UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY
A INFORMATIKY

# BAKALÁŘSKÁ PRÁCE

2008 Petr Vytlačil

*Univerzita Pardubice*

*Fakulta elektrotechniky a informatiky*

# Tvorba a implementace jednoduché webové aplikace „Najdi si parťáka" pro seznamování lidí se společným zájmem pro sport.

Petr Vytlačil

Bakalářská práce

2008

**Univerzita Pardubice**
**Fakulta elektrotechniky a informatiky**
**Katedra informačních technologií**
Akademický rok: **2007/2008**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Petr VYTLAČIL**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Tvorba a implementace jednoduché webové aplikace „Najdi si parťáka" pro seznamování lidí se společným zájmem pro sport.**

Zásady pro vypracování:

Cílem práce je vytvořit informační systém pro seznamování lidí se společným zájmem pro sport. Teoretická část bude obsahovat popis webového frameworku CakePHP jeho vlastnosti, výhody a ukázka použití na jednoduchém příkladu. Implementační část představuje návrh a tvorbu konkrétního informačního systému s využitím frameworku CakePHP.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

**Hlavenka a kol. Vytváříme WWW stránky a spravujeme moderní web site.**
**Sedláček J. E-komerce internetový a mobil marketing - od A do Z**
**Gutmans, A. Mistrovství v PHP 5**
**Smička R. Optimalizace pro vyhledávače - SEO**
**Nondek, L. Řenčová, L. Komerční využití Internetu**

Vedoucí bakalářské práce: **Ing. Jana Holá, Ph.D.**
Ústav elektrotechniky a informatiky

Datum zadání bakalářské práce: **30. listopadu 2007**

Termín odevzdání bakalářské práce: **16. května 2008**

Univerzita Pardubice
Česká republika
L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

V Pardubicích dne 29. dubna 2008

**SOUHRN**

*Cílem práce je vytvořit jednoduchou webovou aplikaci „Najdi si parťáka" pro seznamování lidí se společným zájmem pro sport. Pro realizaci aplikace je použit webový framework CakePHP 1.2, jehož vlastnosti a základní použití je popsáno v teoretické části.*

**KLÍČOVÁ SLOVA**

*Internet, MySQL, PHP, CakePHP, MVC*

**TITLE**

*Web Application "Find a Buddy" for Meeting People that is based on Common Sport Interest.*

**ABSTRACT**

*The goal of this work is to create a simple web application "Find a Buddy" for Meeting People that is based on Common Interest of Sport. For the implementation application is used web framework 1.2 CakePHP. Characteristics of this web framework and the basic use are described in the theoretical part.*

**KEYWORDS**

*Internet, MySQL, PHP, CakePHP, MVC*

# OBSAH

**1 Úvod ..... 9**

1.1 Obsah práce ..... 9

1.2 Popis a vlastnosti webové aplikace „Najdi si parťáka" ..... 9

1.2.1 Obecná charakteristika ..... 10

1.2.2 Funkce aplikace ..... 10

1.2.3 Nároky na uživatele pro používání aplikace ..... 11

**2 Webový framework CakePHP 1.2 ..... 12**

2.1 Co je to CakePHP ..... 12

2.2 Srovnání CakePHP s ostatními frameworky ..... 14

2.3 Podmínky nutné pro použití frameworku CakePHP ..... 15

2.4 Model View Controller architektura ..... 15

2.4.1 Princip MVC ..... 16

**3 Obecný postup tvorby aplikace postavené na frameworku CakePHP 1.2 ..... 18**

3.1 Instalace ..... 18

3.2 Adresářová struktura ..... 19

3.3 Adresář app ..... 19

3.4 Nastavení připojení k databázi ..... 19

3.5 Nastavení bezpečnostní konstanty pro sessions ..... 20

3.6 Test správné instalace frameworku ..... 21

3.7 ER diagram databáze - ilustrační příklad ..... 21

3.8 Modely a Active Record ..... 22

3.8.1 Definice relací mezi tabulkami ..... 24

3.8.2 Práce s daty – zápis, čtení, mazání a validace ..... 26

3.9 Vrstva s řadiči ..... 27

3.10 Prezentační vrstva ..... 29

**4 Google Maps API ..... 32**

4.1 Nastavení Google Maps API ..... 32

**5 Návrh a realizace webové aplikace „Najdi si parťáka" ..... 34**

5.1 Požadavky na webovou aplikaci „Najdi si parťáka" ..... 34

5.2 Technologie použité při realizaci aplikace „Najdi si parťáka" ..... 35

5.3 Nástroje použité při realizaci aplikace „Najdi si parťáka" ..... 35

5.3.1 TopStyle Pro verze 3.5 ..... 36

5.3.2 DBDesigner ..... 36

5.3.3 jEdit verze 4.3 ..... 37
5.3.4 Adobe Photoshop CS2 ..... 38
5.4 Adresářová struktura aplikace ..... 39
5.5 Návrh databáze ..... 40
5.5.1 ER diagram databáze ..... 40
5.5.2 Popis tabulek databáze ..... 41
5.6 Vrstva s modely aplikace ..... 45
5.7 Prezentační vrstva a design aplikace ..... 48
5.7.1 Struktura webové aplikace ..... 48
5.7.2 Design aplikace ..... 49
5.8 Vrstva s radiči – řízení logiky aplikace „Najdi si parťáka" ..... 50
5.8.1 Autentizace a autorizace uživatele ..... 50
5.8.2 Vyhledávání v poptávkách podle zvolených kriterií ..... 52
**6 Základní obsluha webové aplikace „Najdi si parťáka" uživatelem ..... 54**
6.1 Vytvoření vlastního účtu a jeho aktivace ..... 54
6.2 Přihlášení uživatele ..... 56
6.3 Přidání nové poptávky po parťákovi ..... 56
**7 Závěr ..... 58**
**Použité zdroje ..... 59**
**Přílohy ..... 62**
Příloha A – UML diagram tříd řadičů ..... 62
Příloha B – USE CASE diagram používání aplikace ..... 63

# Seznam použitých symbolů a zkratek

| *Zkratka* | *Anglický význam* | *Český význam* |
|---|---|---|
| AJAX | Asynchronous JavaScript and XML | Tech. pro vývoj interaktivních web. aplikací |
| API | Application programming interface | Rozhraní pro programování aplikací |
| CSS | Cascading Style Sheets | Kaskádové styly |
| HTML | HyperText Markup Language | Jazyk pro tvorbu webových stránek |
| MVC | Model-View-Controller | Model-Pohled-Řadič |
| PHP | PHP: Hypertext Preprocessor | PHP: Hypertextový preprocesor |
| SQL | Structured Query Language | Strukturovaný dotazovací jazyk |
| UML | Unified Modeling Language | Grafický jazyk pro vizualizaci, spec., navrhování |
| XML | eXtensible Markup Language | Rozšiřitelný značkovací jazyk |

# 1 Úvod

## 1.1 Obsah práce

Cílem této práce je vytvořit jednoduchou webovou aplikaci „Najdi si parťáka", která je určena k vzájemnému seznamování a vyhledávání takzvaných parťáků pro sport. Například, někdo hledá parťáka pro pravidelný trénink na horském kole nebo běh, svoji poptávku zadá do systému a na mapě označí místo, kde chce sportovat. Ostatní návštěvníci si budou moc vyhledávat v záznamech a filtrovat je podle různých kritérií (sport, místo, atd.), najít potenciálního parťáka a posléze ho oslovit.

První část práce je teoretická a popisuje technologie, které byly použity pro vývoj aplikace. Převážná část teoretické části se zabývá vlastnostmi a základním použití webového frameworku[1] CakePHP 1.2, za pomoci kterého byla aplikace „Najdi si parťáka" naprogramována. Ve zbývající části je popsáno základní použití Google Maps API[2], které umožňuje práci s mapami a jejich použití ve webových aplikacích.

## 1.2 Popis a vlastnosti webové aplikace „Najdi si parťáka"

V této kapitole je popsána obecná charakteristika aplikace „Najdi si parťáka", soupis funkcí, které má aplikace poskytovat uživatelům a nároky na uživatele, aby mohli aplikaci používat.

---

[1] Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat knihovny, návrhové vzory nebo doporučené postupy při vývoji (1).

[2] API je rozhraní pro programování aplikací, sbírka procedur, funkcí či tříd nějaké knihovny. (2)

### 1.2.1 Obecná charakteristika

Obecná charakteristika Webová aplikace „Najdi si parťáka”:

- Aplikace je provozována na internetové adrese *www.najdisipartaka.cz.*
- Všichni uživatelé mohou aplikaci využívat zcela zdarma.
- Aplikace musí být uživatelsky přívětivá – přehledné grafické rozhraní s intuitivním ovládáním.
- Bezpečnost aplikace – nikdo nesmí mít možnost mazat, editovat nebo mít přístup k informacím, které mu nepatří či, které nejsou veřejné pro ostatní uživatele.

### 1.2.2 Funkce aplikace

Funkcionalita aplikace vychází z USE CASE diagramu, který je v příloze 1A. Přehled funkcí, které aplikace poskytuje:

- Uživatel má možnost registrace, při registraci musí zadat heslo, kontaktní e-mail, věk, pohlaví, bydliště a vybrat si přezdívku, pod kterou bude vystupovat.
- Po registraci uživatel obdrží e-mail s aktivačním kódem, pomocí kterého si musí aktivovat svůj účet.
- Přihlášení uživatel může přidávat nové poptávky po parťácích, editovat je a mazat.
- Přihlášení uživatel může reagovat na poptávky ostatních uživatelů a kontaktovat je odesláním zprávy.
- Uživatelé aplikace, mají možnost filtrovat a hledat poptávky, které byly již zadány ostatními uživateli.

### 1.2.3 Nároky na uživatele pro používání aplikace

Aplikace je určena pro používání na PC s monitorem LCD nebo CRT. Je nutné, aby uživatel měl nainstalovaný webový prohlížeč (Internet Explorer 6 nebo vyšší, Mozilla Firefox 1.5 nebo vyšší, Opera 9 nebo vyšší) s podporou JavaScriptu. Aplikace není přizpůsobena pro používání mobilními zařízeními.

# 2 Webový framework CakePHP 1.2

V této kapitole jsou popsány vlastnosti a přednosti webového frameworku CakePHP 1.2, který jsem si zvolil pro vývoj webové aplikace „Najdi si parťáka".

## *2.1 Co je to CakePHP*

CakePHP je open source[3] framework učený pro vývoj webových aplikací založených na programovacím jazyku PHP verze 4 nebo 5. CakePHP je postaven na vzoru MVC[4] a ORM[5]. (3)

Tento framework je velmi oblíbeným mezi programátory, protože pomocí něho je jednoduché vyvíjet aplikace, má dobře zpracovanou dokumentaci, ale především podporuje řadu technologií:

- ***MVC a ORM*** - Zefektivňují práci programátora a současně mu dávají do rukou silný vývojový nástroj, díky kterému programátor ušetří čas a zdrojový kód aplikace je přehlednější.
- ***Týmová integrace*** - Programátor může vyvíjet datovou a aplikační vrstvu zatímco designér bude vytvářet uživatelské prostředí aplikace.
- ***Rozsáhlá podpora databází*** - MySQL (4 nebo vyšší), PostgreSQL, Firebird DB2, Microsoft SQL, Oracle, SQLite, ODBC, ADOdb.
- ***Podpora AJAX***[6] - CakePHP poskytuje podporu pro vývoj interaktivních webových aplikací. Vývojář nemusí psát složité kódy JavaScriput, ale

---

[3] Open-source software (OSS) je počítačový software s otevřeným zdrojovým kódem. (4)

[4] Model-View-Controller (MVC) je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má minimální vliv na ostatní. (5)

[5] ORM (objektově relační mapování), což není nic jiného, než, že tabulku v databázi převedeme na objekt. (6)

používá předefinované komponenty, které mu práci usnadní. CakePHP primárně podporuje AJAX postavený na JavaScriptovém[7] frameworku Prototype, ale není problém použít jiný JavaScriptový framework (jQuery, atd.).

- ***Podpora SEO[8]*** - Framework poskytuje možnost pro tvorbu „hezkých" URL (odkazů).

- ***Podpora i18n[9] a l10n[10]*** - Framework zahrnuje kompletní podporu pro tvorbu vícejazyčných aplikací.

- ***Podpora Scaffoldingu*** - CakePHP podporuje scaffolding což znamená, že vývojář si muže nechat frameworkem automaticky generovat rozhraní pro základní operace s daty jako je mazání, přidávání, editace a zobrazování záznamů z databáze.

- ***Baker generátor zdrojového kódu*** - Baker je speciální nástroj, který programátorovi poskytuje při vývoji aplikace úsporu času, protože nemusí psát běžný zdrojový kód. Pomocí Bakeru je možné generovat modely, prezentační vrstvu a řadiče. Použití je velmi jednoduché, při generování modelu se zvolí konkrétní tabulka v databázi, Baker sám nabízí nastavení relací mezi ostatními modely (tabulkami), validační kriteria, potom

---

[6] AJAX je technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání. (7)

[7] JavaScript je programovací jazyk, který se používá v internetových stránkách. Zapisuje se přímo do HTML kódu, což je velká výhoda, protože je to jednoduché. (8)

[8] Search Engine Optimization - optimalizace pro vyhledávače. Cílem SEO je vylepšit pozice ve vyhledávačích. (9)

[9] Příprava aplikace na podporu různých kulturních zvyklostí se nazývá internacionalizace, používá se zkratka i18n. (10)

[10] Úprava internacionalizované aplikace pro konkrétní jazyk se nazývá lokalizace, spočívá především v přeložení textů do konkrétního jazyka a využívá se pro ní zkratka l10n. (10)

vygeneruje zdrojový kód pro model. Takto podobně lze pomocí Bakeru generovat řadiče a prezentační vrstvu.

- ***Rozsáhlá základna uživatelů a dobrá dokumentace*** - CakePHP se stal velmi oblíbeným frameworkem, a proto je k dispozici velké množství návodů, komponent rozšiřujících jeho možnosti a diskusních fór, kde vám ostatní vývojáři poradí při řešení složitějších úloh nebo problémů.

## 2.2 Srovnání CakePHP s ostatními frameworky

Přehled vlastností a podpory technologií frameworku CakePHP a ostatních nejpoužívanějších frameworků je v tabulce 1.

| Framework | CakePHP | CodeIgniter | Prado | Symfony | Zend |
|---|---|---|---|---|---|
| PHP4 | ano | ano | - | - | - |
| PHP5 | ano | ano | ano | ano | ano |
| MVC | ano | ano | ano | ano | ano |
| Multiple DB's | ano | ano | ano | ano | ano |
| ORM | ano | - | ano | ano | ano |
| DB Objects | ano | ano | ano | ano | ano |
| Templates | - | ano | ano | - | - |
| Caching | ano | ano | ano | ano | ano |
| Validation | ano | ano | ano | ano | ano |
| AJAX | ano | - | ano | ano | - |
| Modules | ano | - | ano | ano | ano |

**Tabulka 1: Srovnání CakePHP s ostatními nejoblíbenějšími frameworky (11)**

**Popis termínů použitých ve srovnávací tabulce**

- ***MVC*** - Model-View-Controller architektury.
- ***Multiple DB's*** - Možnost použití více databází naráz.
- ***ORM*** - Podpora (objektového relačního mapování).
- ***DB Objects*** - Objektový přístup k datům.
- ***Templates*** - Podpora šablon.

- ***Caching*** - Možnost cachovat objekty a získané data z databáze.
- ***Validation*** - Validace vstupní dat.
- ***AJAX*** - Podpora pro vývoj interaktivních aplikací.
- ***Modules*** - Podpora modulů pro rozšíření možností frameworku.

Z porovnání frameworků v tabulce 1, je patrné, že frameworky disponují podobnými vlastnostmi a podporou technologií. Důvodem proč jsem si pro vývoj webové aplikace „Najdi si parťáka" zvolil framework CakePHP je ten, že s ním mám dobré zkušenosti, je pro něj k dispozici řada rozšiřujících komponent a disponuje generátorem zdrojového kódu, který ušetří při realizaci aplikace spoustu času.

## 2.3 Podmínky nutné pro použití frameworku CakePHP

Pro použití CakePHP, musí být splněny tyto podmínky:

- HTTP server (nejlépe Apache) s podporou sessions a mod_rewrite[11].
- PHP verze 4.3.2 nebo vyšší.
- Databázový systém (CakePHP 1.2 podporuje databáze DB2, Firebird, MSSQL, MySQL (mysql i mysqli), ODBC, Oracle, PostgreSQL, SQLite, Sybase) (12)

## 2.4 Model View Controller architektura

Model-View-Controller (Model-Pohled-Řadič) je šablona návrhu aplikací, která pomáhá logicky oddělit kód a udělat ho znovupoužitelný, lehce upravovatelný

[11] Modul pro webový server Apache, který umožňuje přesměrovávat a podstrkávat jiné webové adresy podle nadefinovaných pravidel. (13)

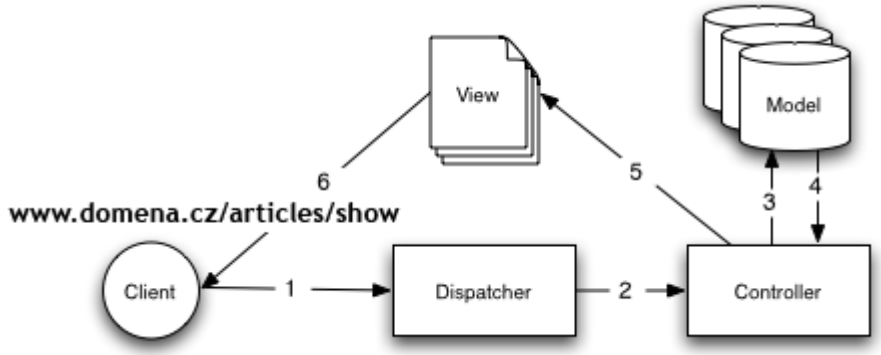
a celkově přehlednější. MVC je způsob rozdělení aplikace, nebo jen části aplikačního rozhraní, na tři vrstvy: Model, View, Controller.

Velkou výhodou MVC je, že chování jednotlivých vrstev, lze snadno upravit nebo dokonce je úplně nahradit jinou komponentou, aniž by bylo nutné dělat rozsáhlé změny ve zdrojovém kódu aplikace. Kvůli těmto výhodám architektury MVC, jsem si volil framework CakePHP 1.2, který architekturu MVC implementuje.

### 2.4.1 Princip MVC

Obecně řečeno, vytváření aplikací s využitím architektury MVC vyžaduje vytvoření tří komponent.

- **Model (model)** - Reprezentace reprezentuje data a informace, s nimiž aplikace pracuje a které jsou prostřednictvím uživatelského rozhraní zpřístupněny uživateli.
- **View (pohled)** - Zpřístupňuje informace reprezentované modelem uživateli.
- **Controller (řadič)** - Řídí celý proces, reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu. (14)




**Obrázek 1: Diagram vyjadřující závislosti v MVC (14)**

Na obrázku 1. je zobrazen proces požadavku klienta na daný URL a jeho zpracování. Sled událostí, které se vykonají, aby klient získal odpověď:

1. klient klikne na odkaz směřující na URL *www.domena.cz/articles/show*, jeho webový prohlížeč vyšle požadavek na webový server.
2. Dispatcher kontroluje požadavek ‘articles/show’ a směřuje ho na controller articles.
3. Controller vykoná požadovanou akci show, získá data z modelu a
4. potom je předá vrstvě View, která jé zpracuje,
5. zformátována data jsou poslány klientovi.

# 3 Obecný postup tvorby aplikace postavené na frameworku CakePHP 1.2

V této kapitole je popsán obecný postup tvorby aplikace pomocí frameworku CakePHP 1.2. V první části je popsána instalace frameworku a jeho adresářová struktura. Ve zbývající části jsou popsány jednotlivé vrstvy třívrstvého modelu MVC, pravidla pro jejich realizaci a základní popis funkcí, které poskytují programátorovi.

## *3.1 Instalace*

Instalace frameworku CakePHP není složitá, pouze stačí stáhnout archív CakePHP a rozbalit. Postup instalace:

1. stáhnout aktuální verzi frameworku z domovské stránky (http://www.cakephp.org),
2. rozbalit archív do adresáře DocumentRoot[12] , který je dostupný z adresy http://localhost,
3. za předpokladu použití databáze v aplikaci, je nutná konfigurace připojení k databázi a jádra frameworku.

[12] DocumentRoot je kořen adresářového stromu, ve kterém server Apache hledá dokumenty.

## 3.2 Adresářová struktura

Popis adresářové struktury frameworku, po jeho rozbalení:

| | |
|---|---|
| **/app** | vlastní soubory aplikace (models, views, controllers, CSS, img) |
| **/cake** | jádro CakePHP (knihovny CakePHP) |
| **/docs** | můžete smazat, zde nic důležitého není |
| **/vendors** | můžete smazat, zde nic důležitého není |
| .htaccess | konfigurační soubor pro úpravu nastavení web serveru |
| index.php | základní soubor nutný pro chod frameworku |

## 3.3 Adresář app

Složka app obsahuje zdrojové kódy aplikace, konfigurační soubory frameworku, kontroléry, modely a adresář webroot, kam se ukládají CSS styly, obrázky a JavaScript kódy. Adresářová struktura aplikace je popsána v kapitole 5.4.

## 3.4 Nastavení připojení k databázi

Nastavení databáze se nachází v souboru *database.php.default*, který je umístěn ve složce *app/config*. Je nutné soubor přejmenovat na *database.php*. Soubor obsahuje část zdrojového kódu, jenž je uvedený na obrázku 2.

```
<?php.
class DATABASE_CONFIG {.
    var $default = array(.
        'driver' => 'mysql', .
        'persistent' => false,.
        'host' => 'localhost',.
        'port' => '',.
        'login' => 'user', .
        'password' => 'password',.
        'database' => 'project_name',.
        'schema' => '',.
        'prefix' => '',.
        'encoding' => 'utf8'.
    );.
}.
?>.
```

**Obrázek 2: Obsah konfiguračního souboru database.php**

**Popis nejdůležitějších parametrů pro nastavení připojení k databázi**

- **Driver** – Typ databáze (mysql, postgresql).
- **Host** – Server, na kterém běží databáze, nejčastěji localhost.
- **Port** – Port, na kterém běží databáze.
- **Login** – Uživatelské jméno pro přístup k databázi.
- **Password** – Heslo pro přístup do databáze
- **Database** – Název databáze.
- **Prefix** – Prefix, který je používán u názvu tabulek.
- **Encoding** – Kódování pro přístup k datům v databázi.

## 3.5 Nastavení bezpečnostní konstanty pro sessions

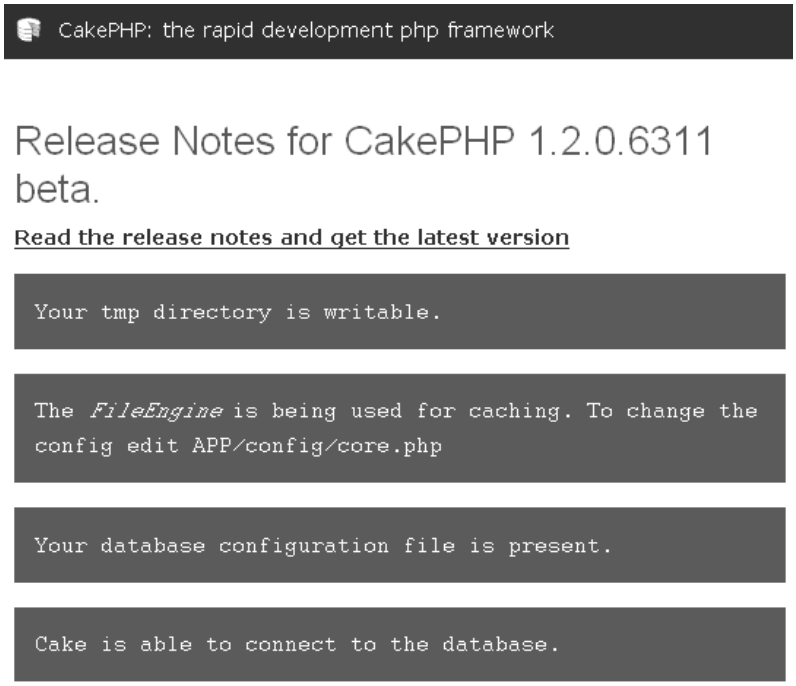
Ve složce *app/config* je konfigurační soubor *core.php*, ve kterém se konfiguruje jádro frameworku. Nejdůležitější část se nachází na řádku 153, zde se nastavuje hashovaní konstanta, která se využívá pro zvýšení bezpečnosti a zamezení odcizení sessions. Ukázka nastavení hash konstanty je na obrázku 3.

```
Configure::write('Security.salt', 'DYhG93b0qyJfIxfs2guVoUub.
WwvniR2G0FgaC9minajdisipartaka');.
```

**Obrázek 3: Nastavení sessions konstanty na řádku 153 v souboru core.php**

## 3.6 Test správné instalace frameworku

Zadáním adresy *http://localhost* do prohlížeče, se zobrazí informační stránka o stavu instalace, jestliže je framework správně nainstalován stránka by měla odpovídat obrázku 4.



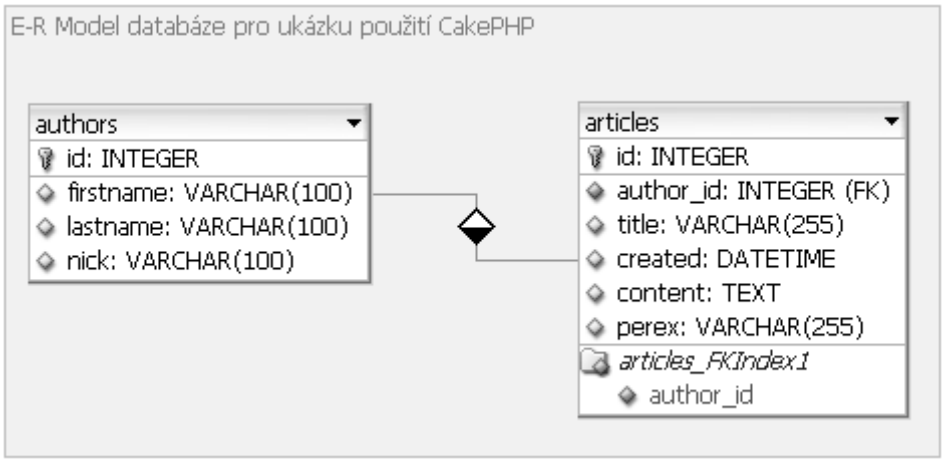

**Obrázek 4: Úvodní obrazovka indikující nastavení frameworku**

## 3.7 ER diagram databáze - ilustrační příklad

Pro názornou ukázku je zvolen jednoduchý příklad databáze, která obsahuje tabulky **authors** a **arcticles**, E-R diagram tabulek je na obrázku 5. Pro tvorbu tabulek musí být dodrženy tyto konvence, které CakePHP vyžaduje:

- názvy tabulek a cizí klíče pojmenované v anglickém jazyce,
- názvy tabulek musí být v množném čísle,
- cizí klíče jsou v jednotném čísle a končí „_id“,
- primární unikátní klíč tabulky musí mít název „id“.

Tyto konvence nejsou dogmatem, ve specifikaci modelu je možné nastavit název tabulky i cizích klíčů, ale pro pohodlnější vývoj aplikace je vhodnější dodržet doporučenou konvenci.




**Obrázek 5: ER diagram databáze pro ukázku použití CakePHP**

## 3.8 Modely a Active Record

Modely reprezentují jednotlivé tabulky. V CakePHP model implementuje návrhový vzor[13] Active Record, který mapuje databázové tabulky na třídy, řádky na objekty a sloupce na jejich atributy (toto však v Cake není přesně implementováno - rozdíl je v tom, že v současné verzi CakePHP 1.2 se řádky a sloupce mapují na dvojrozměrné pole, mapování na objekty je plánované až ve verzi 2.0). Active Record umožňuje definovat vztahy mezi tabulkami (hasMany, hasOne, hasManyAndBelongsTo, belongsTo) a kromě toho obsahuje výkonné nástroje pro práci s daty, jako je zápis, čtení a validace.

Jednotlivé modely se ukládají do adresáře *app/models*. Název modelu odpovídá názvu tabulky ale v jednotném čísle, například model pro tabulku *authors* musí být pojmenován Author a zdrojový soubor se nazývat *author.php***.** Ukázka jak by měl vypadat zdrojový kód modelu Author je na obrázku 6.

[13] Návrhový vzor představuje obecné řešení problému, které se využívá při návrhu programů

```php
<?php
class Author extends AppModel {

    /* Název modelu používá se pro přístup v controlleru. */
    var $name = 'Author';

    /* Validační kritéria pro vstupní data */
    var $validate = array(
        'id' => array('_extract'),
        'firstname' => array('blank'),
        'lastname' => array('blank'),
        'nick' => array('blank')
    );

    /* Nastavené typu relace s ostaními tabulkamy */
    var $hasMany = array(
            'Article' => array('className' => 'Article',
                               'foreignKey' => 'author_id'
            )
    );

}
?>
```

**Obrázek 6: Ukázka zdrojového kódu modelu Author**

### Popis atributů modelu Author

- **name:** Název modelu, pomocí kterého se v kontroléru přistupuje k funkcím modelu.
- **validate**: Pole, v němž se nastavují validační kritéria pro ukládání dat do databáze.
- **hasMany**: Jedná se o typ relace s druhou tabulkou, jedná se o typ relace N:1.

Ukázka zdrojového kódu modelu Article je na obrázku 7.

```php
<?php
class Article extends AppModel {

    /* Název modelu používá se pro přístup v controlleru. */
    var $name = 'Article';

    /* Validační kritéria pro vstupní data. */
    var $validate = array(
        'id' => array('_extract'),
        'author_id' => array('alphaNumeric'),
        'title' => array('blank'), // nesmí být prázdný
        'content' => array('blank'),
        'perex' => array('blank')
    );

    /* Nastavení typu relace s ostatními tabulkami */
    var $belongsTo = array('Author');
}
?>
```

**Obrázek 7: Ukázka zdrojového kódu modelu Article**

### 3.8.1 Definice relací mezi tabulkami

**hasOne** odpovídá relaci typu 1:1, například uživatel může mít jen jeden profil. Ukázka zdrojového kódu pro nastavení relace je na obrázku 8.

```php
<?php
/*
    Ukázkový model s použitím hasOne.
    Uživatel může vá jen jeden profil.
    Typ relace 1:1.
*/
class User extends AppModel {
    var $name = 'User';
    var $hasOne = 'Profile';
}

?>
```

**Obrázek 8: Ukázka zdrojového kódu pro nastavení relace hasOne**

**hasMany** představuje relaci typu 1:N, například uživatel může mít n-počet komentářů. Ukázka nastavení relace v modelu je na obrázku 9.

```
<?php
/*
    Ukázkový model s použitím hasMany.
    Uživatel může mít hodně komentářů.
    Typ relace 1:N.
*/
class User extends AppModel {
    var $name = 'User';
    var $hasMany = 'Comment';
}
```

**Obrázek 9: Ukázka zdrojového kódu pro nastavení relace hasMany**

**belongsTo** se používá, když některý model náleží druhému. Například mezi uživatelem a profilem je relace typu 1:1, uživatel má jeden profil, profil patří uživateli. Ukázka nastavení relace je na obrázku 10.

```
<?php
/*
    Ukázkový model s použitím belongsTo.
    Profil patří uživateli.
*/
class Profile extends AppModel {
    var $name = 'Profile';
    var $belongsTo = 'User';
}
?>
```

**Obrázek 10: Ukázka zdrojového kódu pro nastavení relace belongsTo**

**hasAndBelongsToMany** jedná se o relaci typu N:M, učitel může učit více žáků a žáci mohou mít více učitelů. Ukázka nastavení relace v modelu je na obrázku 11.

```php
<?php
/*
    Ukázkový model s použitím hasAndBelongsToMany.
    Uživatel může mít hodně komentářů.
    Typ relace N:M.
*/
class Teacher extends AppModel {
    var $name = 'Teacher';
    var $hasAndBelongsToMany = array(
        'Student' =>
            array('className'            => 'Student',
                'joinTable'              => 'teachers_students',
                'foreignKey'             => 'student_id',
                'associationForeignKey'  => 'teacher_id',
            )
        );
}

?>
```

**Obrázek 11: Ukázka zdrojového kódu pro nastavení relace hasAndBelongsToMany**

### 3.8.2 Práce s daty – zápis, čtení, mazání a validace

Nejdůležitější a podstatná věc, kterou Active Record v modelu přináší je, značné ulehčení práce s daty, díky tomu se nemusí psát běžné SQL[14] příkazy, ale jsou k dispozici funkce, které Active Record poskytuje.

**Základní funkce pro získání dat:**

**find(**string $conditions, array $fields, string $order, int $recursive**)**

- vrací pouze první záznam odpovídající podmínce v proměnné $conditions,
- *příklad požití*: `$this->Author->find();` vrátí jednoho autora,

**findAll(**string $conditions, array $fields, string $order, int $limit, int $page, int $recursive**)**

- vrací data se specifikovanými poli $fields do limitu $limit,
- *příklad požití*: `$this->Author->findAll()`, vrátí všechny autory,

**findAllBy<fieldName>(**string $value**)**

- vrací všechny záznamy odpovídajících hodnotě $value v poli fieldName.
- *příklad požití*: `$this->author->findAllByLastname("petr");`

[14] SQL je dotazovací jazyk používaný pro práci s daty v relačních databázích.

**findBy<fieldName>(**string $value**)**

- vrací první záznam odpovídajících hodnotě $value v poli fieldName.

**query(**string $query**)**

- vykoná a vrátí data SQL příkazu v proměnné $query.
- *Příklad požití*: `$this->Author->query("SELECT * FROM users")`, vrátí všechny autory.

**Funkce pro mazání dat:**

**del(**int $id = null, boolean $cascade = true**)**

- odstraní záznam s primárním klíčem $id, proměnná $cascade definuje jestli se mají mazat i závislé záznamy.
- *příklad požití*: `$this->Author->del(1,true)`, odstraní autora s id 1 i všechny jeho články,

**deleteAll(**mixed $conditions, $cascade = true**)**

- odstraní všechny data odpovídající podmínce $conditions.

**Funkce pro vkládání dat:**

**save(**array $data = null, $validate = true, $fieldList = array()**)**

- uloží pole dat, $validate určuje, jestli se mají data validovat před uložením do databáze, $fieldList udává jaké pole se mají uložit.

## 3.9 Vrstva s řadiči

Tato vrstva se skládá z kontrolérů, neboli řadičů, které se používají k řízení logiky aplikace. Řadič se většinou stará o řízení logiky jednoho modelu. V CakePHP se řadič jmenuje podle názvu modelu, ale v množném čísle. Například pro řízení logiky a práci s články by se řadič nazýval ArticlesController. Jednotlivé řadiče se ukládají do složky *app/controllers*, ArticlesController musí mít název souboru *articles_controller.php*. Tyto řadiče mohou obsahovat libovolný počet metod, které se označují jako akce. Tyto akce se využívají k zobrazení takzvaných views, které již spadají pod prezentační vrstvu. Dispečer v CakePHP vykoná určitou akci, v okamžiku kdy uživatel vznese požadavek prostřednictvím URL v prohlížeči.

Například dispečer z URL *http://localhost/articles/show* ví, že má vykonat akci *show*, která je nadefinovaná v řadiči *ArticlesController*. Ukázka kontroléru pro model Article:

```php
<?php
    // /app/controllers/articles_controller.php
    class ArticlesController extends AppController {
        // název controlleru
        var $name = 'Articles';

        // výčet helpru pro views, které se budou používat
        var $helpers = array('Html', 'Form');

        // komponenty, které se budou využívat v controlleru
        var $components = array('Email');

        // definuje přístup k metodám modelů
        var $uses = array('Article','RateArticle');

        /* akce show - získa článek s daným ID a uloží
        do proměnné $article pomocí, které bude prezentační
        vrstva pracovat s daty. */
        function show($id)     {
            $article = $this->Article->find($id);
            $this->set('article',$article);
        }

        /* akce index - získa všecny články a uloží do
        proměnné $articles pomocí, které bude prezentační vrstva
        pracovat s daty. */
        function index() {
            $articles = $this->Article->findAll();
            $this->set('articles',$articles);
        }
    }
?>
```

**Zdrojový kód 9: Řadič ArticlesController**

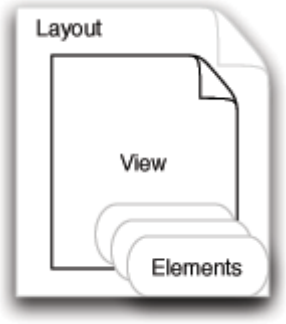
### Popis atributů kontroleru ArticlesController

- **name:** Název kontroléru, je důležité použít v případě PHP 4, aby CakePHP mohl mapovat data z modelu.
- **helpers**: Definuje, které helpery se budou moc využívat v prezentační vrstvě.
- **componets**: Definuje komponenty, které lze používat v kontroléru, například odesílání emailu, nebo vlastní komponenty.

- **uses**: Definuje modely, které je možné používat v kontroléru, abychom mohli pomoci jejich metod získávat data.

## 3.10 Prezentační vrstva

Prezentační (view) vrstva má na starost zobrazit data uživateli, které jí předává kontrolér. Nejčastěji se data zobrazují uživateli pomocí značkovacího jazyka (X)HTML[15], ale mohou se prezentovat jako prostý text nebo PDF[16], XML[17] atd. Prezentační vrstva v CakePHP se může skládat ze tří částí: layout, view a elementy. Tyto části jsou zobrazeny na obrázku 4.




**Obrázek 12: Tři části, z kterých se skládá prezentační vrstva**

**Vlastnosti a popis layout**

- layout obsahuje obecné prvky, které jsou pro všechny views společné. Například hlavička, patička, levé menu a cesty ke stylům.
- Díky layoutu není problém mít jiné grafické rozhraní pro klientskou část aplikace a pro administrační část, u akce kontroléru lze nastavit, jaký layout se má použít.
- Layouty se ukládají do složky *app/views/layouts/*.

---

[15] (X)HTML je značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW. (15)

[16] Přenosný formát dokumentů nezávisle na softwaru i hardwaru, na kterém byly pořízeny.(16)

[17] XML je obecný značkovací jazyk, umožňující snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat. (17)

- Layout má tři proměnné: **$scripts_for_layout:** vkládá JavaSriptové knihovny nebo CSS styly, které jsou nadefinované v kontroléru, **$content_for_layout** vkládá obsah views pro danou akci a **$title_for_layout** zobrazuje titulek, který je nastaven pro akci v kontroléru.

  Ukázka zdrojového kódu základního layout je na obrázku 13.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<!-- Vkládá titulek, který je nastaven pro view
v controlleru -->
<title><?php echo $title_for_layout?></title>
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
<!-- Vkládá externí JavaScript knihovny definované
u akce v controlleru -->
<?php echo $scripts_for_layout?>
</head>
<body>
<div id="header">
    <div id="menu">...</div>
</div>
<!-- Vkládá se obsah view -->
<?php echo $content_for_layout ?>
<div id="footer">...</div>
</body>
</html>
```

**Obrázek 13: Zdrojový kód základního layoutu**

### Popis tvorby view pro jednotlivé akce řadičů

- view zobrazuje data, které jí jsou předána kontrolérem na základě vykonaní požadované akce uživatelem.
- Každá akce, která chce vracet data, musí mít vlastní view, tyto views se ukládají do složky *app/views/controller/akce.ctp*.
- K předání dat prezentační vrstvě se v kontroléru používá metoda $this->set("nazevPromenneProView","data"), ukázka použití je na obrázku 14.

```php
<?php
    // /app/controllers/articles_controller.php
    class ArticlesController extends AppController {
        // název controlleru
        var $name = 'Articles';

        /* akce show - získa článek s daným ID a uloží
        do proměnné $article pomocí, které bude prezentační
        vrstva pracovat s daty. */
        function show($id)      {
            $article = $this->Article->find($id);
            $this->set('article',$article);
        }
    }
?>
```

**Obrázek 14: Ukázka předání dat vrstvě view z kontroleru**

- Ve view se k datům přistupuje prostřednictvím pojmenování, které bylo definováno v kontroléru. V ukázce je předáno pole article, které obsahuje informace o autorovi a článku:

```
        <hl><?php echo $article['Article']['title']; ?></hl>.
        <p>.
            <span><?php echo $article['Article']['created']; ?></sp
            <span class="autor">.
            <?php echo $article['Author']['firstname']; ?>.
            <?php echo $article['Author']['lastname']; ?>.
            </span>.
        </p>.
        <div class="obsahClanku">.
        <?php echo $article['Article']['content']; ?>.
        </div>.
```

**Obrázek 15: Ukázka výpisu dat ve view, které je uloženo ve složce: app/views/articles/show.ctp**

### Popis tvorby elementů, které se vkládají do view nebo layoutů

- Elementy jsou malé části stránky, které se často opakují a musí se přenášet ze stránky na stránku, například nákupní košík, navigační menu, odhlašovací formulář atd.
- Elementy se ukládají ve složce *app/views/elements*, element se ve view nebo layoutu zobrazí tímto příkazem:

```
<?php echo $this->element('nazevElementu'); ?>
```

# 4 Google Maps API

Společnost Google poskytuje webovou službu Mapy Google, která nabízí výkonnou technologii zobrazení map se snadným ovládáním a informace o firmách včetně jejich umístění, kontaktních informací. Tato služba je volně dostupná na adrese *http://maps.google.cz*. (18)

Pro tuto službu společnost Google poskytuje API, díky kterému je možné službu implementovat do webových aplikací.

Google Maps API je bezplatná služba, kterou je možné používat v jakékoliv webové aplikaci, která je volně dostupná všem uživatelům. API poskytuje řadu utilit pro manipulaci s mapami, přidávání obsahu na mapy prostřednictvím různých služeb, což umožňuje vytvářet robustní aplikace využívající mapy. (19)

Google Maps API jsem použil v aplikaci pro zobrazení míst na mapě, kde uživatelé hledají parťáky.

## *4.1 Nastavení Google Maps API*

Aby bylo možné Google Maps API používat v aplikaci je nutné získat tzv. Google Maps API Key, o který se žádá pomocí registračního formuláře na adrese *http://code.google.com/apis/maps/signup.html*. Klíč je použitelný pouze na předem určené doméně a v předem určeném adresáři, proto je nutné tuto cestu vyplnit v žádosti o klíč. Po vyplnění žádosti je vygenerován klíč s jednoduchou ukázkou použití API.

Pro použití API na webové stránce je potřeba vložit do hlavičky stránky cestu na API, v této cestě se uvádí klíč získaný při žádosti. Na stránku stačí přidat prázdný blok s identifikátorem a nastavenou výškou a šířkou. Ukázka použití je na obrázku 16.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN".
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">.
<html xmlns="http://www.w3.org/1999/xhtml">.
  <head>.
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>.
    <title>Google Maps JavaScript API Example</title>.
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=KLIC".
      type="text/javascript"></script>.
    <script type="text/javascript">.
    //<![CDATA[.
    function load() {.
      if (GBrowserIsCompatible()) {.
        var map = new GMap2(document.getElementById("map"));.
        map.setCenter(new GLatLng(37.4419, -122.1419), 13);.
      }.
    }.
    //]]>.
    </script>.
  </head>.
  <body onload="load()" onunload="GUnload()">.
    <div id="map" style="width: 500px; height: 300px"></div>.
  </body>.
</html>.
```

Vygenerovaný klíč

**Obrázek 16: Ukázka zdrojového kódu použití Google Maps API**

# 5 Návrh a realizace webové aplikace „Najdi si parťáka”

Tato část práce se zabývá popisem návrhu a realizace webové aplikace „Najdi si parťáka”. Abych aplikaci mohl volně poskytnout uživatelům, předplatil jsem webhosting od společnosti Darkmay s.r.o. a zaregistroval doménu *najdisipartaka.cz*. Aplikace je tedy dostupná na internetové adrese *http://www.najdisipartaka.cz*. Hlavním podnětem k realizaci této aplikace bylo, poskytnout lidem nástroj pro vzájemné seznamování s lidmi se zájmem pro sport. Při realizaci aplikace jsem postupoval následovně:

1. návrh USE CASE modelu aplikace,
2. návrh modelu databáze,
3. vytvoření databáze s tabulkami,
4. instalace framworku,
5. vygenerování základní kostry aplikace (modely, kontroléry a view),
6. programování funkcionality aplikace,
7. návrh uživatelského prostředí a designu aplikace.

## *5.1 Požadavky na webovou aplikaci „Najdi si parťáka”*

Webová aplikace musí poskytovat uživatelské rozhraní, které bude pro uživatele přehledné a jednoduše použitelné. Při implementaci uživatelského rozhraní bude vhodně použita technologie AJAX, díky které se docílí pohodlnější ovládání aplikace. Uživatelé budou mít možnost si vytvořit vlastní účet, díky kterému získají možnost přidávat poptávky a posílat si zprávy s ostatními uživateli aplikace. Běžní uživatelé tyto možnosti nebudou mít a budou si moc záznamy jen prohlížet a filtrovat.

Jakmile se uživatel přihlásí pomocí hesla a emailu, budou mu zpřístupněny tyto funkce:

- přidávat neomezený počet poptávek,
- spravovat poptávky,

- kontaktovat ostatní uživatelé,
- číst a mazat přijaté zprávy,
- editovat si profil a heslo.

## 5.2 Technologie použité při realizaci aplikace „Najdi si parťáka"

Statickou část aplikace tvoří značkovací jazyk XHTML 1.1 Strict, který je použit pro tvorbu uživatelského rozhraní. Grafické rozhraní je stylováno pomocí kaskádových stylů CSS[18] a obrázků.

Aplikace je vyvinuta ve webovém frameworku CakePHP 1.2, který je naprogramován v jazyce PHP, tento jazyk je používán i v prezentační vrstvě pro zobrazení dat, které jsou zabaleny do XHTML kódu. Veškeré data jsou ukládány v databází MySQL 5. Tato databáze byla zvolena, protože je běžnou součástí webhostingu.

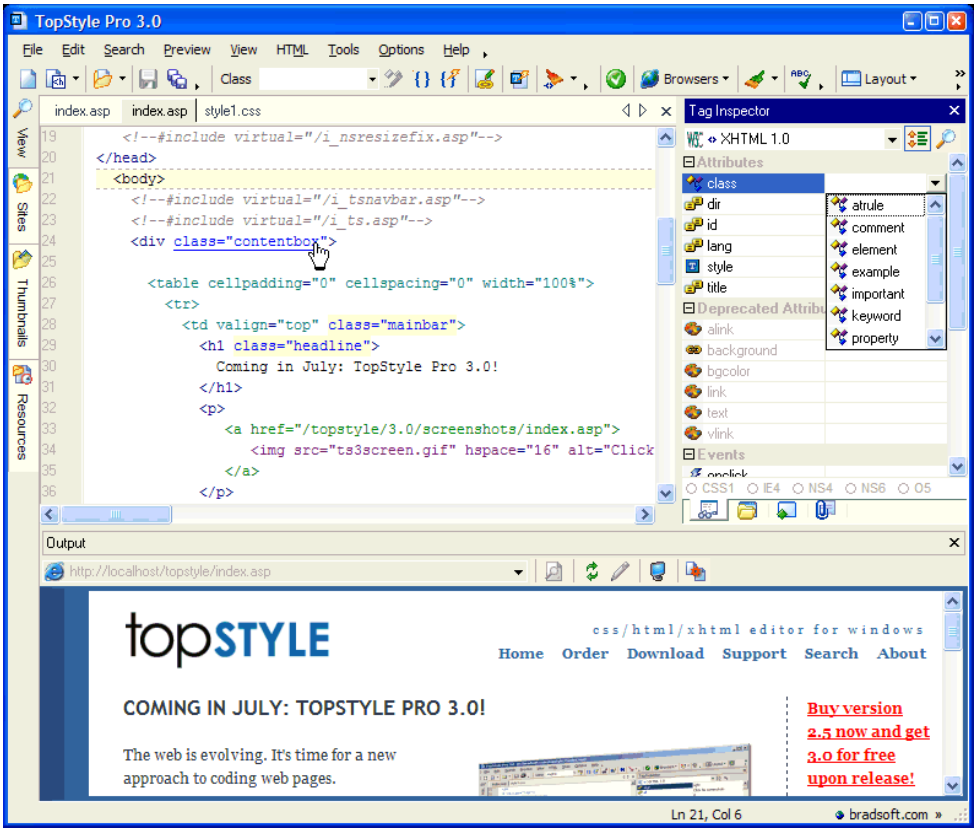
## 5.3 Nástroje použité při realizaci aplikace „Najdi si parťáka"

Webové aplikace se skládají z několika technologií, například pro prezentaci dat uživateli a tvorbu uživatelského rozhraní se nejčastěji používá značkovací jazyk (X)HTML, který je graficky stylován pomocí CSS stylů a obrázku. Řídící logika a jádro aplikace je naprogramováno v některém z programovacích jazyků (PHP, JSP, .NET), které podporují vývoj webových aplikací. Proto k vývoji webových aplikací je potřeba řada nástrojů, díky kterým je vývoj efektivnější, rychlejší a pohodlnější. V této podkapitole je přehled nástrojů, které jsem použil pro realizaci aplikace „Najdi si parťáka".

[18] Je to kolekce metod pro grafickou úpravu webových stránek. Ta zkratka znamená Cascading Style Sheets, česky "kaskádové styly".

### 5.3.1 TopStyle Pro verze 3.5

TopStyle Pro společnosti Bradbury Software je profesionální komerční editor kaskádových stylů pro Windows. Editor podporuje editaci HTML/XHTML dokumentů a CSS stylů, obsahuje integrované validátory HTML dokumentů a kaskádových stylů HTML Tidy a CSE HTML Validator. Dále nabízí kontrolu syntaxe CSS stylů ve více prohlížečích, generování zpráv o dokumentu a také celoobrazovkové náhledy prostřednictvím MS Internet Exploreru a Netscape Gecko (Mozilla Firefox aj.). Disponuje výběrem barvy z palet barev, CSS validátor a export kaskádových stylů přizpůsobený pro zvolený prohlížeč či CSS standard. I když TopStyle Pro editor je silným nástrojem, použil jsem ho pouze pro psaní CSS kódu.

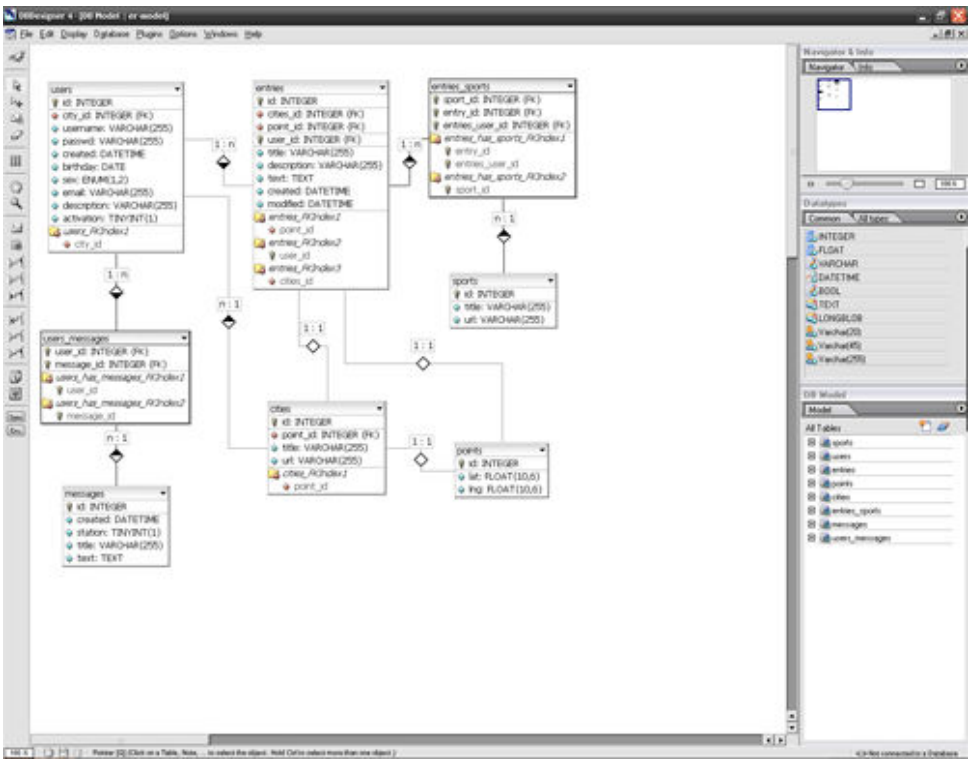



**Obrázek 17: Editor TopStyle Pro 3.5**

### 5.3.2 DBDesigner

Tento nástroj jsem využil pro návrh modelu databáze. DBDesigner je komplexní vizuální návrhové prostředí, které je určeno pro databáze MySQL a je

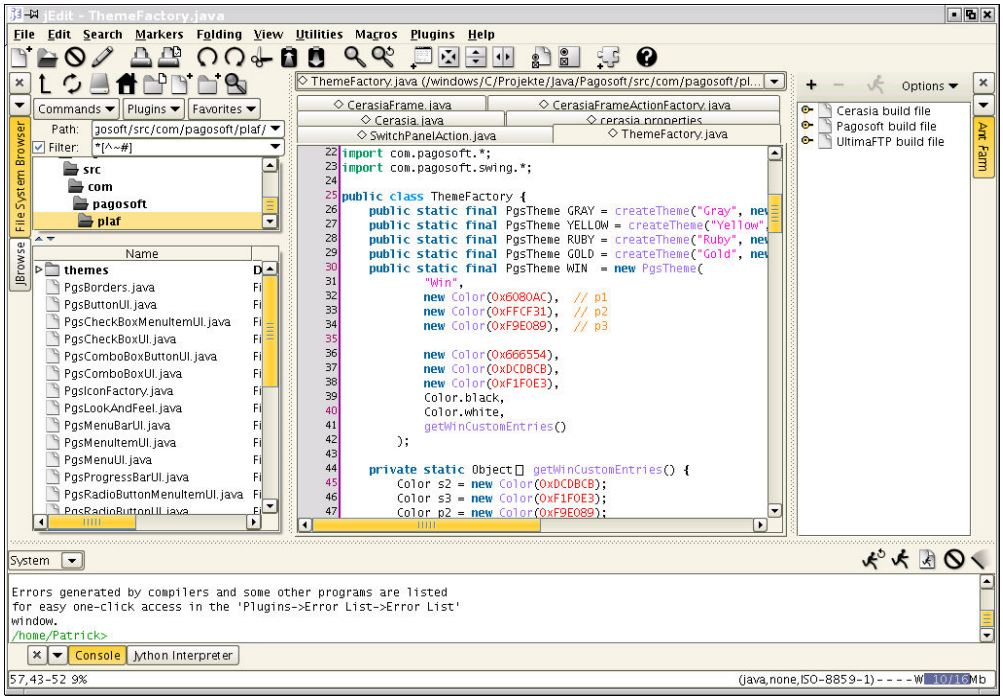
zcela zdarma. Podporuje všechny specifické vlastnosti databází MySQL a umožňuje jejich rychlý návrh a vizuální zobrazení. Uživateli poskytuje možnost vygenerovat SQL příkazy pro tvorbu tabulek z navrženého modelu. Jeho výhoda je přehledné a lehce ovládatelné uživatelské prostředí.



**Obrázek 18: DBDesigner**

### 5.3.3 jEdit verze 4.3

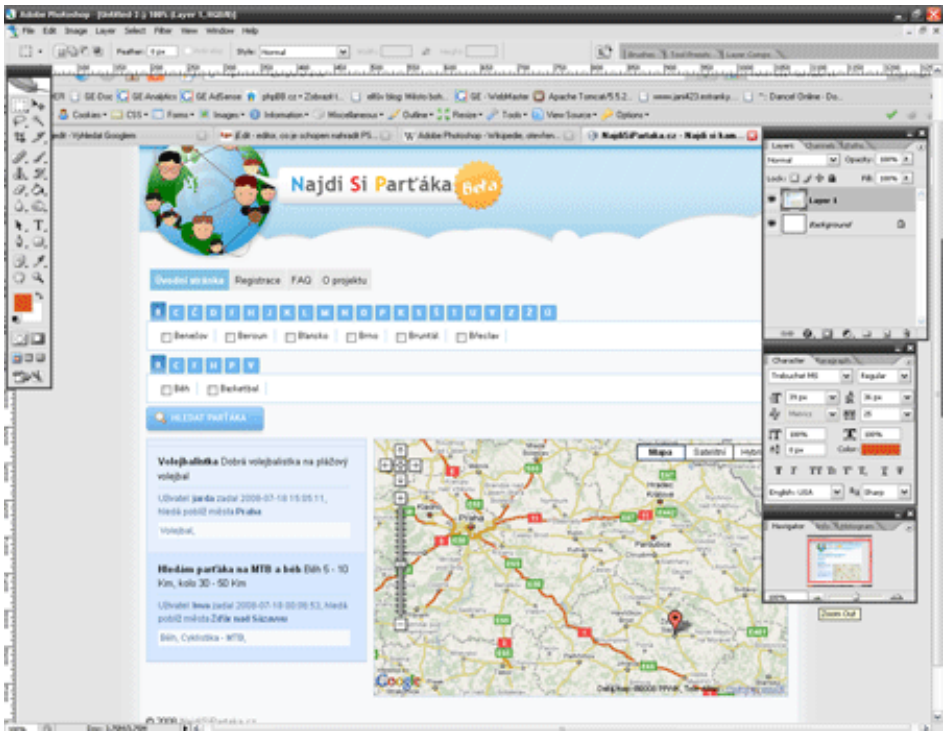
jEdit je textový editor dostupný jako open-source pod licencí GNU implementovaný v programovacím jazyce Java. Editor jsem použil při psaní veškerého zdrojového kódu aplikace a psaní HTML kódu pro prezentační vrstvu Podporuje zvýrazňování syntaxe pro zdrojové texty 130 jazyků a velké množství kódování (včetně UTF-8). Editor je multiplatformní, takže ho lze používat na operačních systémech MacOS X, OS/2, Unix, VMS a Windows. Jeho velkou výhodou je možnost rozsáhlé konfigurace, uživatel má k dispozici nastavit si klávesové zkratky a makra. Pro editor je dostupné rozsáhlé množství pluginů, díky kterým je lehce rozšiřitelný a konkuruje funkcemi komerčním editorům.




**Obrázek 19: jEdit**

### 5.3.4 Adobe Photoshop CS2

Tento editor jsem použil pro návrh grafického rozhraní aplikace. Adobe Photoshop je profesionální grafický editor především pro tvorbu a úpravy bitmapové grafiky. Využívá se pro návrh grafických rozhraní aplikací a webových stránek, ale také na úpravu fotografií. Disponuje možností práce ve vrstvách, několik druhů efektů, stylu štětců. Editor je rozšiřitelný paginami a dají se do něj nahrávat nové štětce, tapety jiné symboly.




**Obrázek 20: Adobe Photoshop CS2**

## 5.4 Adresářová struktura aplikace

Popis adresářové struktury aplikace „Najdi si parťáka“.

**/web_root**

**/app**

| | | |
|---|---|---|
| **/config** | - | konfigurační soubory aplikace |
| **/controllers** | - | kontroléry |
| **/locale** | - | lokalizační soubory pro počeštění aplikace |
| **/models** | - | modely |
| **/views** | - | šablony pro prezentační vrstvu |
| **/webroot** | - | root webové aplikace |
| **/css** | - | css styly |
| **/desing** | - | obrázky pro design |
| **/js** | - | javascriptové knihovny |
| **/cake** | - | jádro CakePHP (knihovny CakePHP) |

**index.php**

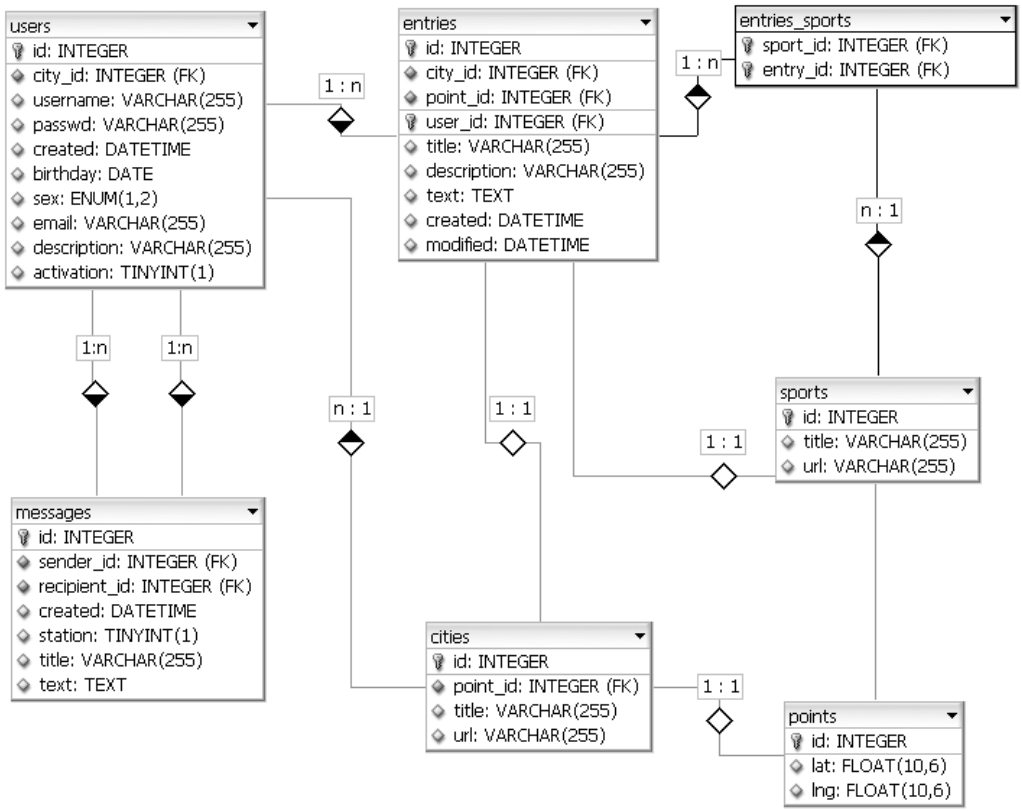
## 5.5 Návrh databáze

Návrh databáze se odvíjel od předpokládané funkcionality aplikace. Jelikož uživatelé mají možnost registrace a přidávání záznamu, je nutné záznamy o uživatelích a poptávek uživatelů ukládat do databáze. Ke správné funkčnosti aplikace je potřeba mít zdroj seznamu měst a souřadnice polohy, kde se města nacházejí, aby je bylo možné zobrazit uživateli na interaktivní mapě.

### 5.5.1 ER diagram databáze

Vzájemné vztahy tabulek a jejich atributy jsou znázorněny v ER diagramu databáze na obrázku 21. Při návrh modelu databáze je vycházeno z těchto požadavků:

- každý uživatel může mít vlastní účet s volbou města, ve kterém žije.
- Uživatel může odeslat neomezený počet zpráv jakémukoliv uživateli a současně může obdržet neomezený počet zpráv. U zpráv se zaznamenává stav, jestliže byla přečtená nebo doposud nepřečtená.
- Uživatel může přidat neomezený počet poptávek, u kterých je potřeba evidovat souřadnice místa kde chce sportovat, nejbližší město místu kde chce sporty provozovat a jaké sporty chce provozovat.
- Každé místo a město má svoje souřadnice, díky kterým mohou být vykresleny na interaktivní mapě.




**Obrázek 21: ER diagram databáze aplikace**

### 5.5.2 Popis tabulek databáze

Přehled tabulek databáze, jejich popis atributů a funkcionality:

- *users* – tabulka slouží k ukládání profilů uživatelů a jejich hesla pro přihlášení, atributy tabulky a jejich popis je v tabulce 2.

| **atribut** | **popis** |
|---|---|
| id | primárním klíčem[19] |
| city_id | cizí klíč[20]pro spojení s tabulkou cities |

[19] Primární klíč jednoznačně identifikující každý záznam v databázové tabulce.

[20] Cizí klíč je integritní omezení, které u tabulky vytvoří spojení jednoho nebo více jejích sloupců se sloupcem nebo sloupci jiné („cizí“) tabulky.

| | |
|---|---|
| username | uživatelské jméno, musí být unikátní |
| passwd | heslo zakódované pomocí MD5 algoritmu |
| created | datum vytvoření účtu |
| birthday | datum narození uživatele |
| sex | pohlaví uživatele |
| email | email uživatele, musí být unikátní a slouží pro přihlášení do administračního modu |
| description | popis uživatele, jeho záliby atd. |
| activation | stav aktivace uživatelského účtu, 1 – aktivní, 0 - neaktivní |

**Tabulka 2: Atributy tabulky users**

- *messages* – do tabulky se ukládají zprávy uživatelů. Obsahuje cizí klíč *sender_id* - identifikace odesílatele zprávy a cizí klíč c*recipient_id* - identifikace příjemce zprávy. Atributy tabulky a jejich popis je v tabulce 3

| **atribut** | **popis** |
|---|---|
| id | primárním klíčem |
| recipient_id | cizí klíč identifikující příjemce zprávy |
| sender_id | cizí klíč identifikující odesílatele zprávy |
| created | datum odeslání zprávy |
| station | stav zprávy – přečtena (1) nebo nepře. (0) |
| title | titulek zprávy |
| text | obsah zprávy |

**Tabulka 3: Atributy tabulky messages**

- *entries* – tabulka slouží k ukládání poptávek uživatelů, u každé poptávky musí být přiřazeno id nejbližšího města k místu, kde chce uživatel sportovat a přiřazen bod se souřadnicemi kde přesně chce sportovat, aby se záznam mohl zobrazit na interaktivní mapě uživateli. Atributy tabulky a jejich popis je v tabulce 4

| atribut | popis |
|---|---|
| id | primárním klíčem |
| city_id | cizí klíč pro spojení s tabulkou cities |
| point_id | cizí klíč pro spojení s tabulkou points |
| user_id | cizí klíč pro spojení s tabulkou users |
| title | titulek poptávky |
| description | krátký popis poptávky |
| text | delší popis poptávky |
| created | datum vytvoření poptávky |
| modified | datum editace poptávky |

**Tabulka 4: Atributy tabulky entries**

- *entries_sports* – tato tabulka je pomocná, slouží k realizaci relace typu M:N mezi poptávkou a sporty. Každá poptávka může mít neomezený počet poptávaných sportů a každý sport se může vyskytovat v n-počtu poptávkách. Atributy tabulky a jejich popis je v tabulce 5.

| atribut | popis |
|---|---|
| sport_id | cizí klíč pro spojení s tabulkou cities |
| entry_id | cizí klíč pro spojení s tabulkou cities |

**Tabulka 5: Atributy tabulky entries**

- *sports* – v této tabulce jsou uloženy sporty, které si uživatel může přiřadit ke své poptávce. Atributy tabulky a jejich popis je v tabulce 6.

| atribut | popis |
|---|---|
| id | primární klíč |
| title | název sportu |
| url | titulek bez diakritiky a mezer, používá se pro generování pěkných odkazů |

**Tabulka 6: Atributy tabulky sports**

- *cities* – v této tabulce jsou uložena města, které si uživatel může přiřadit ke své poptávce, každé město má cizí klíč *point_id* pomocí, kterého získává souřadnice uložené v tabulce *points*. Atributy tabulky a jejich popis je v tabulce 7.

| **atribut** | **popis** |
|---|---|
| id | primární klíč |
| point_id | cizí klíč pro spojení s tabulkou points |
| title | název sportu |
| url | titulek bez diakritiky a mezer, používá se pro generování pěkných odkazů |

**Tabulka 7: Atributy tabulky cities**

- *points* – do této tabulky se ukládají body, kromě primárního klíče *id* má tabulka dva atributy *lat* a *lng*. Atribut *lat* je pro zeměpisnou šířku a atribut *lng* pro zeměpisnou výšku. Záznamy z tabulky *points* jsou v relaci se záznamy z tabulek *entries* a *cities*, U měst se eviduje poloha a u poptávek uživatele se zaznamenává poloha, kde chce sportovat. Atributy tabulky a jejich popis je v tabulce 8.

| **atribut** | **popis** |
|---|---|
| id | primární klíč |
| lat | zeměpisná šířka |
| lng | zeměpisná výška |

**Tabulka 8: Atributy tabulky cities**

## 5.6 Vrstva s modely aplikace

Modely vychází z návrhu databáze aplikace a vazeb mezi tabulkami. Každý model odpovídá jedné tabulce v databázi. V adresáři *app/models* jsou uloženy tyto modely aplikace: *city.php*, *entry.php*, *message.php*, *point.php*, *sport.php*, *user.php*.

Modely jsou navrženy podle pravidel a postupu popsaného v kapitole kapitole 3.8. Ukázka zdrojových kódu jednotlivých modelů aplikace je na obrázcích 22 – 27.

```
<?php.
class City extends AppModel {.
    var $name = 'City';.
.
    var $belongsTo = array(.
            'Point' => array('className' => 'Point').
    );.
    var $hasMany = array(.
            'Entry' => array('className' => 'Entry'),
            'User' => array('className' => 'User').
    );.
}.
?>.
```

**Obrázek 22: Ukázka zdrojového kódu modelu City**

```
<?php.
class Entry extends AppModel {.
    var $name = 'Entry';.
.
    var $validate = array(.
        'title' => array(.
            'alphanumeric' => array(.
                'rule' => array('minLength', '15'),.
                'message' => 'Titulek musí obsahovat minimálně 15 znaků'
            ).
        ),.
        'description' => array(.
            'rule' => array('minLength', '30'),.
            'message' => 'Popis musí obsahovat minimálně 30 znaků'.
        ).
    );.
    var $belongsTo = array(.
            'City' => array('className' => 'City'),.
            'Point' => array('className' => 'Point'),.
            'User' => array('className' => 'User').
    );.
    .
    var $hasAndBelongsToMany = array(.
            'Sport' => array('className' => 'Sport',.
                        'joinTable' => 'entries_sports',.
                        'foreignKey' => 'entry_id',.
                        'associationForeignKey' => 'sport_id',.
                        'unique' => true.
            ).
    );.
}.
?>.
```

**Obrázek 23: Ukázka zdrojového kódu modelu Entry**

```
<?php.
class Message extends AppModel {.
    var $name = 'Message';.
.
    var $belongsTo = array(.
            'Sender' => array('className'  => 'User'),.
            'Recipient' => array('className' => 'User')
    );.
}.
?>.
```

**Obrázek 24: Ukázka zdrojového kódu modelu Message**

```
<?php.
class Point extends AppModel {.
    var $name = 'Point';.
.
    var $hasOne = array(.
            'City' => array('className' => 'City'),.
            'Entry' => array('className' => 'Entry').
    );.
}.
?>.
```

**Obrázek 25: Ukázka zdrojového kódu modelu Point**

```
<?php.
class Sport extends AppModel {.
    var $name = 'Sport';.
.
    var $hasAndBelongsToMany = array(.
            'Entry' => array('className' => 'Entry',.
                        'joinTable' => 'entries_sports',.
                        'foreignKey' => 'sport_id',.
                        'associationForeignKey' => 'entry_id',.
                        'unique' => true.
                    ).
    );.
}.
?>.
```

**Obrázek 26: Ukázka zdrojového kódu modelu Sport**

```
<?php.
class User extends AppModel {.
    var $name = 'User';.
.
    var $validate = array(.
            'email' => array(.
                'rule' => array('email', true),.
                'message' => 'Musíte zadat správný email!'.
            ).
        );.
        .
    var $belongsTo = array(.
            'City' => array('className' => 'City').
    );.
    var $hasMany = array(.
            'Entry' => array('className' => 'Entry'),.
            'SentMessage' => array( 'className' => 'Message'),.
            'ReceivedMessage' => array( 'className' => 'Message').
    );.
    .
    function validateLogin($data){.
        $user = $this->find(array('email' => $data['email'],.
            'passwd' => md5($data['password']), 'activation' => 1 ),
            array('id', 'username', 'activation'));.
        if(empty($user) == false).
            return $user['User'];.
        return false;.
    }   .
}.
?>.
```

**Obrázek 27: Ukázka zdrojového kódu modelu User**

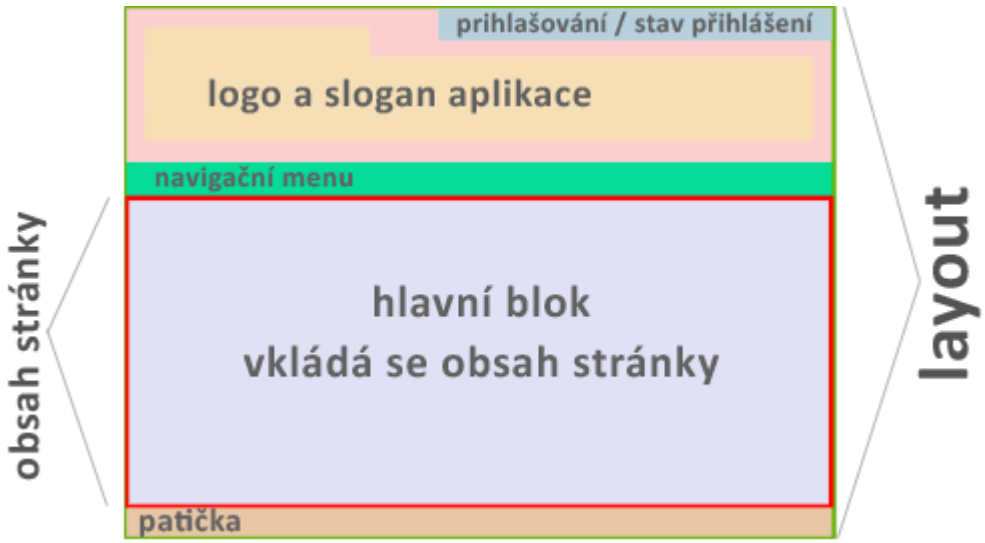
## 5.7 Prezentační vrstva a design aplikace

V této kapitole je popsána struktura webové aplikace a návrh designu aplikace.

### 5.7.1 Struktura webové aplikace

Webová aplikace generuje převážně HTML stránky, proto je implicitně nastaveno, aby se stránky vkládali do výchozího layoutu, ve kterém jsou nastaveny HTML hlavičky. Díky tomu mohou webové prohlížeče správně interpretovat

webovou aplikaci uživateli. Pro přehlednost aplikace jsem volil jednoduchý systém rozložení grafických prvků na stránce. Na horní části stránky je hlavička s informačním blokem o stavu přihlášení uživatele, pod ním logem se sloganem. Pod hlavičkou je jednoduché hlavní menu a pod ním je umístěn hlavní blok, do kterého se vláda obsah stránek, na spodní části stránky je patička. Návrh tohoto rozložení je vidět na obrázku 28.




**Obrázek 28: Rozložení stránky**

### 5.7.2 Design aplikace

Výsledný vzhled úvodní stránky aplikace je vidět na obrázku 29. Tohoto vzhledu bylo docíleno díky použití CSS stylů a obrázků. Snaha byla udělat jednoduchý vzhled aplikace, aby byla pro uživatele snadno ovladatelná a přehledná. Na úvodní stránce jsou filtry pro výběr města a typu sportů, pomocí nich si uživatel může nechat vypsat jen relevantní poptávky uživatelů. Pod filtrem se vypisují nabídky uživatelů a vpravo mapa, na které se zobrazují ikony jednotlivých nabídek. Po kliknutí na ikonu se nabídka červeně zvýrazní, aby si mohl uživatel přečíst její podrobnosti.


**Obrázek 29: Design aplikace**

## 5.8 *Vrstva s radiči – řízení logiky aplikace „Najdi si parťáka”*

V této kapitole je popsána jen vybraná část implementace řídící logiky aplikace. Popsána je autentizace, autorizace uživatelů a vyhledávání v databázi poptávek parťáků, podle zvolených kriterií uživatelem.

Jak již bylo popsáno v kapitole 3.9, o řízení logiky aplikace zajišťují řadiče neboli kontroléry. UML diagram tříd radičů aplikace „Najdi si parťáka” se nachází v příloze 1A.

### 5.8.1 Autentizace a autorizace uživatele

*Autentizace* je proces, při kterém se ověřuje, zda je uživatel nebo entita opravdu ten, za koho se vydává. K uživatelské autentizaci dochází ve webových

aplikacích obvykle při přihlašování zadáním uživatelského jména a hesla. Poté server zašle klientovi session[21] token, což je nejčastěji náhodně vygenerovaný identifikátor aktuální session. Ten si prohlížeč uchovává ve své paměti. Společně s každým dalším požadavkem pak prohlížeč zasílá i tento session token, čímž je prováděna autentizace entitní. (19).

Autentizaci uživatele v aplikaci provádí metoda *login* kontroleru *AdminsController*, tato metoda je zavolána jakmile uživatel odešle svůj e-mail a heslo pomocí přihlašovacího formuláře, jestliže uživatel tyto údaje odeslal, metoda zavolá funkci *validateLogin($this->data)*, která je nadefinovaná v modelu *User*. Tato funkce zjistí, vyskytuje-li se v tabulce *users* uživatel se zadaným e-mailem, heslem a aktivovaným účtem, jestliže ano, funkce vrací získané informace o uživateli v opačném případě hodnotu *false*. Jestliže návratová hodnota funkce *validateLogin* není rovná logické nule, metoda *login* vytvoří sessions, do které uloží informace o uživateli, tímto je uživatel úspěšně přihlášen a přesměrován na úvodní stránky aplikace. Ukázka zdrojového kódu metody *login* je na obrázku 30.

[21] Session představuje množinu proměnných, které dovolují uchovávat hodnoty, které jim byly nastaveny, po dobu připojení. (20)

```
function login().
{.
    $this->pageTitle = 'Přihlásit se';.
    if(empty($this->data) == false).
    {.
        $user = $this->User->validateLogin($this->data);.
        if($user != false).
        {.
            $this->Session->write('User', $user);.
            $this->Session->setFlash('Byl jste úspěšně přihlášen.');.
            $this->redirect('/');.
            exit();.
        }.
        else.
        {.
            $this->Session->setFlash('Bohužel nebyl jste přihlášen, z
            $this->redirect('/prihlasit-se');.
        }.
    }.
}.
```

**Obrázek 30: Zdrojový kód metody login pro autentizaci uživatele.**

*Autorizace* je ověření zda má uživatel dostatečná práva pro přístup ke konkrétním částem aplikace a datům.

O ověření je-li uživatel stále přihlášen do administrace, má možnost spravovat přijaté zprávy a poptávky se stará metoda *__validateLoginStatus()*, tato metoda je volána na začátku každé akce řadiče, která má být přístupná jen pro přihlášeného uživatele. Tato metoda kontroluje, jsou-li stále uložené informace o přihlášeném uživateli v sessions, jestliže sessions již vypršely nebo jsou prázdné metoda *__validateLoginStatus()* automaticky uživatele přesměruje na úvodní stránku aplikace kde se musí znovu přihlásit.

### 5.8.2 Vyhledávání v poptávkách podle zvolených kriterií

Na úvodní stránce aplikace se uživatelům vypisuje pět nejnovějších poptávek po parťácích. Tento výpis je proveden pomocí AJAXu, který automaticky při načtení úvodní stránky získá obsah stránky, která je na adrese *www.najdisipartaka.cz/entries/getall,* tento obsah se AJAXem vloží do bloku vedle mapy. Z požadované adresy stránky je patrné, že se zavolá řadič *EntriesController* a

jeho metoda *getAll*. Tato metoda získá z databáze pět nejnovějších poptávek a předá je prezentační vrstvě, kde se poptávky vypíší. Při získání poptávek z databáze je použita komponenta paginate[22], která se automaticky stará o stránkování požadovaných dat a v prezentační vrstvě poskytuje výpis odkazu pro stránkování. Uživatelé si mohou tedy listovat mezi poptávkami.

Uživatelé mají možnost si filtrovat výpis poptávek, tato filtrace se provádí pomocí volby města, ve kterém hledá pařtáka a jaké sporty chce provozovat. Jakmile uživatel zvolí požadované parametry a klikne na tlačítko hledat parťáka, AJAX na pozadí odešle formulář s parametry metodě *getEntries* kontroleru *EntriesController*. Tato metoda vyhodnotí předané parametry a získá z databáze poptávky, které odpovídají požadavkům klienta. Tyto poptávky předá prezentační vrstvě.

[22] Paginate je komponenta frameworku CakePHP 1.2 pro stránkování dat.

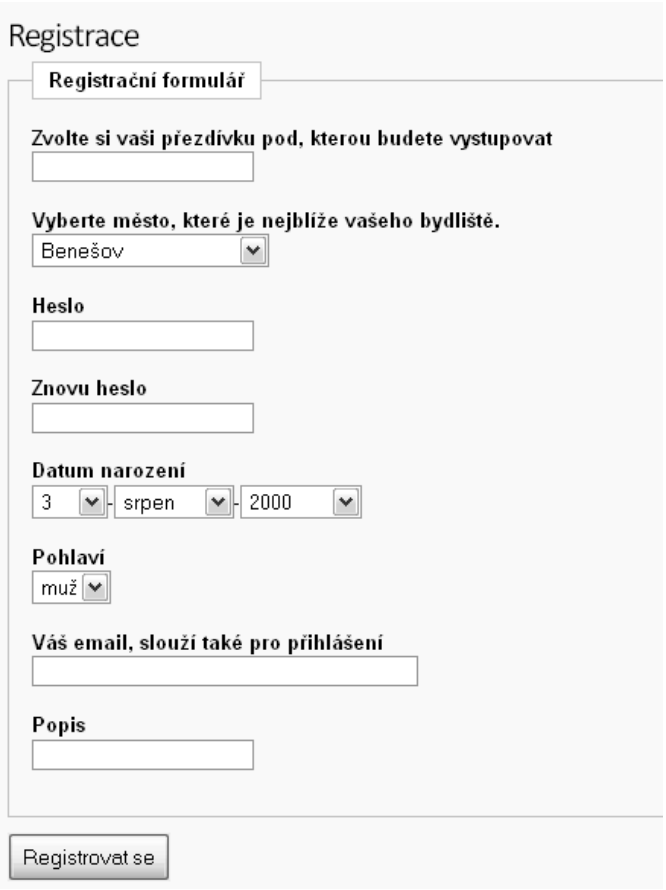
# 6 Základní obsluha webové aplikace „Najdi si parťáka” uživatelem

V této kapitole je popsán postup, jak si návštěvník může vytvořit vlastní účet, přihlásit se do administrace a přidat svoji poptávku. Přehled všech funkcí a použití aplikace je znázorněno v USE CASE diagramu v příloze 1B.

## *6.1 Vytvoření vlastního účtu a jeho aktivace*

Návštěvníci aplikace si mohou vytvořit vlastní účet, díky kterému získají možnost přidávat a spravovat svoje poptávky po parťácích a kontaktovat ostatní registrované uživatele.

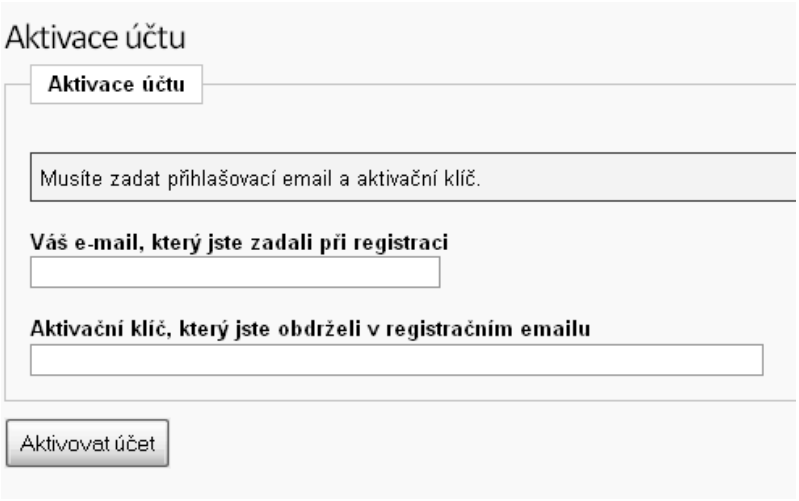
Formulář pro registraci uživatele se nachází na internetové adrese *www.najdisipartaka.cz/registrace* a jeho náhled je na obrázku 31.




**Obrázek 31: Formulář pro registraci.**

Při registraci návštěvník musí vyplnit přezdívku, pod kterou bude vystupovat, místo bydliště, datum narození, pohlaví, e-mail a heslo pomocí, kterých se bude přihlašovat. Po vyplnění povinných údajů a odeslání registračního formuláře, je uživateli odeslán informační e-mail, ve kterém jsou uvedeny přihlašovací údaje a aktivační klíč, pomocí kterého se aktivuje účet.

Formulář pro aktivaci účtu se nachází internetové adrese *www.najdisipartaka.cz/aktivace-uctu*, náhled formuláře je na obrázku 32. V tomto formuláři musí uživatel vyplnit e-mail, který si zvolil při registraci a aktivační klíč, který mu byl zaslán. Po úspěšné aktivaci účtu se již uživatel může přihlásit.




**Obrázek 32: Formuář pro aktivaci účtu.**

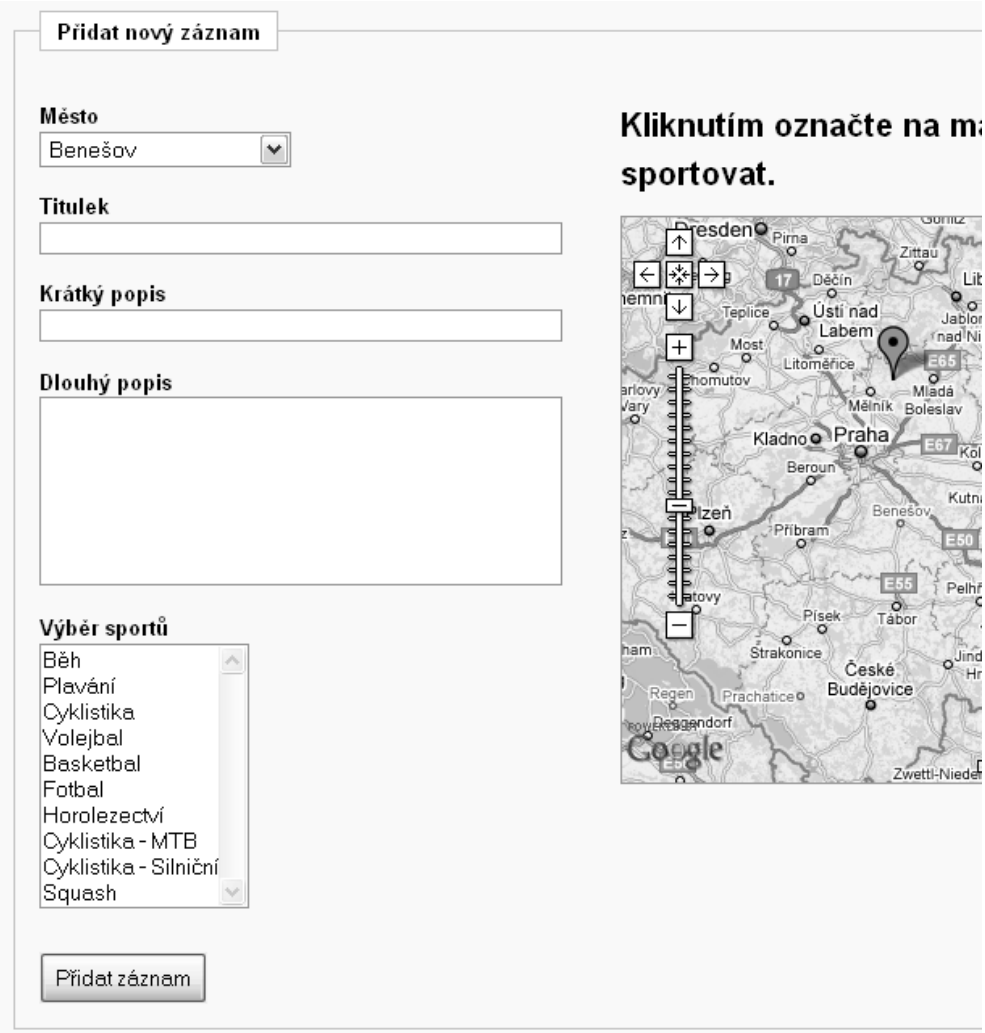
## 6.2 Přihlášení uživatele

Formulář pro přihlášení uživatele je umístěn v pravé části hlavičky stránky. Uživatel se přihlašuje pomocí e-mailu a hesla, které si zvolil při registraci, jakmile se uživatel úspěšně přihlásí, místo přihlašovacího formuláře se mu zobrazí jeho přezdívka, odkaz na jeho zprávy a odkaz pro odhlášení.

## 6.3 Přidání nové poptávky po parťákovi

Přihlášený uživatel má možnost přidávat a spravovat svoje poptávky po parťákovi. Formulář pro přidání nové poptávky je dostupný na adrese *http://www.najdisipartaka.cz/pridat-zaznam,* jeho náhled je na obrázku 33. Při zadávání nové poptávky, uživatel musí zvolit nejbližší město místu, kde hledá potenciálního parťáka, vyplnit titulek poptávky, který ji nejlépe charakterizuje a popsat poptávku v krátkém a delším popisu. Uživatel si může vybrat jaké sporty poptávka má zahrnovat muže si zvolit neomezený počet sportů, volba více sportů se provádí pomocí klávesy Ctrl a kliknutí myší a daný sport. Aby poptávka byla kompletní, uživatel musí kliknutím levého tlačítka myši na mapě zvolit místo, kde hledá potenciálního parťáka.

Jestliže uživatel vyplnil všechny povinné údaje a odeslal formulář, jeho poptávka je uložena do databáze a ostatní návštěvníci stránek si jí mohou najít na úvodní stránce aplikace, která je na adrese *www.najdisipartaka.cz*.




**Obrázek 33: Formulář pro přidání nové poptávky.**

# 7 Závěr

V teoretické části práce je popsán obecný postup tvorby webové aplikace postavené na využití frameworku CakePHP 1.2 a popsána implementace Google Maps API do webové aplikace.

Hlavním cílem práce bylo vytvořit webovou aplikaci „Najdi si parťáka", což se úspěšně podařilo. Aplikace je spuštěna na adrese *www.najdisipartaka.cz* a splňuje požadavky, které byly v zadání. Výsledná aplikace je zcela funkční a poskytuje následující funkce:

- pohodlné procházení, vyhledávání a zobrazování poptávek uživatelů,
- možnost registrace uživatelů,
- registrovaní uživatelé mohou kontaktovat ostatní uživatele pomocí zpráv,
- registrovaní uživatelé mohou spravovat přijaté zprav od uživatelů,
- registrovaní uživatelé mohou spravovat svoje poptávky po parťácích a vytvářet nové.

Do budoucna bych chtěl aplikaci rozšířit o další funkce, jako je možnost zobrazení profilu uživatelů a jejich poptávek po parťákovi na speciální adrese *www.najdisipartaka.cz/uzivatelske_jmeno*, aby mohli odkazovat na svoje poptávky z vlastních stránek a blogů. Dalším vylepšením by měla být možnost registrace obchodů se sortimentem pro sport, sportovních center a klubů. Aktuální verze aplikace neposkytuje administrátorovi možnost správy uživatelských účtů, poptávek uživatelů a veškerých dat potřebných pro chod aplikace. Administrace databáze se provádí pomocí webového správce phpMyAdmin, což není zrovna nejšťastnější řešení, proto v budoucnu se aplikace rozšíří o modul pro administrátora.

# Použité zdroje


1) *Framework* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Framework>.

2) *API* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/API>.

3) *CakePHP* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW:<http://cs.wikipedia.org/wiki/CakePHP>.

4) *MVC* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MVC>.

5) *Open Source software* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Open_source_software >.

6) *Object-relational mapping* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Object-relational_mapping>.

7) *Asynchronous JavaScript and XML* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML>.

8) *JavaScript* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Javascript>.

9) *I18n - Internacionalizace* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://kore.fi.muni.cz:5080/wiki/index.php/I18n_-_Internacionalizace>.

10) *PHP Frameworks* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://www.phpframeworks.com/index.php>.

11) ŠKRÁŠEK, Jan. *CakePHP - začínáme s frameworkem. Programujte.com* [online]. 2007 [cit. 2008-07-21]. Dostupný z WWW: <http://programujte.com/?akce=clanek&cl=2007061601-cakephp-zaciname-s-frameworkem>.

12) *Mod rewrite* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Mod_rewrite >.

13) *Cookbook CakePHP 1.2* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://book.cakephp.org/view/10/understanding-model-view-contr>.

14) *HyperText Markup Language* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/HyperText_Markup_Language>.

15) *Portable Document Format* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Pdf >.

16) *Extensible Markup Language* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Extensible_Markup_Language>.

17) *Co je služba Mapy Google?* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://maps.google.cz/support/bin/answer.py?answer=7060&topic=10778.

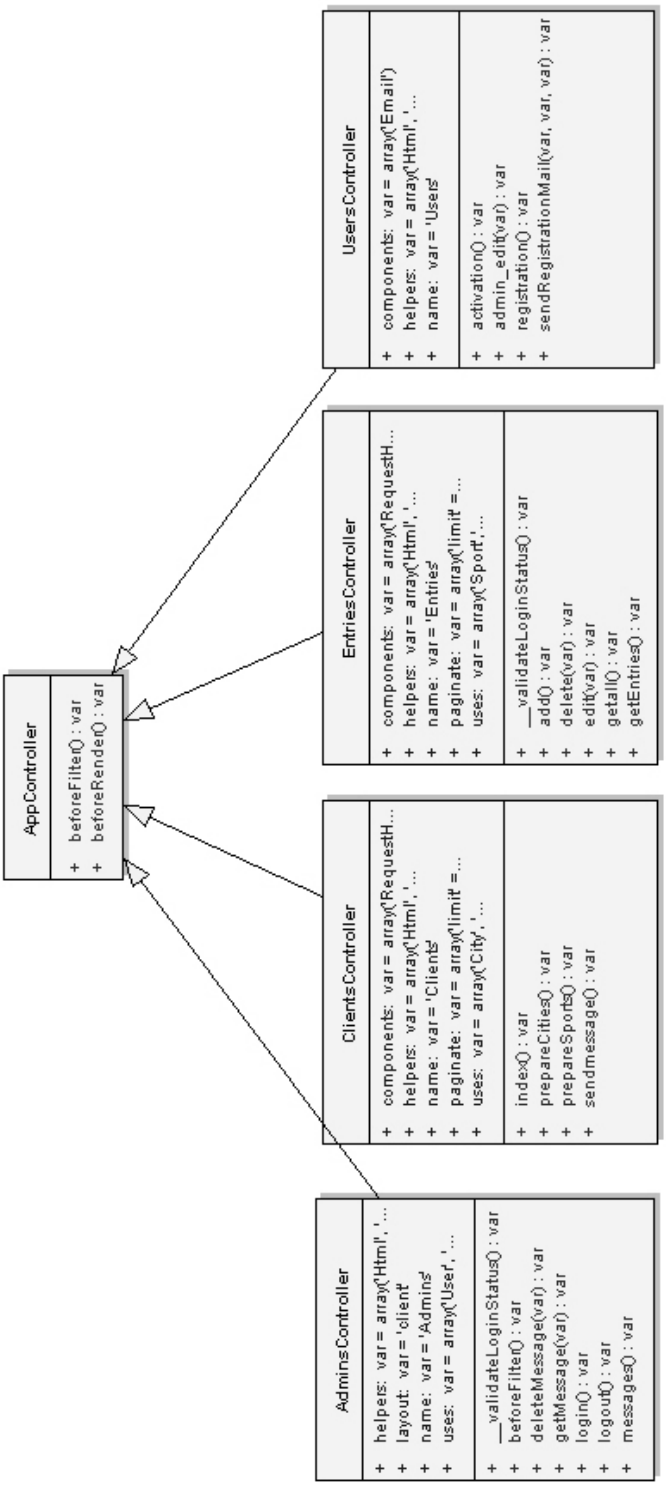
18) *Google Maps API* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://code.google.com/apis/maps>

19) TICHÝ, Jan. *Autentizace a správa uživatelů* [online]. 2007 [cit. 2008-07-21]. Dostupný z WWW: <http://www.jantichy.cz/diplomka/pozadavky/autentizace>.

20) *Session* [online]. 2008 [cit. 2008-07-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Session>.

# Přílohy

## *Příloha A – UML diagram tříd řadičů*



*Příloha B – USE CASE diagram používání aplikace*