České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Tomáš Skalický**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **SocioCars - sociální síť pro řidiče**

Pokyny pro vypracování:

Dokončete vývoj projektu SocioCars. Existující prototyp rozšiřte o funkcionalitu:

1. Zaznamenávání tras s možností trasu nebo její úseky komentovat.
2. Systém skupin uživatelů a sdílení příspěvků v těchto skupinách.
3. Zobrazení statistik vozidla a jízdy na konkrétní trase.
4. Podpora pro půjčování vozidel mezi uživateli a spolujízdu.
5. Porovnávání tras.

Dále proveďte úpravy v uživatelském rozhraní tak, aby bylo uživatelsky přívětivé a opravte chyby známé z prototypu. Veškeré výstupy práce vhodně otestujte.

Seznam odborné literatury:

1. Skalický, T. Salich, L.: Sociocars, ČVUT FEL projekt A7B36SI2, 2013
2. Skalický, T.: Sociocars, ČVUT FEL semestrální projekt, 2013
3. Nette Framework http://nette.org/
4. Google Maps API https://developers.google.com/maps/

Vedoucí: Ing. Ondřej Macek

Platnost zadání: do konce letního semestru 2014/2015

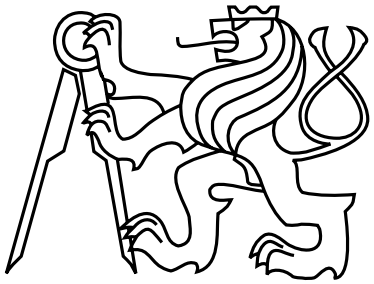
L.S.

doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 25. 2. 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

SocioCars - sociální síť pro řidiče

Tomáš Skalický

Vedoucí práce: Ing. Ondřej Macek

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

22. května 2014

Poděkování

Díky patří vedoucímu této práce Ing. Ondřeji Mackovi za jeho cenné rady a připomínky při vývoji a psaní této práce. Dále bych rád poděkoval své rodině a přátelům za jejich podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 18. 5. 2014 .....

Abstract

SocioCars is a social network for motorists, which allows its users to share their driving experience with other users. Apart from sharing common text messages, it also allows to view a map with recorded routes and work with these routes. The subject of this thesis is to continue developing the prototype of application SocioCars. This will be done mainly by adding new features for working with recorded routes. These features will include comparing users' routes, obtaining information on the progress of these routes and sharing them with other users. Furthermore, adding the option to assign friends into groups and share comments only in these groups, and the ability to share and borrow vehicles among users. Finally, the aim of this work is to also correct errors in existing prototype and its user interface.

Abstrakt

SocioCars je sociální síť se zaměřením na motoristy, která umožňuje svým uživatelům sdílet své zážitky z jízd s ostatními uživateli sítě. Mimo sdílení běžných textových zpráv umožňuje také zobrazit své zaznamenané ujeté trasy na mapě a dále s těmito trasami pracovat. Předmětem této bakalářské práce je pokračování ve vývoji prototypu aplikace SocioCars, především přidání nových funkcí pro práci s trasami, jako je porovnávání uživatelových tras, získání informací o průběhu těchto tras a jejich sdílení s ostatními uživateli. Dále pak přidání možnosti přiřadit přátele do skupin a sdílet příspěvky pouze v těchto skupinách a možnost sdílet a půjčovat vozidla mezi uživateli. V neposlední řadě je cílem této práce také opravit chyby v existujícím prototypu a v jeho uživatelském rozhraní.

Obsah

Seznam obrázků

Kapitola 1

Úvod

SocioCars je projekt, jejímž cílem je vytvořit sociální síť, která bude určena převážně pro motoristy. Tato síť by se měla točit převážně kolem informací získaných přímo z vozidla uživatele (pomocí mobilní aplikace nebo ODB2 [24] jednotky) a tato data zpracovávat pro účely informovanosti uživatele nebo pro sdílení s ostatními uživateli.

1.1 Cíle bakalářské práce

Cílem této bakalářské práce je pokračovat ve vývoji projektu SocioCars, opravit známé chyby prototypu, upravit uživatelské rozhraní, aby bylo jednotné a uživatelsky přívětivé, a rozšířit prototyp o následující funkcionality:

Zaznamenávání tras s možností trasu nebo její úseky komentovat
: Upravit stávající API pro přidání trasy, aby bylo možné přidat trasu i z jiných zařízení podporujících vytvořené API. Přidané trasy zobrazit na mapě i s možností zobrazit dlouhé trasy.

Systém skupin uživatelů a sdílení příspěvků v těchto skupinách
: Každý uživatel si bude moci vytvořit skupiny, do kterých bude řadit své přátele. Tyto skupiny budou soukromé a budou sloužit k tomu, aby uživatel mohl své příspěvky zobrazit pouze lidem z určité skupiny.

Zobrazení statistik vozidla a jízdy na konkrétní trase
: Upravit ukládané informace o vozidle tak, aby odpovídaly informacím uvedených v technickém průkazu vozidla. U vozidel dále zobrazit počet kilometrů, které vozidlo od doby registrace do systému ujelo. U tras zobrazit podrobné informace o jízdě.

Podpora pro půjčování vozidel mezi uživateli a spolujízdu
: Uživatelé budou moci svá vozidla půjčovat svým přátelům. Bude existovat několik způsobů půjčení: půjčení na určitý časový interval a možnost nastavit vozidlu pravidelné řidiče. Dále bude implementována funkce pro označení uživatele jako spolujezdce na určité trase.

Porovnávání tras

Uživatelé si budou moci u svých tras zobrazit trasy, které jsou podobné. Tyto trasy si poté mohou přidat do mapy pro bližší porovnání.

1.2 Historie projektu SocioCars

Projekt SocioCars vznikl v rámci předmětu A7B36SI2 - Řízení SW projektů[1], ve kterém jsem na něm pracoval společně s Lukášem Šalichem a dále jsem v práci (již sám) pokračoval v předmětu A7B36PRO - Semestrální projekt[2].

V rámci těchto předmětů vnikl prototyp aplikace. Tento prototyp umožňoval uživateli využívat základní prvky běžné sociální sítě, jako je například přidávání ostatních uživatelů do svých přátel, sdílení příspěvků na své zdi, zobrazení příspěvků za zdí ostatních uživatelů a vytváření zájmových uživatelských skupin s možností sdílení příspěvků v rámci těchto skupin. Mimo to umožnoval prototyp také nahrání ujeté trasy automobilu z palubní jednotky (získané pomocí API vytvořeného na základě zkušeností s palubními jednotkami z projektu Metrocars [9]) a zobrazení takto získané trasy na mapě.

1.3 Motivace pro práci na projektu

V projektu SocioCars jsem se rozhodl pokračovat i v rámci bakalářské práce z důvodu, že se jedná o zajímavý projekt, který slučuje princip sociálních sítí s evidencí stavu a pohybu vozidel. Výhodou samotné aplikace jsou velké možnosti využití a to nejen jako sociální sítě, ale po nějaké úpravě by mohla sloužit i firmám k evidenci stavu a pohybu jejich vozidel, včetně takových informací jako kdy a kde řidič tankoval, případně jak s automobilem (z hlediska využití motoru) zachází.

[1] <http://www.fel.cvut.cz/education/bk/predmety/13/95/p1395706.html>
[2] <http://www.fel.cvut.cz/education/bk/predmety/14/02/p1402506.html>

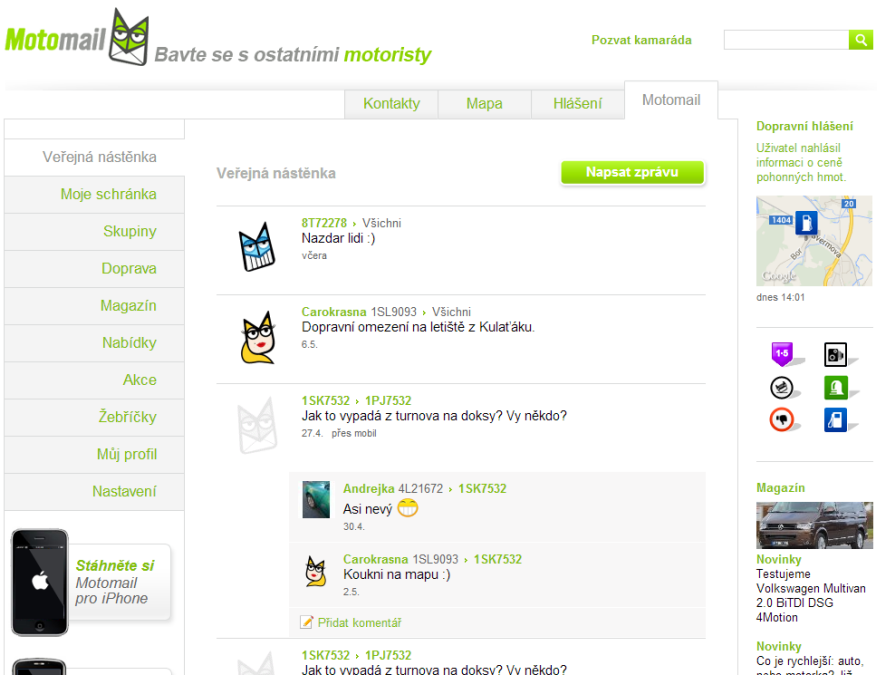
Kapitola 2

Rešerše existujících řešení

V této kapitole se zaměřím na již existující aplikace, které nabízejí podobnou funkčnost jako SocioCars, jak v Česku, tak v zahraničí. Tyto aplikace poté porovnám se samotným SocioCars.

2.1 Motomail

Služba Motomail [10] slouží ke komunikaci mezi motoristy. Umožňuje zveřejňování aktuálních dopravních situací pomocí mobilní aplikace, posílání zpráv mezi uživateli pomocí SPZ a sdílení informací o cenách pohonných hmot. Obsahuje také komunitní skupiny.



Obrázek 2.1: Motomail

Aplikace se hodně zaměřuje na podporu mobilních zařízení pomocí své mobilní aplikace. Oproti Sociocars se nesoustředí na uživatelské ujeté trasy a interakci mezi přáteli, ale spíše komunikaci mezi neznámými řidiči, kteří se navzájem informují o aktuální dopravní situaci nebo mohou například náhodného řidiče upozornit, pokud používá motomail, na rozbité světlo pomocí zprávy v rámci aplikace.

2.2 Bump

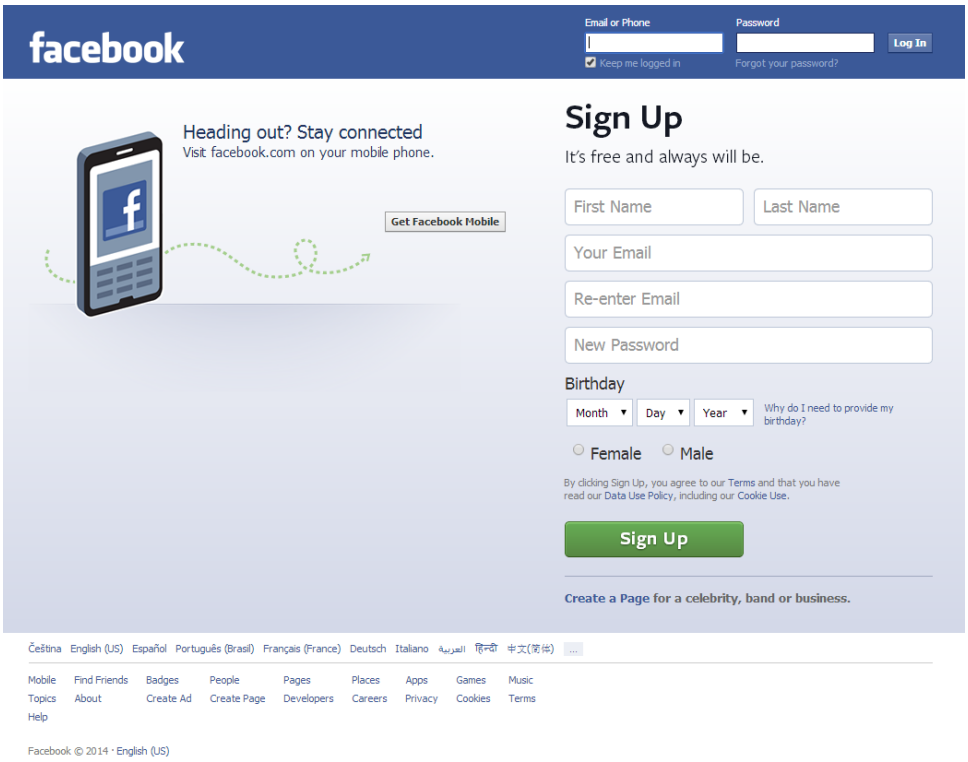
Bump [3] je americká služba umožňující komunikaci mezi řidiči pomocí jejich SPZ. Jedním z cílů této služby je možnost řidičů upozorňovat ostatní řidiče na viditelné poruchy na vozech (například rozbitá světla atd.). Bohužel se mi nepodařilo prozkoumat službu víc do hloubky, protože je dostupná pouze pro lidi žijící v US, vlastnící auto s americkou SPZ. Z článků o této aplikaci se však zdá, že tato Bump opravdu slouží pouze ke komunikaci mezi řidiči a další funkce nenabízí.

Obrázek 2.2: Bump

2.3 Facebook

Facebook [5] je sociální síť, která není specializována na vozidla, ale i přesto zde nějaké komunity okolo vozidel vznikají. Nativně sice nepodporuje vedení si databáze vozidel a in-

formací o jejich stavu, ovšem díky možnosti přidávat vlastní aplikace je možné tyto služby externě dodělat. Velkou výhodou této sociální sítě je obrovský počet uživatelů, kteří ji používají.



Obrázek 2.3: Facebook

2.4 Zhodnocení

Z vybraných služeb se všechny věnují převážně sociální stránce, ovšem o záznamy o ujetých trasách se již nestarají. Obě služby Motomail a Bump sprostředkovávají komunikační kanál mezi navzájem se neznajícími řidiči, což ovšem není cíl projektu SocioCars, který se spíše zaměřuje na komunikaci se svými přáteli. Možnost komunikace s ostatními řidiči na silnici přes SPZ je nicméně zajímavá a mohla by se implementovat při další rozšiřování tohoto projektu.

Kapitola 3

Analýza

V této kapitole se budu věnovat analýze projektu SocioCars. Budu vycházet z analýzy, kterou jsem vytvořil pro závěrečnou zprávu v předmětu A7B36PRO, ale zaměřím se převážně na nové části aplikace, které budu k stávajícímu systému přidávat v rámci této práce a na části, které se budou výrazně měnit.

3.1 Funkční požadavky

Funkční požadavky určují, co vše by měl systém umět. Všechny požadavky je možné vidět na obrázku 3.1, podrobnější popis ovšem uvedu pouze u požadavků, které jsou hlavní pro práce provedené v této bakalářské práci.

REQ039 Správa skupiny přátel

- **REQ040** Vytvoření skupiny přátel
  Uživatel může vytvořit novou skupinu přátel.
- **REQ041** Úprava skupiny přátel
  Uživatel může upravit existující skupinu přátel.
- **REQ042** Přidání přátel do skupiny
  Uživatel může do existující skupiny přidat některé ze svých přátel.
- **REQ043** Odstranění přátel ze skupiny
  Uživatel může z existující skupiny své přátele odstranit.

REQ011 Správa ujetých tras

- **REQ012** Přidání ujeté trasy
  Uživatel může, pomocí externí aplikace za použití API v aplikaci SocioCars, přidat ujetou trasu.
- **REQ013** Zobrazení ujeté trasy
  - **REQ044** Zobrazení detailů o ujeté trase
    Uživatel si může zobrazit detaily o ujeté trase - ujetá vzdálenost, čas cesty, maximální a průměrná rychlost.

REQ045 Zobrazení podobných ujetých tras
Uživateli se u každé trasy zobrazí seznam podobných tras s informacemi o nich.

REQ045 Zobrazení ujetých tras na mapě
Uživateli se zobrazí mapa s ujetou trasou s možností přidat do ní další podobné trasy pro jejich srovnání.

REQ048 Přidání spolujezdce
Uživatel může ke svým trasám přidat spolujezdce, který poté může trasu prohlížet a porovnávat jí se svými trasami.

REQ049 Správa půjčování vozidel

REQ050 Přidání pravidelného řidiče vozidla
Uživatel může některého ze svých přátel označit jako pravidelného řidiče svého vozidla. Takto označení uživatelé mohou poté využívat vozidlo bez omezení.

REQ051 Odstranění pravidelného řidiče vozidla
Uživatel může uživateli, který je označen jako pravidelný řidič jeho vozidla, toto označení zrušit.

REQ052 Půjčení vozidla uživateli na určitý časový interval
Uživatel může půjčit své vozidlo některému z přátel na určitou dobu ohraničenou datem půjčení a datem vrácení vozidla.

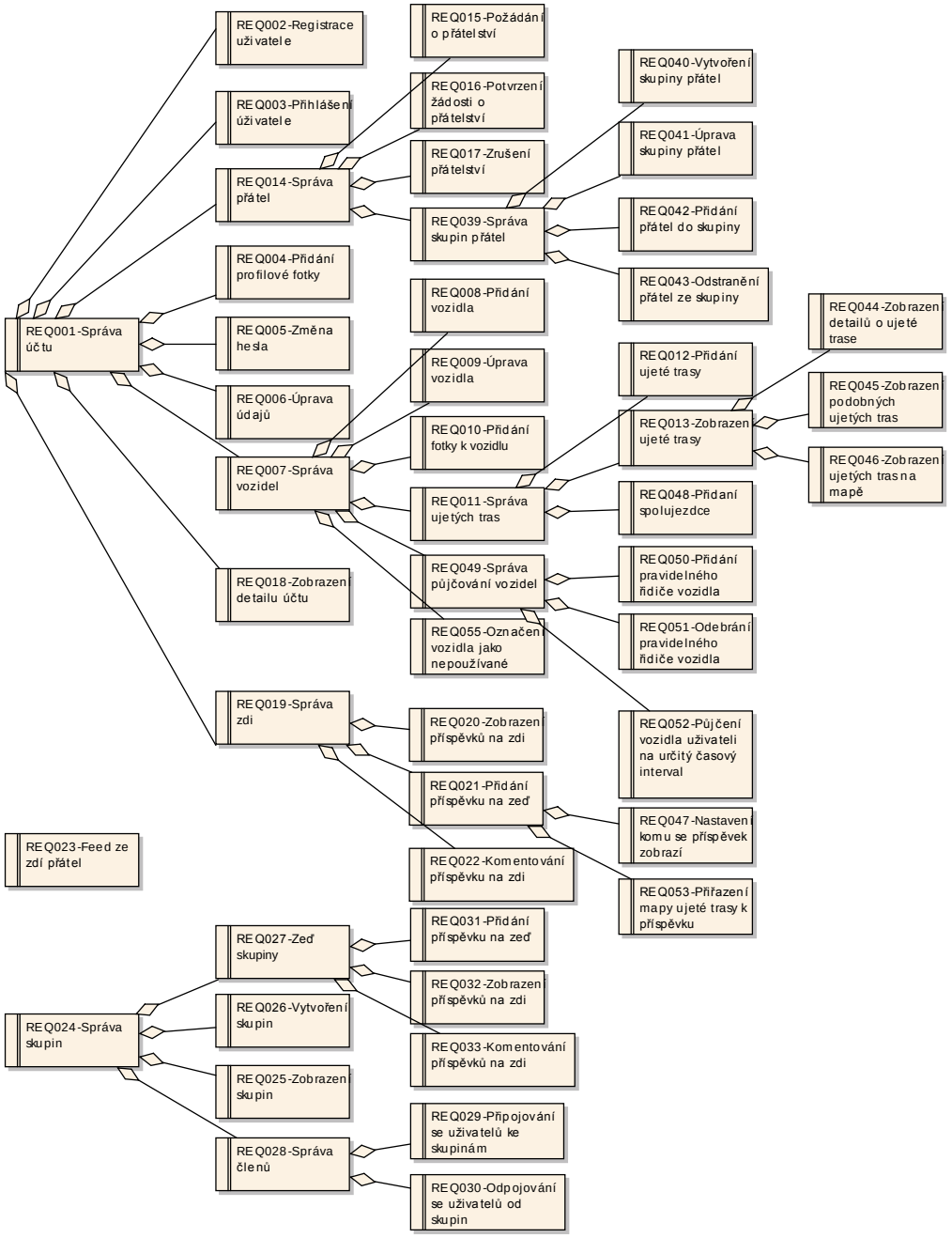
REQ019 Správa zdi

REQ021 Přidání příspěvku na zeď
Uživatel může přidat příspěvek na svou zeď.

REQ047 Nastavení komu se příspěvek zobrazí
Uživatel může zvolit, komu se jím přidaný příspěvek zobrazí. Zda všem, pouze přátelům nebo jen vybraným jím nadefinovaným skupinám přátel.

REQ053 Přiřazení mapy ujeté trasy k příspěvku
Uživatel může k příspěvku přiřadit mapu své ujeté trasy, která se poté zobrazí u příspěvku na zdi.

REQ023 Feed ze zdí přátel
Každý uživatel bude mít přístup ke stránce, kde budou vypsány všechny příspěvky ze zdí všech jeho přátel s možností komentovat tyto příspěvky. Tyto příspěvky budou filtrovány dle nastavení zobrazení těchto příspěvků.



Obrázek 3.1: Funkční požadavky

3.2 Nefunkční požadavky

Nefunkční požadavky jsou požadavky na samotné povedení, design a omezení aplikace.

SYS001 Webová aplikace
Systém bude fungovat ve formě webové aplikace.

SYS002 Jazyk PHP
K vývoji bude použit jazyk PHP.

SYS003 Databáze MySQL
Jako databázový systém bude využita MySQL databáze.

SYS004 API pro nahrání ujetých tras mobilní aplikací
Aplikace bude integrovat API pro nahrávání ujetých tras pomocí mobilního zařízení.

SYS005 API pro ovládání webové aplikace pomocí mobilní aplikace.
Aplikaci bude možné ovládat pomocí externí aplikace za pomoci API. Toto API bude implementováno postupně podle potřeb paralelně vyvíjené mobilní aplikace.

3.3 Případy užití

Případy užití popisují, jací uživatelé systém používají a také jak jej používají. Případy užití této aplikace jsem se rozhodl rozdělit na několik částí podle toho, čeho se týkají. Těmito částmi jsou:

- Správa přátel
- Správa skupin
- Správa vozidla
- Správa ujetých tras
- Správa účtu

Za účelem zachování přehlednosti dokumentace, v této kapitole popíši případy užití právě po těchto celcích. Samotné UML diagramy případů užití si poté můžete prohlédnout v příloze B.

3.3.1 Aktéři

Systém obsahuje celkem 3 aktéry:

Nepřihlášený uživatel
Návštěvník stránky, který není přihlášen.

Přihlášený uživatel
Návštěvník stránky, který je přihlášen.

Mobilní aplikace
Podporovaná mobilní aplikace, která přistupuje k webu přes implementované API.

3.3.2 Správa přátel

Přihlášení uživatelé mohou spravovat své přátele. Přátelé jsou ostatní uživatelé systému, které si konkrétní uživatel zvolil a přidal si je do seznamu svých přátel. Poté si může zobrazit feed ze všech zdí takovýchto uživatelů a příspěvky na těchto zdech může komentovat. Pro bližší určení s kým sdílet jaký příspěvek, si může uživatel vytvořit skupiny svých přátel a sdílet příspěvky pouze s určitými skupinami. Takto vytvořené skupiny jsou neveřejné a pouze pro potřeby uživatele.

3.3.3 Správa skupin

Přihlášení uživatelé mohou vytvářet zájmové skupiny nebo se do již vytvořených skupin hlásit. V těchto skupinám mohou poté sdílet příspěvky s ostatními členy těchto skupin a tyto příspěvky komentovat.

3.3.4 Správa vozidla

Přihlášení uživatelé si mohou ke svému účtu přiřadit vozidla, který nejprve v systému vytvoří. Vozidla poté mohou půjčovat ostatním uživatelům buď na určitý časový úsek, nebo tyto uživatele označit jako pravidelné řidiče. Uživatelé, kteří mají vozidlo půjčeno nebo jsou označení za pravidelného řidiče, mohou tato vozidla využívat, jako kdyby byly jejich, pouze mají omezené možnosti editace těchto vozidel. Vozidla lze také převést pod nového majitele nebo označit jako již nepoužívané.

3.3.5 Správa ujetých tras

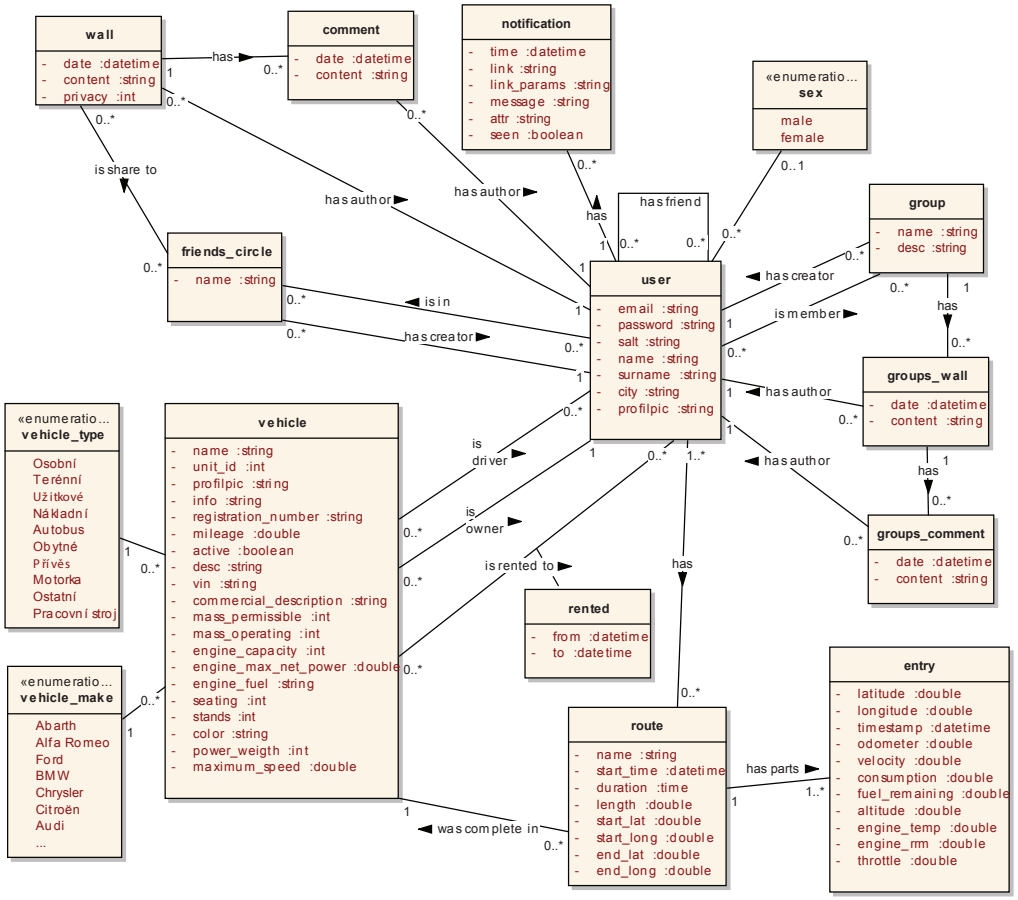
Uživatelé mohou nahrát pomocí mobilní aplikace svou ujetou trasu, kterou si poté mohou zobrazit ve svém profilu na mapě, sdílet ji na své zdi, kde ji mohou jeho přátelé komentovat, nebo ji porovnat s podobnými trasami, které již uživatel absolvoval a nahrál. Ke každé trase může uživatel určit spolujezdce, s kterými tuto trasu jel. Spolujezdci si poté mohou trasu prohlížet a sdílet stejně, jako by byla přímo jejich.

3.3.6 Správa účtu

Každý přihlášený uživatel má svůj profil, který může spravovat. Mimo editace údajů jako je jeho jméno, věk či heslo může uživatel psát příspěvky na svou zeď, vybírat s kým se má takový příspěvek sdílet (se všemi, pouze s přáteli nebo s vybranými skupinami přátel). Může také příspěvky na své zdi komentovat a reagovat tak na komentáře od ostatních uživatelů. Dále má uživatel možnost si přidat nebo upravit profilovou fotografii.

Nepřihlášený uživatel se může do aplikace registrovat nebo se přihlásit, pokud již je zaregistrovaný.

3.4 Doménový model



Obrázek 3.2: Doménový model

Na obrázku je vidět celý doménový model aplikace. V následující několika řádcích popíši vybrané důležité třídy:

user

Třída, která reprezentuje registrovaného uživatele. Nejdůležitější informace, které o nich uchovává, je jejich email a heslo, pomocí kterých se uživatelé mohou do systému přihlásit a jsou podle nich identifikováni. Ostatní atributy jsou již nepovinné a je pouze na uživateli, jestli je vyplní či nevyplní.

vehicle

Třída vehicle obsahuje informace o všech vozidlech, které jsou v systému využívané. Její atributy korespondují s informacemi uvedenými v technickém průkazu vozidla, navíc obsahují popisy vozidla, pomocí kterých bude identifikováno v systému.

route

Entita zastřešující informace o jedné trase. Mimo volitelného názvu trasy, která slouží k snadnějšímu rozpoznání trasy uživatelem, obsahuje informace o trvání, délce, začátku a konci trasy.

entry

Tato třída nese informace o jednotlivých bodech trasy. Mimo samotných souřadnic a času projetí těmito souřadnicemi obsahuje i další informace, jako je například rychlost, spotřeba paliva, otáčky motoru, teplota motoru a další. Tyto položky jsou nepovinné z důvodu, že ne každá jednotka zvládne poslat všechny tyto informace.

Kapitola 4

Návrh

4.1 Použité technologie

4.1.1 PHP 5.3

Jako programovací jazyk využiji PHP 5.3 [15]. PHP je programovací jazyk určený pro tvorbu webových stránek malého a středního rozsahu. Jedná se o nejpoužívanější jazyk pro tvorbu stránek tohoto rozsahu. Důvody, proč jsem vybral právě PHP, jsou snadná dostupnost, množství hostingových služeb, které nabízí hostingy pro aplikace založené na PHP, dobrá podpora objektů od verze 5 a v neposlední řadě to, že s tímto programovacím jazykem mám zkušenosti, čímž bych se měl vyhnout problémům při samotném programování a mohu se více soustředit na problémy, které se objeví pro konkrétní úkoly.

4.1.2 Nette Framework 2.1

Při vývoji použiji Nette Framework [13]. Jedná se o český framework pro tvorbu webových aplikací v PHP. Důvody pro volbu toho frameworku jsou především

- Kvalitní šablonovací systém Latte
- Skvělé debugovací nástroje
- Automatické zabezpečení aplikace proti hrozbám jako jsou XSS, CSFR a další
- Množství kvalitních doplňků, které pro framowork existuje
- Aktivní komunita uživatelů

Nette mimo jiné také podporuje následující návrhové vzory:

MVP
: Aplikace v Nette Frameworku se stává ze 3 vrstev: Modelu, view a presenteru. Model se stará o uchování a změnu dat, presenter získává data od modelu, které dále předává view, případně posílá modelu data v závislosti na vstupu uživatele. View se stará o zobrazení dat uživateli.

Dependency injection
: Jedná se o princip, při které neuvádíme závislosti tříd natvrdo uvnitř třídy a místo toho jim tyto závislosti předává někdo jiný pomocí konstruktoru, settrů, jiných metod nebo pomocí anotací (v této aplikace používám inject pomocí anotací v presenterech a pomocí inject metod v ostatních částech aplikace). O to jakou třídu kam injectnout se stará container v Nette Framework.

4.1.3 MySQL

Jako databázový systém využiji MySQL [12]. Důvod pro zvolení toho systému je, že je zdarma a pro potřeby SocioCars je zcela dostačující. Také rozšířenost mezi hostingy v kombinaci s PHP je větší než v případě ostatních vhodných databázových systémů.

4.1.4 Bootstrap 3

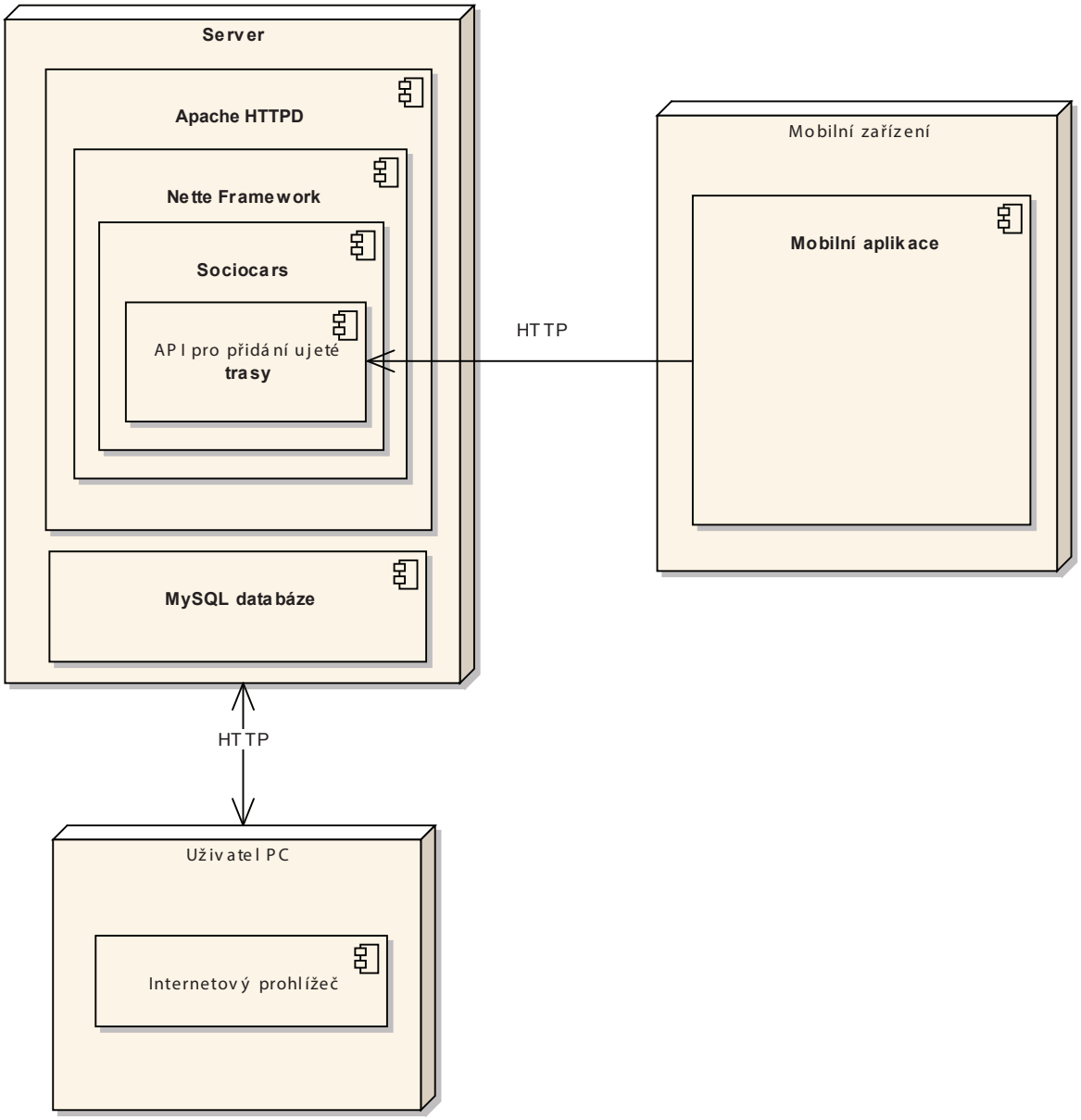
Bootstrap [2] je front-end framework pro tvorbu prezenční vrstvy webových aplikací. Obsahuje celou řadu prvků, jako jsou tabulky, formuláře, menu aj., které lze jednoduše implementovat do stránky. Obrovskou výhodou, pro kterou jsem Bootstrap zvolil, je, že mám jako tvůrce stránky jistotu, že jsou všechny prvky otestovány a že fungují ve všech nejpoužívanějších prohlížečích.

4.1.5 Google Maps

V aplikaci je potřeba zobrazovat ujeté trasy na mapě. K tomuto účelu jsem zvolil Google Maps Javascript API [7], které umožňuje zobrazit na stránce mapu od společnosti Google a poté s ní dále pracovat. Výhodou je dobře zpracovaná dokumentace a obrovské možnosti nastavení a využití. Další výhodou je podpora Google Maps všemi hlavními mobilními operačními systémy.

4.2 Diagram nasazení

Následující obrázek ukazuje, jakým způsobem bude webová aplikace komunikovat s klienty (mobilní aplikací nebo internetovým prohlížečem).



Obrázek 4.1: Diagram nasazení

Kapitola 5

Implementace

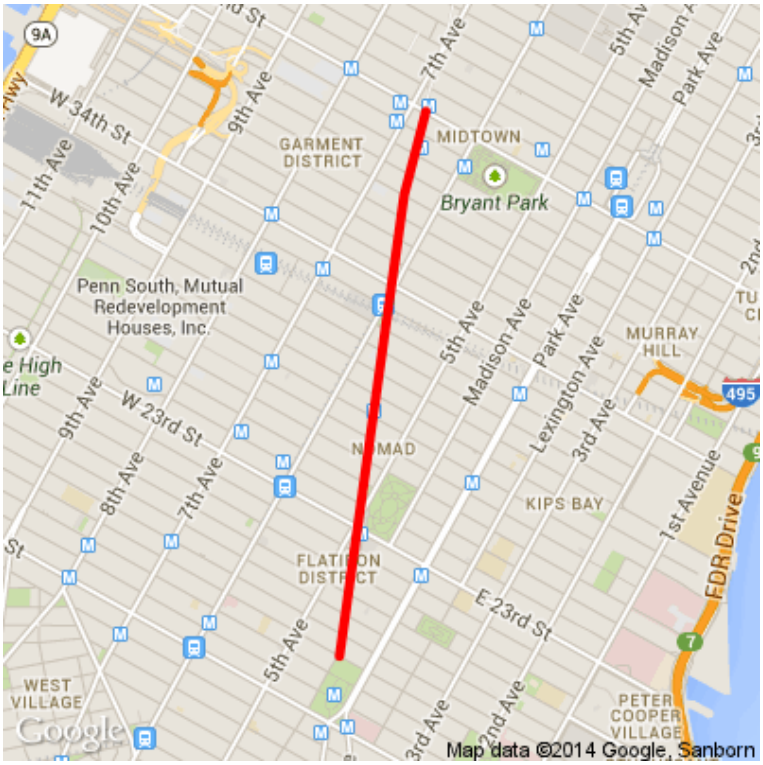
V této kapitole se budu zabývat samotnou implementací zadaných úkolů a pokusím se popsat vybrané nejzajímavější části, kterými jsou zobrazení ujetých tras na mapě 5.1 a porovnávání tras 5.2.

5.1 Zobrazení ujetých tras na mapě

V rámci aplikace se využívají mapy k zobrazení tras. Původní prototyp používal k této akci Google Static Maps API [18], které je velice jednoduché na užívání a pro zobrazení trasy na mapě stačí do stránky nalinkovat obrázek se správnými parametry v URL. Například

```
http://maps.googleapis.com/maps/api/staticmap?size=400x400&sensor=true
&path=color:0xff0000ff|weight:5|40.737102,-73.990318|
40.749825,-73.987963|40.752946,-73.987384|40.755823,-73.986397
```

zobrazí mapu jako je vidět na obrázku 5.1



Obrázek 5.1: Mapa vytvořená pomocí Google Static Maps API

Jak je vidět použití je velice jednoduché. V URL pouze uvedeme jak velká má mapka být, jakou má mít barvu a tloušťku čára zobrazující trasu a poté vyčteme souřadnice, kterými má trasa procházet. Bohužel jsem ale musel najít jiné řešení pro zobrazení mapy, protože použití Google Static Maps API přináší následující problémy:

Nemožnost přiblížení/oddálení mapky pro uživatele
: Toto je problém hlavně u dlouhých tras, kdy už je mapka tak oddálená, že nejde přečíst názvy ulic, vesnic případně i měst nebo naopak u hodně krátkých tras, kdy je mapka příliš přiblížená a uživateli může uniknout kontext.

Nemožnost zobrazení více tras na jedné mapě
: Tento problém se objevil teprve nyní z důvodu, že potřebuji na mapě zobrazit více tras pro jejich porovnávání, což Google Static Maps API neumožňuje.

Délka URL
: Vzhledem k tomu, že se souřadnice přenášejí v URL adrese, může tato adresa narůst do velké délky. Ačkoliv protokol HTTP délku URL nijak neomezuje [22], tak prohlížeče už nějaká omezení na délku URL aplikují. Například u prohlížeče Internet Explorer 8 [1], který je stále hojně používaný, je délka URL omezená na 2083 znaků [21] a většina ostatních prohlížečů má omezení podobné délky. Proto se může stát, že v případě dlouhé trasy bude adresa prohlížečem zkrácena, a nezobrazí se tedy celá trasa, nebo úplně ignorována.

Z možných API jsem zvolil opět API od Google a to konkrétně Google Maps Javascript API [7], které umožňuje pomocí javascriptu zobrazit na stránce interaktivní mapu z možností přiblížení/oddálení, posunu, přepnutí mapy na satelitní a také s možností zobrazit si Street View[1] určitého místa.

Pro přenos trasy jsem poté zvolil převod trasy na soubor formátu KML [8], což je formát, který se používá k zaznamenávání tras například v Google Earth [6]. KML má velice jednoduchou syntaxi a obsahuje informace nejenom o trase, ale i o tom, jakou barvu a velikost má čára označující trasu mít, název trasy, kde má být mapa vycentrovaná. Co pro mě ale bylo nejdůležitější je, že v jednom souboru může být uvedeno i několik tras a Google Maps Javascript API s tímto formátem umí pracovat pomocí vrstvy KML Layer.

Samotný javascript, pomocí kterého se mapa zobrazuje, není také komplikovaný. Kód, který je na to potřeba pro zobrazení mapky ze souboru na adrese http://147.32.80.205/data/routes/4.kml :

```
<script
  src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false">
</script>
<script>
function initialize(){
  var mapOptions = {
    zoom: 13,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  }

  var map = new google.maps.Map(document.getElementById('map-canvas'),
      mapOptions);
  var road = new google.maps.KmlLayer(
    "http:\/\/147.32.80.205\/data\/routes\/4.kml",
    { preserveViewport:true }
  );
  road.setMap(map);
  google.maps.event.addListenerOnce(road, "defaultviewport_changed",
  function() {
    recenter = road.getDefaultViewport().getCenter() ;
    map.panTo( recenter ) ;
  });
}
google.maps.event.addDomListener(window, 'load', initialize);
</script>
<div id="map-canvas" style="width: 800px; height: 450px"></div>
```

V kódu nejprve nalinkuji samotnou Google Maps Javascript API knihovnu a vytvořím funkci initialize, které pomocí DOM listeneru nastavím, že se má zavolat vždy při načtení stránky. V samotné metodě initialize poté načítám jednoduchým způsobem mapu: Nejprve vytvořím novou mapu pomocí volání google.maps.Map, které předám prvek, kam se má mapa

[1] Zobrazení panoramatického pohledu místa v rámci Google Maps

vykreslit a nastavení mapy. Následně vytvořím vrstvu KML pomocí google.maps.KmlLayer, které předám URL chtěného KML souboru a nastavení vrstvy. Následně vrstve KML nastavím mapu, v které se má zobrazit a nastavím jednorázový listener, který mapu vycentruje.

Pomocí takovéhoto volání API se zobrazí mapka s trasou podle definice v KML. Příklad takto vygenerované mapy s 3 trasami je vidět na obrázku 5.2.

Obrázek 5.2: Mapa vytvořená pomocí Google Maps Javascript API a souboru KML

Samozřejmě i toto řešení má své problémy, ovšem tyto problémy mají relativně snadné řešení:

Vytváření souborů KML a uchovávání na serveru

K zobrazení trasy je potřeba vygenerovat KML soubor, který je nutno uložit na server. Při velké návštěvnosti stránek a užívání zobrazení tras na mapách můžou takto vytvořené soubory zabrat hodně prostoru na disku serveru. Toto se dá vyřešit pravidelným mazáním těchto souborů například za použití cronu[2].

Omezení velikosti souboru KML

Google Maps Javascript API omezuje maximální velikost čteného souboru na 3MB čistého souboru KML [20]. K dosažení této velikosti ale dojde jen u hodně dlouhých cestu. Nejvyšší mnout testovaná velikost souboru KML je 50KB, který obsahoval informace o 3 trasách, každý o 800-1000 souřadnicových bodech. Tedy pro dosažení velikosti 3MB by musela trasa mít okolo 200 000 záznamů. V případě potřeby lze odesílat soubor

[2]Jedná se o automatický spouštěč úloh, který obsahují Unix-like operační systémy. Na operačních systémech Microsoft Windows [11] lze využít například Plánovač úloh

KML komprimovaný. Maximální velikost takového souboru je poté 3MB v komprimované podobě a až 10MB v podobě dekomprimované. Komprimování KML souborů jsem se v této práci ale nevěnoval.

5.2 Porovnávání tras

Pro porovnávání tras jsem nejdříve musel najít trasy podobné. Možností určit, jaké trasy jsou podobné, jsem měl celkem dvě:

- Podobné jsou trasy, procházející totožnou cestou.
- Podobné jsou trasy, které mají stejnou startovní a konečnou lokaci.

Nakonec jsem se rozhodl pro možnost, že se jedná o trasy se stejnou startovní a konečnou lokací. Důvodem je, že takto si může uživatel porovnat různé své cesty z jednoho místa na druhé a určit, která z nich je nejlepší, což je většinou užitečnější než porovnávání téměř totožných tras, kde rozdíly jsou způsobeny většinou dopravními zácpami, pracemi na silnici aj. Navíc toto řešení umožňuje porovnávat i totožné trasy.

K nalezení takovýchto tras jsem použil Haversine formula [23], což je rovnice, pomocí které se počítá vzdálenost mezi dvěma body z jejich zeměpisné délky a šířky. Její rovnice vypadá následovně:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{lat_2 - lat_1}{2}\right) + \cos(lat_1)\cos(lat_2)\sin^2\left(\frac{long_2 - long_1}{2}\right)}\right)$$

Kde d je vzdálenost bodů, r poloměr zeměkoule a lat_1, lat_2, $long_1$, $long_2$ jsou zeměpisné šířky/délky prvního a druhého bodu.

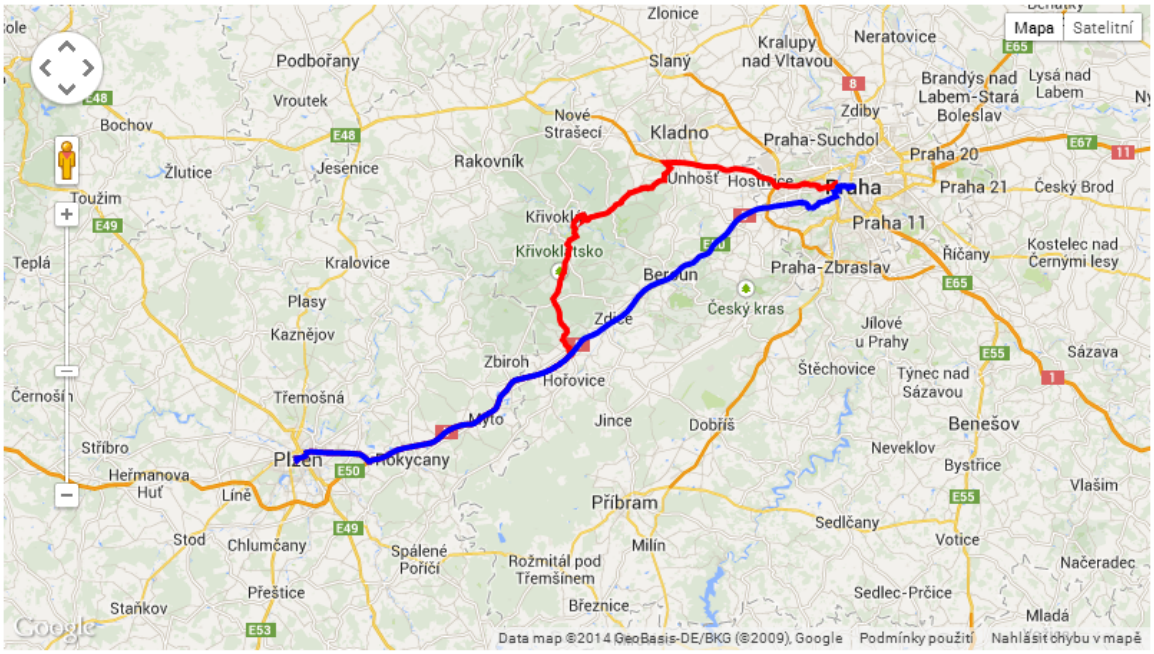
Pro potřeby aplikace bylo potřeba převést rovnici do MySQL a porovnat vzdálenost jak startovního, tak cílového bodu. Převod vypadá následovně [19]

```
SELECT *,
  (6367 * 2 * ASIN (
    SQRT (
      POWER(SIN((start_lat - route_start_lat)*pi()/180 / 2),2) +
      COS(start_lat * pi()/180) *
      COS(route_start_lat *pi()/180) *
      POWER(SIN((start_long - route_start_long) *pi()/180 / 2), 2)
    )
  )) as start_distance,
  (6367 * 2 * ASIN (
    SQRT (
      POWER(SIN((end_lat - route_end_lat)*pi()/180 / 2),2) +
      COS(end_lat * pi()/180) *
      COS(route_end_lat *pi()/180) *
      POWER(SIN((end_long - route_end_long) *pi()/180 / 2), 2)
    )
  )) as end_distance
FROM routes
WHERE id <> route_id
HAVING start_distance < 0.5 and end_distance < 0.5
```

Tento kód najde v tabulce routes všechny trasy, jejichž startovní a cílová pozice je maximálně 0.5km od startovní a cílové pozice trasy, ke které je tyto podobné trasy potřeba nalézt. Vysvětlení k částem kódu:

6367 Poloměr země v kilometrech
route_start_lat Zeměpisná šířka startovní pozice trasy, ke které hledáme trasy podobné
route_start_long Zeměpisná délka startovní pozice trasy, ke které hledáme trasy podobné
route_start_lat Zeměpisná šířka cílové pozice trasy, ke které hledáme trasy podobné
route_start_lat Zeměpisná délka cílové pozice trasy, ke které hledáme trasy podobné
route_id id trasy, ke které hledáme trasy podobné (zamezí, aby se našla ta samá trasa)
start_lat sloupec v databázi obsahující zeměpisnou šířku startovní pozice uložených tras
start_long sloupec v databázi obsahující zeměpisnou délku startovní pozice uložených tras
end_lat sloupec v databázi obsahující zeměpisnou šířku cílové pozice uložených tras
end_long sloupec v databázi obsahující zeměpisnou délku startovní pozice uložených tras
id sloupec v databázi obsahující id uložených tras

Takto vybrané trasy považuji za trasy podobné. Uživateli je vypíšu pod mapou původní trasy společně se základními informacemi, jako jsou délka, čas a datum (obrázek 5.3). Uživatel poté může Přidat na mapu tuto trasu přidat do mapky k trase původní. V případě, že toto provede, vygeneruji nový KML soubor a překreslím mapku viz 5.1



Vaše podobné trasy

ID	Datum	Délka	Trvání	
5	2014-04-20 15:16:32	115.2 km	03:42:45	Odebrat z mapy
6	2014-04-20 15:16:32	121.6 km	03:43:15	Přidat na mapu

Obrázek 5.3: Zobrazení podobných tras

Kapitola 6

Testování

V této kapitole se zaměřím na testování aplikace. Uvedu co, jak a proč je otestováno a jaké nástroje jsem k testování použil.

6.1 Unit testování

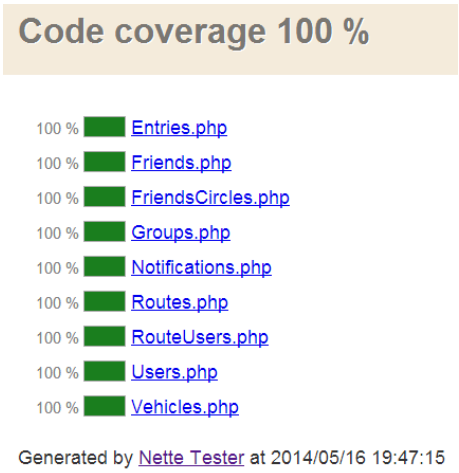
Modely aplikace obsahují business logiku aplikace, která je pravděpodobně nejnáchylnější na chyby. Může se jednat například o chyby při ukládání dat nebo čtení dat. Největší nebezpečí těchto chyb je, že některé z nich nemusí být při používání aplikace patrné a mohou vést k mystifikaci uživatele špatnými daty, k porušení dat v databázi nebo až k takovým bezpečnostním chybám jako je přihlášení špatného uživatele. Proto je potřeba detailně otestovat každou metodu v modelech, zda dělá přesně to, co se od ní očekává a vzhledem k častým změnám modelů je vhodné toto testování zautomatizovat, aby se dalo ověřit, zda změna jedné funkce v modelu nezpůsobila chybu funkce jiné. Právě z toho důvodu jsem zvolil unit testování.

Jako testovací nástroj jsem si vybral Nette Tester [14], který je součástí Nette Frameworku (nicméně lze použít i samostatně). Obrovskou výhodou tohoto nástroje, oproti konkurenci v podobě PHPUnit [16], je jednoduchost nastavení, propojení s Nette a také orientace v něm, díky tomu, že obsahuje pouze funkce, které jsou při běžném testování potřeba. Testy jsou rozděleny do několika tříd (běžně každá testuje jeden model), z nichž každá obsahuje konstruktor, v kterém je načten container aplikace, připojení k databázi, pokud je potřeba, a je také načten samotný testovaný model. Dále obsahuje metodu setUp, která se spouští před každým jednotlivým testem a slouží k nakonfigurování věcí, které jsou pro všechny testy v rámci jedné třídy společné (například vymazání obsahu databáze a její naplnění testovacími daty nebo třeba definování zámků, zamezující paralelnímu chodu s testy se stejným zámkem) a pak samotné testovací metody.

Pro mé potřeby jsem využil možnost Nette Testeru nastavit pro testování jinou databázi (jiný název databáze) než je využit pro samotnou aplikaci. Tato testovací databáze je kopie běžné databáze, z které jsou odstraněny všechna data a je následně (v setUp metodě testu) naplněna testovacími daty pro daný test. Díky tomu mohu provádět testování i za běhu aplikace a nemusím se bát nechtěné změny ostré databáze. Otestována je jednotlivě každá metoda modelu, výjimku tvoří funkce pro přidávání a odebírání přátel v modelu Friends,

které jsou, vzhledem ke své komplexnosti a provázanosti, otestovány najednou procesním testem.

Cílem bylo pokrýt modely testy pokud možno 100%, což se také podařilo (obrazek 6.1).



Obrázek 6.1: Pokrytí modelů

6.2 Manuální testování

Správnou komunikaci mezi presentery, modely a šablonami jsem otestoval manuálně a to tak, že jsem procházel jednotlivé případy užití a zkoušel, zdali fungují správně. Tyto testy jsem se rozhodl prozatím nezautomatizovat (například pomocí nástroje Selenium [17]) z důvodů, že je aplikace stále ještě ve vývoji a neustále se mění, proto by údržba takovýchto automatizovaných testů byla značně náročná a myslím si, že v této fázi i zbytečná. Testy odhalily několik chyb (hlavně při prohlížení stránky nepřihlášeným uživatelem), které jsem ihned opravil.

Kapitola 7

Závěr

V rámci bakalářské práce jsem pokračoval ve vývoji prototypu, který jsem dle zadání rozšířil o zadané funkce a následně jsem celou práci otestoval.

Nejvíce pozornosti a času bylo při vývoji vynaloženo na práci se skupinami uživatelů a na práci s mapou, včetně možnosti uživatelů své zaznamenané trasy porovnávat. Obě tyto funkcionality považuji za hlavní části aplikace vyvíjené v rámci této práce a zároveň za nejpřínosnější funkce pro samotný systém. I když v prototypu již možnost zobrazit trasu na mapě byla, bylo nutné jí předělat od základů, z důvodů nevyhovující zobrazené mapy a nemožnosti zobrazit na jedné mapě více tras. Práce s mapou i práci se skupinami uživatelů se mi povedlo implementovat, tak jak jsem si původně představoval a jsem s výsledkem spokojený.

Mezi další změny, které bylo nutné udělat, patří vytvoření API, které umožňuje ovládání velké části webové aplikace pomocí mobilní aplikace, kterou paralelně vyvíjí Tomáš Nosek. Toto API je implementováno postupně podle potřeby samotné mobilní aplikace, proto během vývoje v rámci této práce bylo implementováno pouze z nutné části.

Ostatní změny již mnoho zádrhelů nepřinesly a s jejich implementací jsem spokojen. Více pozornosti by si určitě zasloužila úprava uživatelského rozhraní, kde jsem se převážně snažil o sjednocení designu celého webu, což se mi podařilo, nicméně je na GUI stále mnoho práce.

S výběrem použitých technologií jsem celkem spokojený. Využití Nette Frameworku se ukázalo jak dobré rozhodnutí, převážně proto, že jsem s ním měl již zkušenosti z předcházejících projektů, díky čemuž jsem eliminovat problémy při samotném psaní aplikace na minimum. Přesto že jsem v Nette již před tímto projektem dělal, seznámil jsem se s ním během vývoje této práce více do detailů, zvláště pak s verzí 2.1, která přinesla spoustu nových změn, o kterých jsem nevěděl. Volba Nette Database jako prostředku pro komunikaci s databází nebyla špatná volba, nicméně vzhledem k velikosti aplikace, kterou jsem na začátku vývoje neodhadl, je udržování modelů, bez užití nějakého ORM frameworku, náročnější a méně přehledné než by být mohlo. Proto bych se příště rozhodl spíše pro využití ORM jako je například Doctrine 2 ORM [4].

Za velký přínos pro sebe považuji seznámení se s Google Maps Javascript API, které, jak se ukázalo, je mocný nástroj, který obsahuje mnoho funkcí pro práci s mapou a jeho ovládání je velmi jednoduché. V této aplikaci jsem jej využil pouze k zobrazení ujeté trasy s využitím souboru ve formátu KML (který se ukázal pro tuto aplikaci jako velice vhodný), nicméně jsem si všímal i dalších funkcí (například hledání tras mezi dvěma místy) a věřím, že je někdy v budoucnu při vývoji jiných aplikací využiji.

Literatura

[1] Internet Explorer 8.
http://www.microsoft.com/en-us/download/internet-explorer-8-details.aspx, stav z 8. 5. 2014.

[2] Bootstrap.
http://getbootstrap.com/, stav z 6. 5. 2014.

[3] BUMP.
https://www.bump.com/, stav z 8. 5. 2014.

[4] Doctrine - PHP Database Libraries.
http://www.doctrine-project.org/projects/orm.html, stav z 17. 5. 2014.

[5] Facebook.
https://www.facebook.com, stav z 8. 5. 2014.

[6] Google Earth, .
http://www.google.com/earth/, stav z 6. 5. 2014.

[7] Google Maps JavaScript API v3 – Google Developers, .
https://developers.google.com/maps/documentation/javascript/tutorial, stav z 6. 5. 2014.

[8] Keyhole Markup Language – Google Developers.
https://developers.google.com/kml/, stav z 6. 5. 2014.

[9] Metrocar Jan Wagner.
https://www.assembla.com/spaces/wagnejan_metrocar/wiki, stav z 2. 2. 2014.

[10] Motomail.cz - bavte se s ostatními motoristy.
http://motomail.cz/, stav z 8. 5. 2014.

[11] Microsoft Windows.
http://windows.microsoft.com/, stav z 8. 5. 2014.

[12] MySQL :: The world's most popular open source database.
http://www.mysql.com/, stav z 2. 2. 2014.

[13] Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework, .
http://nette.org/, stav z 5. 5. 2014.

[14] Nette Tester – pohodové testování, .
`http://tester.nette.org/`, stav z 16. 5. 2014.

[15] PHP: Hypertext Preprocessor, .
`http://www.php.com/`, stav z 2. 2. 2014.

[16] PHPUnit – The PHP Testing Framework, .
`http://phpunit.de/`, stav z 16. 5. 2014.

[17] Selenium – Web Browser Automation.
`http://seleniumhq.org/`, stav z 16. 5. 2014.

[18] Static Maps API V2 Developer Guide - Google Maps Image APIs – Google Developers.
`https://developers.google.com/maps/documentation/staticmaps/`, stav z 6. 5. 2014.

[19] Alexander Rubin. *Geo/Spatial Search with MySQL* [online]. 2008. [cit. 8. 5. 2014]. Dostupné z: <http://www.scribd.com/doc/2569355/Geo-Distance-Search-with-MySQL>.

[20] Google. *KML Support in Google Maps - Keyhole Markup Language - Google Developers* [online]. 2013. [cit. 9. 5. 2014]. Dostupné z: <https://developers.google.com/kml/documentation/mapsSupport>.

[21] Microsoft. *Adresa URL může být v aplikaci Internet Explorer dlouhá maximálně 2 083 znaků* [online]. [cit. 8. 5. 2014]. Dostupné z: <http://support.microsoft.com/kb/208427>.

[22] Network Working Group. *Hypertext Transfer Protocol – HTTP/1.1* [online]. 1999. [cit. 8. 5. 2014]. Dostupné z: <http://www.ietf.org/rfc/rfc2616.txt>.

[23] Přispěvatelé Wikipedie. *Haversine formula* [online]. 2014. [cit. 9. 5. 2014]. Dostupné z: <http://en.wikipedia.org/wiki/Haversine_formula>.

[24] Přispěvatelé Wikipedie. *On-board diagnostics* [online]. 2014. [cit. 9. 5. 2014]. Dostupné z: <http://en.wikipedia.org/wiki/On-board_diagnostics>.

Příloha A

Seznam použitých zkratek

API Application programming interface

CSFR Cross-site request forgery

DOM Document Object Model

GUI Graphical user interface

KML Keyhole Markup Language

ODB On-board diagnostics

ORM Object-relational mapping

PHP PHP: Hypertext Preprocessor

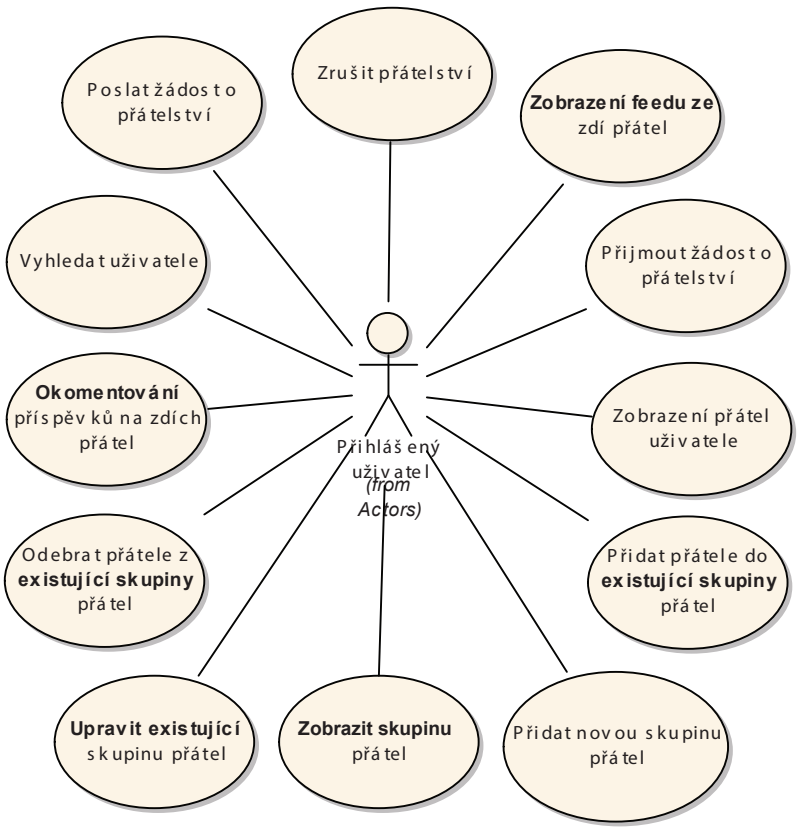
URL Uniform resource locator

XSS Cross-site scripting

Příloha B

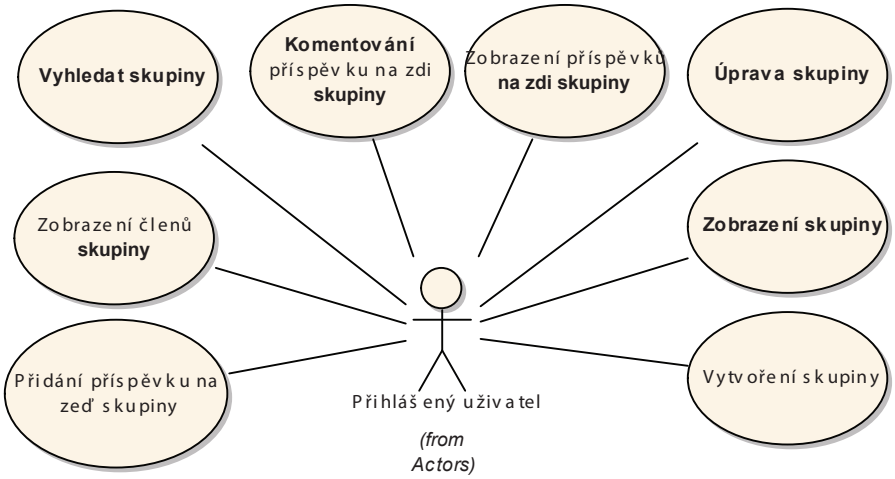
Diagramy užití

B.1 Správa přátel



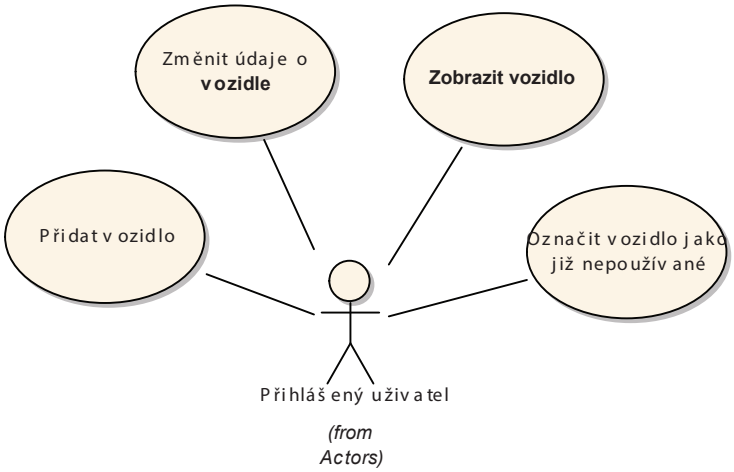
Obrázek B.1: Správa přátel

B.2 Správa skupin



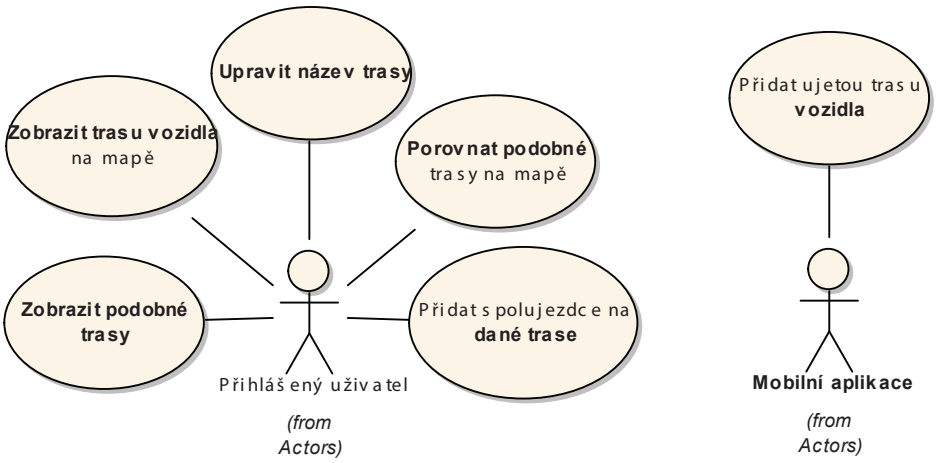
Obrázek B.2: Správa skupin

B.3 Správa vozidla



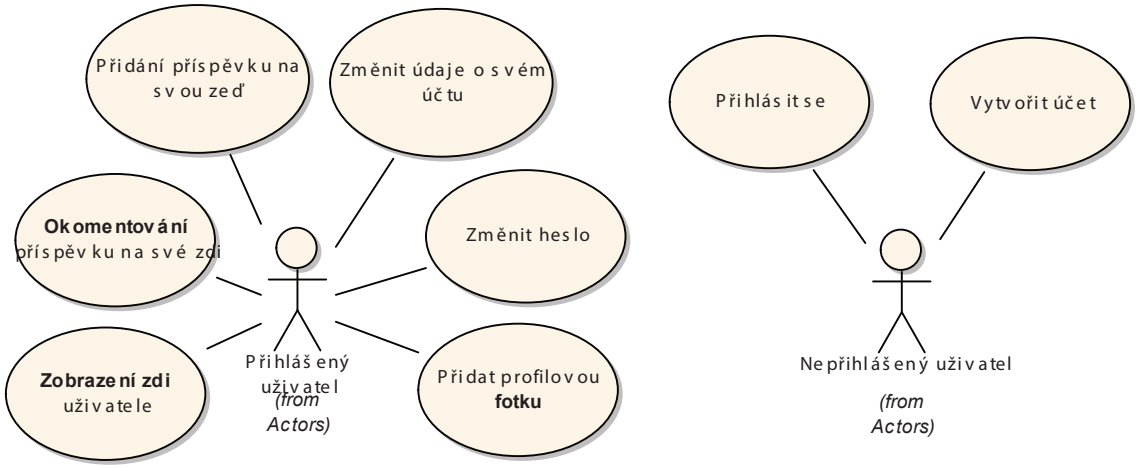
Obrázek B.3: Správa vozidel

B.4 Správa ujetých tras



Obrázek B.4: Správa ujetých tras

B.5 Správa účtu



Obrázek B.5: Správa účtu

Příloha C

Obsah přiloženého CD

readme.txt
: Soubor s popisem obsahu CD a popisem instalace systému

Source
: Složka obsahující zdrojové kódy aplikace

Text
: Složka obsahující text této bakalářské práce, včetně zdrojových souborů v formátu LaTeX

Api
: Složka obsahující dokumentaci vygenerovanou ze zdrojových kódů pomocí nástroje apigen