3.5 Uživatelé a jejich role

V celé aplikaci je několik uživatelských rolí, které poskytují množiny oprávnění v návaznosti na činnosti jednotlivých uživatelů. Níže jsou popsána oprávnění, která do jednotlivých rolí patří.

- **Admin** – účet s maximálním oprávněním, může v systému vše. Určený pro prvotní nastavení systému.
- **Vedoucí** – účet určený pro vedoucí pracovníky. Může vytvářet uživatele s libovolnými uživatelskými jmény, tedy neplatí zde kritérium na uživatelské jméno. Má možnost editovat a mazat uživatele, kteří nemají roli admin. Nadále u objednávek může nastavovat všechny stavy objednávky.
- **Obsluha** – jedná se o obsluhu restaurace, která má omezenější oprávnění než vedoucí. Může vytvářet, editovat a mazat řidiče, upravovat položky v jídelním lístku. U objednávek může editovat, přidávat či mazat položky, ale pouze se stavy nová, ve výrobě, připravená a zrušená. Nikdy nemůže vymazat objednávku ze systému.

- **Řidič** – má přístup k seznamu objednávek se stavem připravená. Editací na stav expedovaná do databáze, uloží svoje jméno k objednávce. Z těchto záznamů si může nechat vygenerovat minimální okružní trasu. Z takto vygenerované okružní jízdy může odstraňovat body průjezdu, za předpokladu že by nebyl s výsledkem spokojen.
- **Uživatel** – nejnižší možné oprávnění přihlášeného uživatele. Může vytvářet objednávky pro svoji osobu a editovat informace o svém účtu. Nadále má možnost nahlížet do historie svých vyřízených a zrušených objednávek. Po vytvoření objednávky může sledovat stav výroby objednávky.
- **Nepřihlášen** – může se registrovat a prohlížet webovou prezentaci, včetně jídelního lístku.

Tyto jednotlivé role je možné si vyzkoušet v aplikaci pod přihlašovacími údaji:

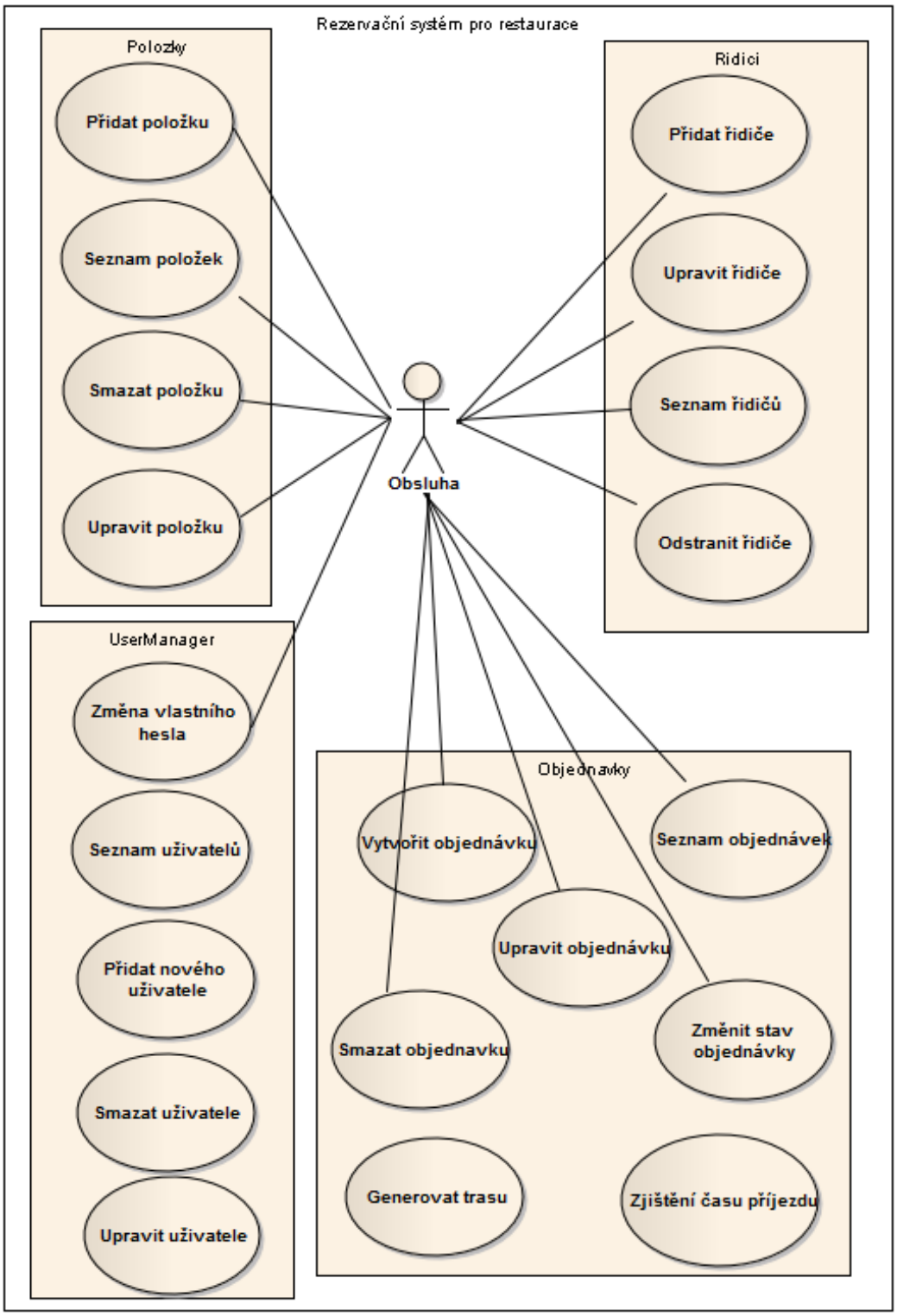
- přihlašovací jméno: *admin*, heslo: *admin*
- přihlašovací jméno: *vedouci*, heslo: *vedouci*
- přihlašovací jméno: *obsluha*, heslo: *obsluha*
- přihlašovací jméno: *ridic*, heslo: *ridic*
- přihlašovací jméno: *uzivatel*, heslo: *uzivatel*

Tabulka 3 - Přehled oprávnění uživatelských rolí

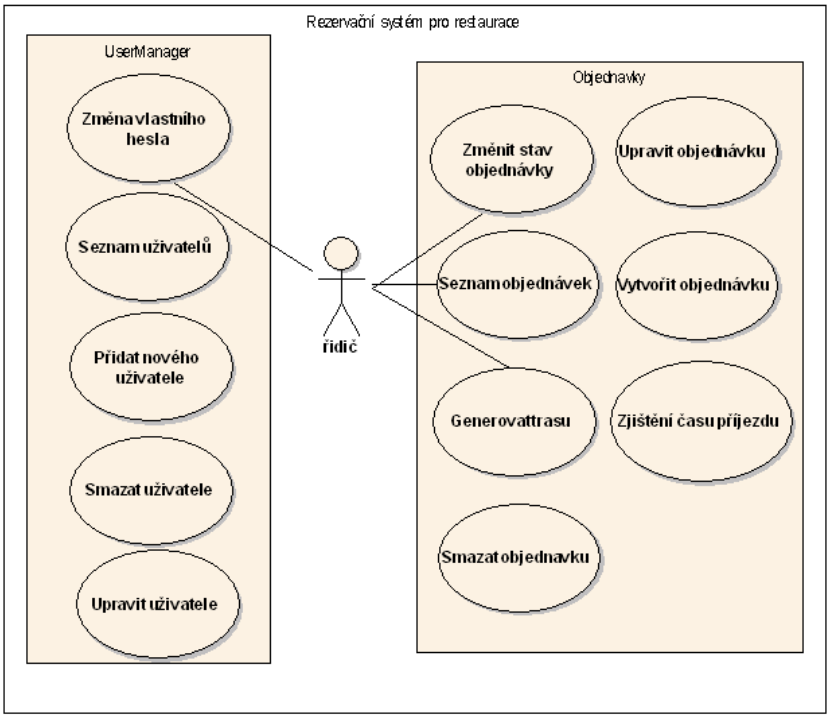
modul / role	*Admin*	*Vedouci*	*obsluha*	*uživatel*
Objednávky	X	X	X	X
Položky	X	X	X	
Řidiči	X	X	X	
UserManager	X	X	X	X

3.5.1 UseCase diagram

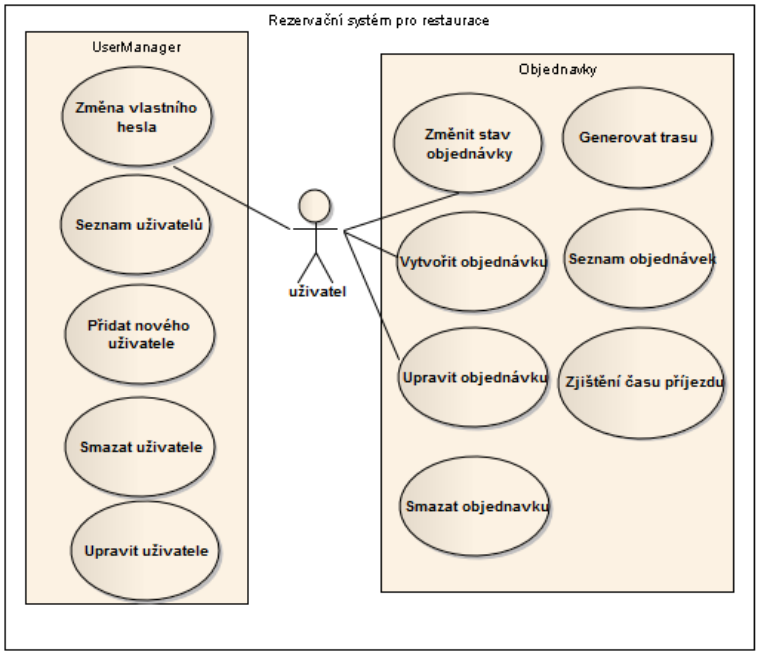
Na obrázcích jsou zobrazeny UseCase diagramy, které zobrazují případy užití pro danou uživatelskou roli. Některé diagramy neobsahují všechny moduly, protože do nich nemá daná role přístup. Z důvodu přehlednosti nejsou zobrazeny UseCase diagramy pro role Admin a Vedoucí, které mají přístup ke všem modulům systému.



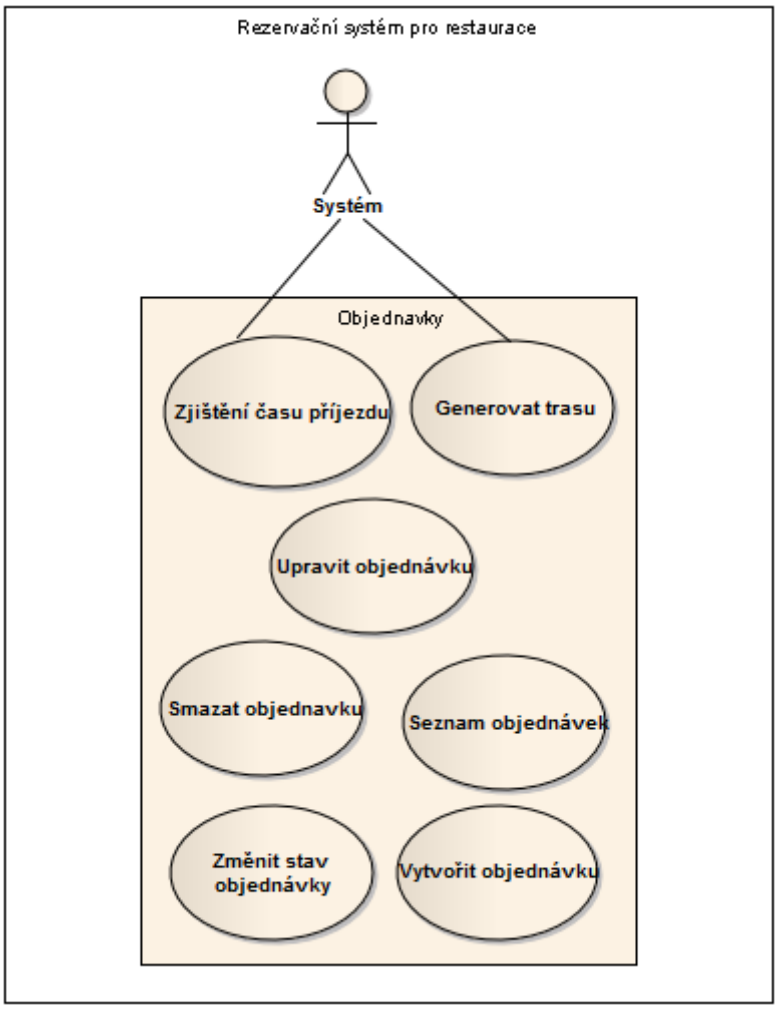
Obrázek 16 - UseCase diagram role Obsluha



Obrázek 17 - UseCase diagram role řidič



Obrázek 18 - UseCase diagram role uživatel

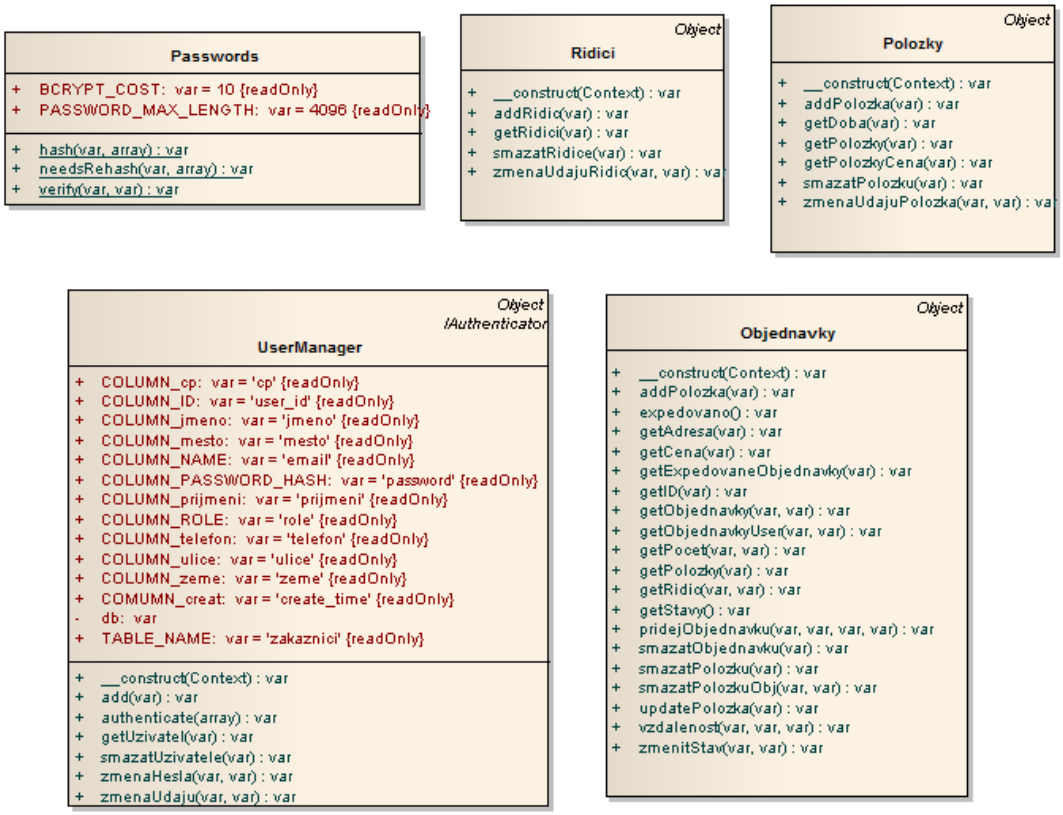


Obrázek 19 - UseCase diagram reprezentující Systém

3.6 UML diagram tříd

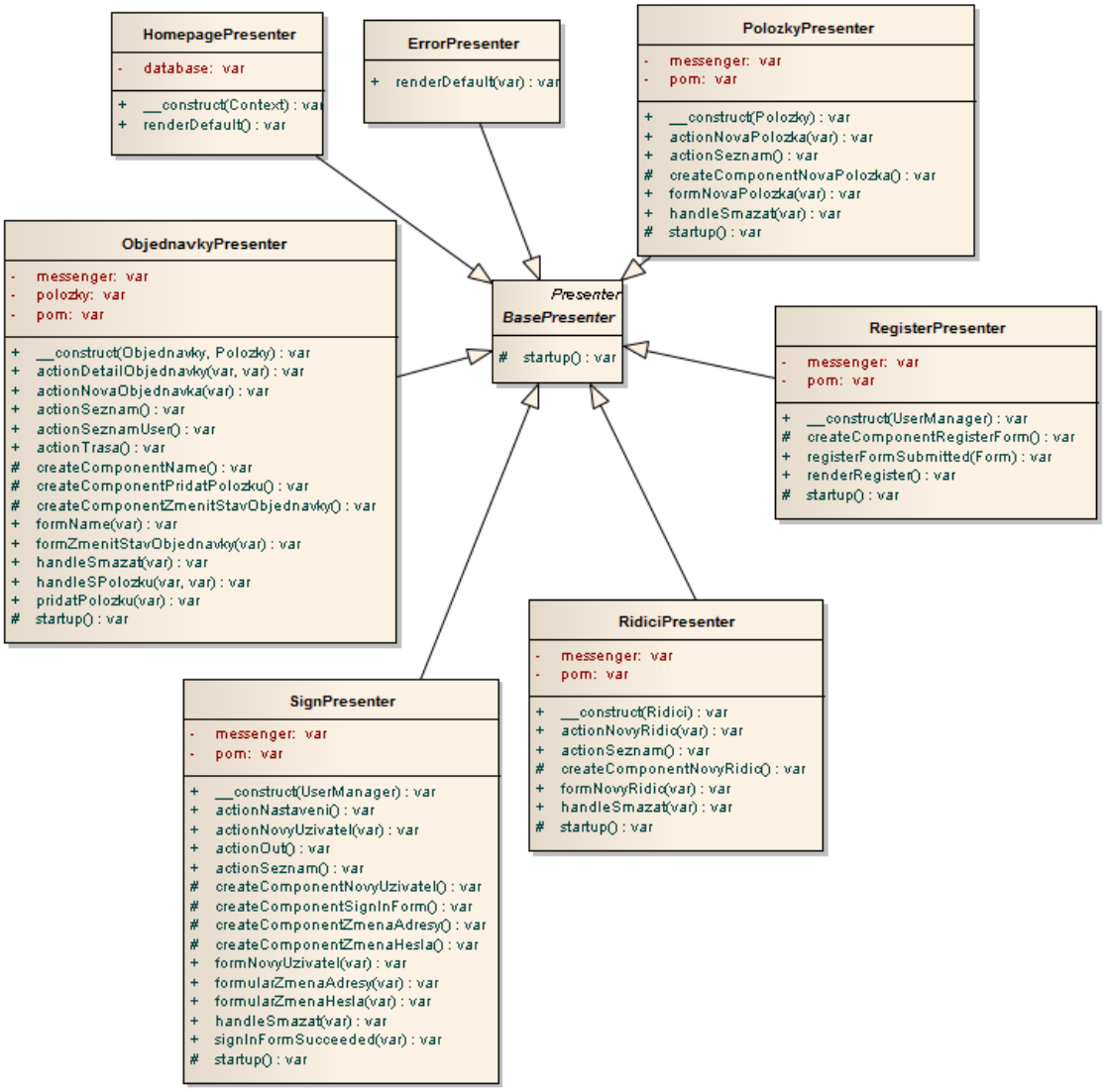
Diagram popisuje statickou strukturu systému, znázorňuje datové struktury a operace u objektů a souvislosti mezi objekty. Znázorňuje datový model systému od konceptuální úrovně až po implementaci.

Jako první je zobrazen modelový diagram tříd aplikace, nikoliv tříd frameworku.



Obrázek 20 - UML diagram modelů

Dále následuje diagram tříd presenterů.



Obrázek 21 - UML diagram presenterů

Závěr

Cílem bakalářské práce bylo vytvořit funkční aplikaci rezervačního systému pro restaurace. Před samotnou implementací bylo však nutné provést analýzu stávajících řešení a nalézt jejich nedostatky a přednosti. Součástí tvorby návrhu aplikace byly i konzultace v komerční sféře. Zpracované údaje vedly ke stanovení rozsahu a potřebným funkcionalitám vzhledem k nárokům kladeným na výsledný návrh rezervačního systému.

Při vytvoření základní struktury práce se zdál vývoj aplikace relativně jednoduchý. Zpracováváním jednotlivých částí práce se postupně objevovaly určité problémy, na které dokumentace frameworku nepomyslela. Tyto komplikace se naštěstí podařilo vyřešit za pomocí ochotné komunity okolo Nette. Jednalo se o problém při vytváření asociativního pole odesílaného z dynamicky generovaného formuláře, kde požadovaná kombinace klíč => hodnota byla vyjadřována jako náhodné číslo => hodnota. Bylo nutno použít dodatečné deklarace parametru ve funkci, která se starala o přijetí dat z formuláře. Tato deklarace je novinkou v poslední stabilní verzi Nette 2.1.2 a zatím nebyla dopsána do oficiální dokumentace frameworku. Při vykreslování „markerů“ v Google mapě docházelo k chybě zapříčiněnou špatnou deklarací pole těchto značek. Při aktualizaci stránky nedocházelo k vymazání obsahu tohoto pole, a proto každé nové vyhodnocení minimální okružní cesty vedlo k hromadění bodů na mapě. Řešením bylo před vytvářením požadavku na vykreslení smazat obsah proměnné typu pole.

Snahou autora bylo vytvořit webovou aplikaci, která by byla uživatelsky přívětivá a jednouchá na ovladatelnost. Lze prohlásit, že aplikace je nejen snadno pro uživatele ovladatelná, ale i do budoucna dle požadavků případného zákazníka upravitelná. Možný příklad rozšíření je zasílání potvrzovacích emailů o přijetí objednávky, napojení aplikace na službu rozesílající informativní SMS o stavu objednávky a aktualizaci času dodání, statistické funkce pro vedení společnosti, hodnocení jídel a věrnostní program pro uživatele. Vzhledem ke stále se rozšiřujícím technickým možnostem GPS systému lze také do aplikace integrovat sledování služebních vozidel pro kontrolu vytíženosti vozidel.

Oproti původnímu záměru se do aplikace nepodařil zapracovat modul telefonických objednávek. Důvodem byla vysoká technická náročnost, která by změnila celou architekturu projektu. Myšlenka zapracování tohoto modulu vznikla téměř před dokončovacími pracemi, a proto nebyla do programu implementována.

Vytvořením funkční aplikace byl cíl práce splněn. Rezervační systém obsahuje pro praxi všechny požadované funkcionality. Nezanedbatelná je i uživatelsky přívětivá grafická úprava, jejíž náhled je uveden v příloze. Momentálně (květen 2014) se vedou konzultace o uvedení systému do praxe. Pochopitelně budou zapracovány i úpravy v aplikaci dle individuálních přání potencionálních zákazníků.

Literatura

1. Rezervační systém. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2013 [cit. 2014-04-25]. Dostupné z: http://cs.wikipedia.org/wiki/Rezervační_systém.

2. Computer reservations system. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2014-04-25]. Dostupné z: http://en.wikipedia.org/wiki/Computer_reservations_system.

3. Rezervační systém pro restaurace. *MATCOMP s.r.o.* [online]. 2013 [cit. 2014-04-25]. Dostupné z: http://www.matesova.cz/webove-aplikace_restaurace.

4. Informační systémy pro hotely a restaurace. *UCS-CZ s.r.o.* [online]. 2014 [cit. 2014-04-25]. Dostupné z: http://www.ucs-cz.cz/produkty/rkeeper.html.

5. Rezervační systém pro restaurace. *Wapis Group s.r.o.* [online]. 2011 [cit. 2014-04-25]. Dostupné z: http://www.wapis.cz/#/sluzby/rezervacni-system-pro-restaurace/.

6. Abstraction Layers. *The PHP Group* [online]. 2014 [cit. 2014-04-25]. Dostupné z: http://www.php.net/manual/en/refs.database.abstract.php.

7. Databáze & ORM. *Nette Foundation* [online]. 2014 [cit. 2014-04-25]. Dostupné z: http://doc.nette.org/cs/2.1/database.

8. VRÁNA, Jakub. NotORM. *PHP triky* [online]. 2010 [cit. 2014-04-25]. Dostupné z: http://php.vrana.cz/notorm.php.

9. GRUDL, David. Dibi - pokrokový databázový layer. *PhpFashion* [online]. 2006 [cit. 2014-04-25]. Dostupné z: http://phpfashion.com/dibi-pokrokovy-databazovy-layer.

10. VRCHOTA, Michal. *Objektově relační mapování v PHP*. Praha, 2010. Bakalářská práce. Vysoká škola ekonomická v Praze, Katedra informačních technologií. Vedoucí práce Ing. Luboš Pavlíček. Dostupné z: https://isis.vse.cz/zp/portal_zp.pl?podrobnosti=52935

11. Mapping Objects to Relational Databases: O/R Mapping In Detail. *Agile Data* [online]. 2013 [cit. 2014-04-25]. Dostupné z: http://www.agiledata.org/essays/mappingObjects.html.

12. FOWLER, Martin. *Patterns of Enterprise Application Architecture. [s.l.] : Addison Wesley, 2002.* ISBN 0-321-12742-0. *S. 560. (anglicky)*

13. FOWLER, Martin. *Catalog of Patterns of Enterprise Application Architecture* [online]. 2003 [cit. 2014-04-25]. Dostupné z: http://martinfowler.com/eaaCatalog/

14. General Information. *The PHP Group* [online]. 2014 [cit. 2014-04-25]. Dostupné z: http://www.php.net/manual/en/faq.general.php.

15. JavaScript Introduction. *W3Schools.com* [online]. 2014 [cit. 2014-04-25]. Dostupné z: http://www.w3schools.com/js/js_intro.asp

16. About MySQL. *Oracle Corporation* [online]. 2014 [cit. 2014-04-25]. Dostupné z: http://www.mysql.com/about/.

17. ENGDAHL, Geir. OptiMap: Fastest Roundtrip Solver. *OptiMap* [online]. 2014? [cit. 2014-04-25]. Dostupné z: http://optimap.net/.

18. The Google Directions API. *Google Developers* [online]. 2014 [cit. 2014-04-25]. Dostupné z: https://developers.google.com/maps/documentation/directions/.

19. MVC aplikace & presentery. *Nette Foundation* [online]. 2014 [cit. 2014-04-25]. Dostupné z: http://doc.nette.org/cs/2.1/presenters.

20. *David Grudl* [online]. 2014 [cit. 2014-04-29]. Dostupné z: http://davidgrudl.com/.

Příloha A – CD s aplikací

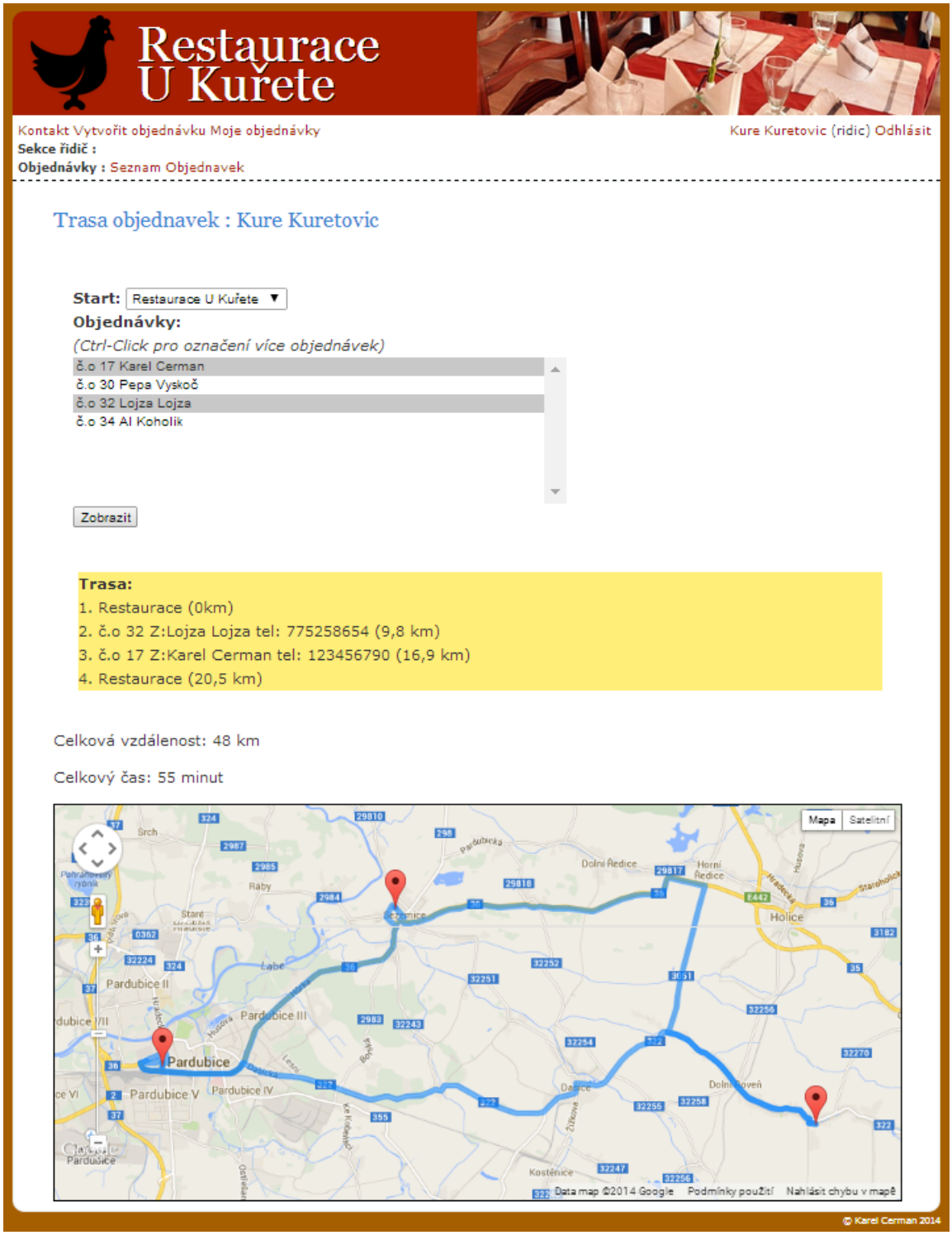
Přílohu tvoří CD se zdrojovými kódy aplikace, export databáze a databázový model pro MySQL Workbench.

Příloha B – Zdrojový kód Directions Service

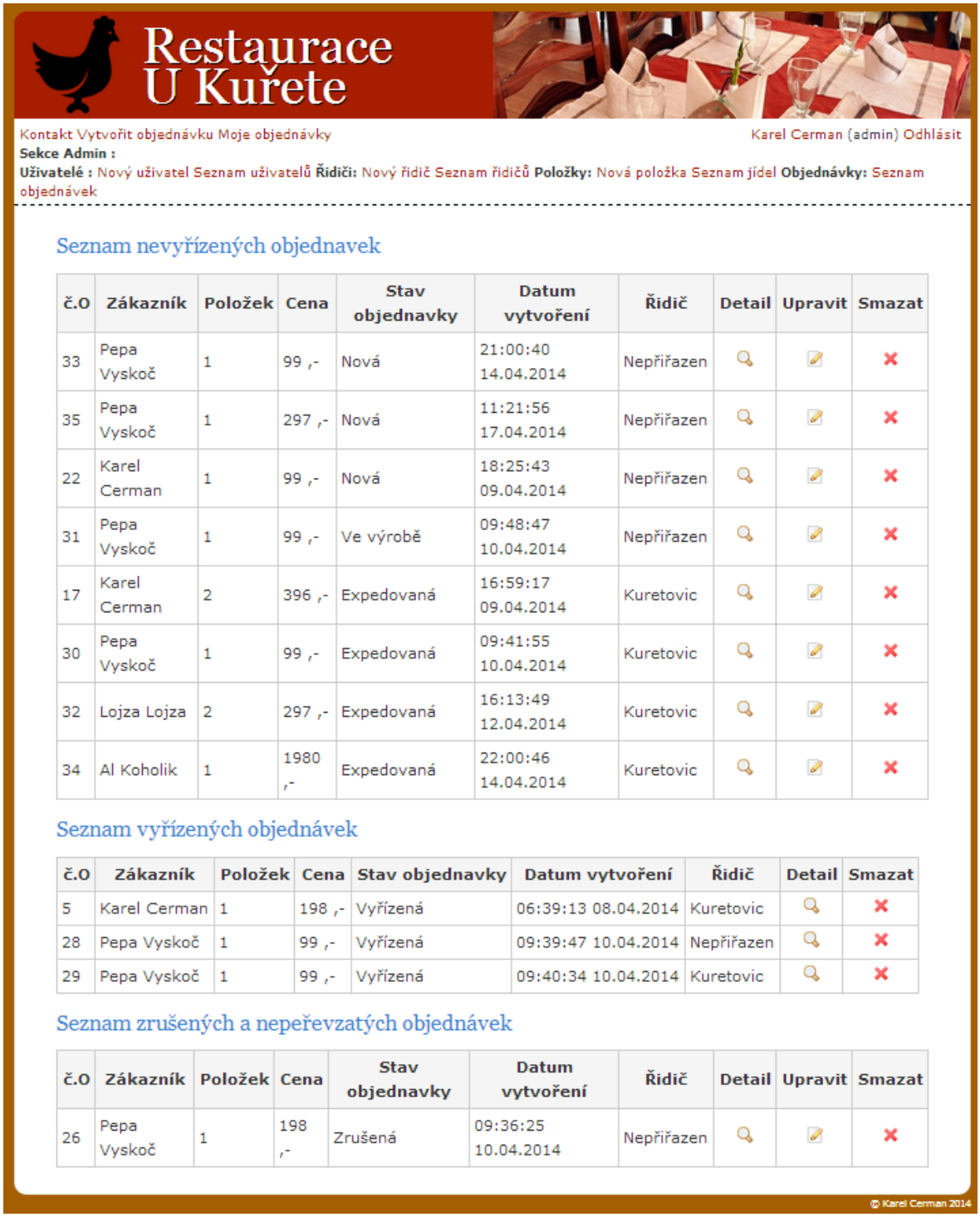
V této příloze je uveden zdrojový kód scriptu v jazyce JavaScript, který zobrazuje nejkratší okružní cestu v Google mapě.

```
<script
src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false"></scri
pt>
    <script>
    var directionsDisplay;
    var directionsService = new google.maps.DirectionsService();
    var map;

    function initialize() {
      directionsDisplay = new google.maps.DirectionsRenderer();
      var upce = new google.maps.LatLng(50.033941, 15.767514);
      var mapOptions = {
      zoom: 13,
      center: upce,000000000000000000
    }
      map = new google.maps.Map(document.getElementById('map-canvas'),
mapOptions);
      directionsDisplay.setMap(map);
    }

    function calcRoute() {
      var start = document.getElementById('start').value;
      var end = document.getElementById('end').value;
      var waypts = [];
      var checkboxArray = document.getElementById('waypoints');
        for (var i = 0; i < checkboxArray.length; i++) {
          if (checkboxArray.options[i].selected == true) {
          waypts.push({
          location:checkboxArray[i].value,
          stopover:true});
          }
        }
    }
  var request = {
      origin: start,
      destination: end,
      waypoints: waypts,
      optimizeWaypoints: true,
      travelMode: google.maps.TravelMode.DRIVING
  };
  directionsService.route(request, function(response, status) {
    if (status == google.maps.DirectionsStatus.OK) {
      directionsDisplay.setDirections(response);
    }
  });
 }

google.maps.event.addDomListener(window, 'load', initialize);
    </script>
```

Příloha C – Snímky z vyvinuté aplikace



Obrázek 22 - Ukázka vygenerované optimální okružní trasy



č.O	Zákazník	Položek	Cena	Stav objednavky	Datum vytvoření	Řidič	Detail	Upravit	Smazat
33	Pepa Vyskoč	1	99 ,-	Nová	21:00:40 14.04.2014	Nepřiřazen			
35	Pepa Vyskoč	1	297 ,-	Nová	11:21:56 17.04.2014	Nepřiřazen			
22	Karel Cerman	1	99 ,-	Nová	18:25:43 09.04.2014	Nepřiřazen			
31	Pepa Vyskoč	1	99 ,-	Ve výrobě	09:48:47 10.04.2014	Nepřiřazen			
17	Karel Cerman	2	396 ,-	Expedovaná	16:59:17 09.04.2014	Kuretovic			
30	Pepa Vyskoč	1	99 ,-	Expedovaná	09:41:55 10.04.2014	Kuretovic			
32	Lojza Lojza	2	297 ,-	Expedovaná	16:13:49 12.04.2014	Kuretovic			
34	Al Koholik	1	1980 ,-	Expedovaná	22:00:46 14.04.2014	Kuretovic			

č.O	Zákazník	Položek	Cena	Stav objednavky	Datum vytvoření	Řidič	Detail	Smazat
5	Karel Cerman	1	198 ,-	Vyřízená	06:39:13 08.04.2014	Kuretovic		
28	Pepa Vyskoč	1	99 ,-	Vyřízená	09:39:47 10.04.2014	Nepřiřazen		
29	Pepa Vyskoč	1	99 ,-	Vyřízená	09:40:34 10.04.2014	Kuretovic		

č.O	Zákazník	Položek	Cena	Stav objednavky	Datum vytvoření	Řidič	Detail	Upravit	Smazat
26	Pepa Vyskoč	1	198 ,-	Zrušená	09:36:25 10.04.2014	Nepřiřazen			

Obrázek 23 - Ukázka seznamu objednávek