TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

WEBOVÁ APLIKACE PRO PLÁNOVÁNÍ SPOLEČNÝCH JÍZD AUTEM

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Vojtěch Marhoul**
Vedoucí práce: doc. Ing. Jiřina Královcová, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

CARSHARING WEB APPLICATION

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Vojtěch Marhoul**
Supervisor: doc. Ing. Jiřina Královcová, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Vojtěch Marhoul**
Osobní číslo:	**M12000162**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Webová aplikace pro plánování společných jízd autem**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Proveďte rešerši existujících řešení systému pro plánování spolujízd. Přehledně shrňte možnosti a nedostatky jednotlivých řešení.
2. Navrhněte vlastní systém pro plánování spolujízd tak, aby zahrnoval zadávání nabídek a poptávek, možnou registraci a přihlašování a vhodnou návaznost na mapový systém. Navrhněte základní datové struktury a databázovou strukturu systému.
3. Seznamte se s potřebnými technologiemi.
4. Na základě předchozího návrhu implementujte systém jako webovou aplikaci s využitím vhodných technologií. Systém otestujte na reprezentativním vzorku požadavků a realizací.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

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

Seznam odborné literatury:

[1] **Andi Gutmans, Derick Rethans, Stig Saether Bakken: Mistrovství v PHP 5. COMPUTER PRESS, 2008, 656 stran. EAN: 9788025115190**

[2] **W. Jason Gilmore: Velká kniha PHP 5 a MySQL. Zoner Press, 2011, 736 stran. ISBN 978-80-7413-163-9**

[3] **Nicholas Z. Zakas: JavaScript pro webové vývojáře. COMPUTER PRESS, 2009, 832 stran. EAN: 9788025125090**

Vedoucí bakalářské práce: **doc. Ing. Jiřina Královcová, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2014

Prohlášení

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

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

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

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

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

Datum:

Podpis:

Poděkování

Na tomto místě bych chtěl poděkovat všem, kteří mi při zpracování této bakalářské práce pomáhali, nebo mě jakkoliv podporovali. V první řadě děkuji vedoucí práce doc. Ing. Jiřině Královcové, Ph.D. za konzultace, cenné rady, podněty a připomínky při zpracování mé bakalářské práce. Na závěr bych chtěl poděkovat rodině za všestrannou podporu.

Abstrakt

Tato práce se zabývá problematikou návrhu aplikace pro efektivní plánování společných jízd autem. V rešeršní části práce jsou detailně popsána a zároveň mezi sebou porovnávána obdobná již existující řešení. Hlavním obsahem celé práce je naprogramování vlastní aplikace pro plánování společných jízd autem, včetně jejího popisu. V první části je popsán návrhový vzor zdrojového kódu, požadavky na aplikaci a základní princip, na kterém je software založen. Další část práce se zabývá návrhem šablon, registrací uživatele, mechanismem soukromých zpráv a vytvořením samotného systému pro plánování společných jízd. V závěru práce je popsáno dokončení uživatelského profilu a postup při programování hodnocení uživatelů.

Klíčová slova

PHP, spolujízda, databáze, MVC (Model View Controller)

Abstract

This thesis deals with the problems of proposal application for effective planning common rides by car. In the theoretical part of this thesis there are described and compared in details already existing solution for planning common rides by car alike. The main content of whole thesis is to programme its application for planning common rides by car, including its describtion. In the first part is described proposal application of source code, requirements for application and basic principle, on which the software is based. Farther away It is concerned with proposal stencils, users registration, mechanism of private messages and creating as such system for planning common rides. In conclusion of my thesis, there is described completion users profile and in the process of programming evaluation users.

Key words

PHP, common ride, database, MVC (Model View Controller)

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Má práce bude pojednávat zejména o softwarovém nástroji pro plánování společných jízd autem (dále též spolujízd). Toto téma je ve světě velice aktuální a i to je důvod volby mé práce. Společné jízdy autem plánují převážně mladší lidé, kteří cestují po světě za dobrodružstvím a chtějí poznat nové lidi. Mezi ně patří i studenti, pro které je cestování spolujízdou do školy nebo na studentské koleje po všech stránkách výhodnější, než cestování hromadnou dopravou.

Toto téma jsem si zvolil i na základě pozorování studentů Technické univerzity v Liberci, kteří převážně plánují společné jízdy autem pomocí sociální sítě Facebook. Tento způsob je efektivní do té doby, než spolujízdu poptává, respektive nabízí velké množství uživatelů. Tyto dny nastávají zejména v pátek, kdy odjíždí drtivá většina studentů z kolejí domů a v neděli, kdy opět studenti přijíždějí zpět na kolej. V této době se na sociální síti běžně objevuje přes 50 poptávek/nabídek spolujízdy za den a tím se tento systém stává velice nepřehledným. Neexistuje v něm vyhledávání, ani jednotná formulace pro psaní příspěvků. Uživatel musí tedy postupně pročíst velké množství nabídek/poptávek, než najde položku, která mu vyhovuje. Tato nabídka/poptávka ale nemusí vůbec existovat a může se tak jednat pouze o ztrátu času.

Hlavní cíl této práce je vytvoření aplikace, která plánování společných jízd autem zefektivní. To znamená, zrychlit a zjednodušit vyhledávání poptávek i nabídek nezávisle na tom, kolik jich v systému je. Zrychlit a zjednodušit komunikaci a informativnost mezi uživateli, vytvořit síť důvěryhodných uživatelů a zvýšit tím bezpečnost a pohodlnost cestování. Tato aplikace je určena pro lidi všech věkových kategorií, převážně však pro studenty.

Vlastní text práce sestává ze dvou hlavních kapitol, těmi jsou rešerše (kap. 2) a řešení (kap. 3). V rešeršní části se zabývám popisem a porovnáváním obdobných již existujících řešení. Tato kapitola je rozdělena na dvě podkapitoly, kterými jsou české aplikace a zahraniční aplikace. Ve třetí kapitole týkající se řešení, popisuji postup při vytváření vlastního software. V první částí třetí kapitoly se zabývám volbou typu aplikace, použitými prostředky pro realizaci a popisem požadavků na aplikaci. Dále tato kapitola sestává z návrhu architektury programu a samotného postupu při programování aplikace.

2 Rešerše

V této kapitole popisuji několik již existujících řešení pro plánování společné jízdy autem. Popis jednotlivých aplikací jsem pro lepší přehlednost rozčlenil do dvou hlavních částí, těmi jsou *hlavní menu* a *uživatelský profil*. Členění popisu těchto hlavních částí jsem uspořádal hierarchicky od nejpodstatnějších prvků dané aplikace až po ty méně důležité. Tuto kapitolu jsem rozdělil na dvě hlavní podkapitoly, těmi jsou české aplikace a zahraniční aplikace.

2.1 České aplikace

Jízdomat (www.jizdomat.cz)

Jízdomat je jedna z nejpropracovanějších českých aplikací pro plánování spolujízd autem. Pro plnohodnotné využití je vyžadována registrace, kvůli větší důvěryhodnosti mezi uživateli. Jedná se o webovou aplikaci.

Hlavní menu

Hlavní menu se skládá ze tří položek: *hledání jízd*, *vložit poptávku* a *vložit jízdu*. Na úvodní stránce je aktivní záložka *Hledání jízd*.

Hledání jízd

Zde se nachází formulář pro vyhledávání konkrétní jízdy. Poptávající zadá startovní, cílovou destinaci a datum. Lze zadat i větší rozsah dní, pokud má v plánu poptávající na jedno místo pravidelně dojíždět. Pokud existuje nabídka dle zadaných kritérii, vypíše se uživateli do pole nabídek. Pokud poptávající nezadá do vyhledávacích formulářů žádné údaje, vypíší se z databáze všechny uložené nabídky. Nabídky jsou dále kategoricky rozděleny do čtyř záložek: *zahraniční jízdy*, *holky s holkama*, *akce* a školy, *lyže*.

Zahraniční jízdy – vypíší se jen nabídky, které obsahují zahraniční lokalitu. Uživatel si může buď zvolit výchozí a cílové město, nebo si vybrat z nabízených států, a prolistovat si všechny spolujízdy, které jsou pro daný stát poskytovány.

Holky s holkama – tato záložka slouží pro ženy, které mají zájem cestovat pouze s ženami. Kromě zadaných kritérií pro vyhledání nabídky spolujízdy je automaticky v aplikaci nastaveno vyhledávání nabídek, které vytvořily pouze ženy.

Akce a školy – V této sekci má uživatel možnost vytvořit událost, například festival a nabídnout na tuto akci odvoz. Tímto způsobem lze nabídnout dojíždění například i do konkrétní školy.

Další záložky se mění podle ročního období. V zimě se zde nachází záložka *lyže* a v letním období například *festivaly*. Tyto sekce jsou založeny na stejném principu jako výše popisovaná záložka *akce a školy*. Poslední fixní položkou jsou poptávky spolujízdy, kde uživatelé poptávají řidiče s autem, který by je odvezl z místa A do místa B, v čase specifikovaném v dané poptávce.

Obsah konkrétní nabídky

- jméno řidiče (kontakty se zobrazí pouze, pokud je uživatel přihlášen)
- cena
- počet volných míst v autě
- vybavení auta a kritéria řidiče (kouření v autě, wifi, převoz zvířat, klimatizace, dálniční známka)
- přihlášení cestující do této jízdy (vidí pouze přihlášení uživatelé)
- diskuze (vidí pouze přihlášení uživatelé)

Informace o trase

- mapa trasy
- čas odjezdu
- vzdálenost a předpokládaný čas příjezdu do cíle
- další jízdy poskytované tímto řidičem

Funkční tlačítka

- Do kalendáře – vloží uživateli tuto událost do osobního google kalendáře.
- Vložit na zeď – vloží tuto jízdu na profil Vašeho Facebooku.
- Chci se svézt – pouze pro přihlášené uživatele, po stisknutí tohoto tlačítka, je uživatel přihlášen do jízdy. Majitel nabídky obdrží informativní email o novém pasažérovi. Dále se mohou domluvit na spolujízdě po emailu nebo po telefonu.

Vložit poptávku

Vložit poptávku mohou pouze registrovaní uživatelé. Poptávající má k dispozici formuláře pro výchozí a cílové město. Dále je zde možnost upřesnit místo, například zadáním názvu čtvrtě nebo ulice.

Další položka je požadované datum a čas odjezdu, kde má uživatel možnost zadat *datum a čas (nejdříve)* a *datum a čas (nejpozději),* to znamená časové rozmezí, ve kterém poptávající potřebuje odcestovat. Na výběr je zde i poptat zpáteční cestu, kde se opět zadá časové rozmezí, ve kterém uživatel potřebuje odjet zpět z cílového města do výchozího. Zadat lze i pravidelnou poptávku, kde je možnost výběru většího počtu dní v týdnu.

Jako poslední si uživatel zvolí počet potřebných míst v autě, pokud například poptává jízdu i za někoho jiného. Po kliknutí na tlačítko *uložit poptávku* se poptávka uloží do databáze a lze jí najít mezi již zveřejněnými poptávkami. Řidiči, kteří si tuto poptávku vyhledají a budou chtít poptávajícího svézt, navážou s poptávajícím kontakt buď pomocí emailu, nebo telefonu.

Vložit jízdu

Pro vložení jízdy musí být uživatel opět zaregistrovaný. Vložení jízdy se od vložení poptávky liší pouze několika položkami. Přidat lze libovolný počet měst a lokalit, přes které řidič bude cestou do cílového místa projíždět. Co se týče data a času, zde je opět možné vložit více dní v týdnu pro pravidelnou jízdu, nebo datum a čas zpáteční jízdy z cílového do startovního místa.

Řidič dále stanoví cenu za osobu, ovšem pouze z výchozího do cílového města. Pokud by chtěl případný pasažér svézt pouze do města, které se nachází na trase, na ceně se již řidič s pasažérem dohodne individuálně. Jako poslední uživatel vkládající nabídku zadá počet míst v autě, které nabízí a značku auta, kterým se bude cestovat. Po stisknutí tlačítka *uložit jízdu*, lze najít nabídku v záložce *hledání jízd* a poptávající se mohou do této jízdy přidávat a kontaktovat řidiče.

Uživatelský profil

Osobní profil uživatele si lze pro větší důvěryhodnost různými způsoby nechat ověřit. Pokud přihlášený uživatel vlastní účet na Facebooku, aplikace Jízdomat automaticky

tento profil propojí s účtem na Facebooku a použije z něj některá data, za účelem co nejkompletnějších informací o uživateli.

Osobní profil se dále dělí na dvě hlavní položky, těmi jsou *Můj jízdomat* a *Nastavení profilu*.

Můj jízdomat

Záložka *Můj jízdomat* obsahuje správu všech akcí Jízdomatu, které se daného uživatele týkají. Nachází se zde správa nabídek, poptávek, aut uživatele a černá listina (dále též blacklist). Uživatel může kontrolovat jízdy, na které se přihlásil, může spravovat nabízené jízdy, měnit jejich parametry nebo je mazat. To samé je možné i s poptávkami. Dále je zde možnost si vytvářet a ukládat šablony poptávek a ty poté používat. V osobním profilu je možné vytvořit si i blacklist, to znamená seznam uživatelů, se kterým nastal dříve nějaký problém a uživatel s nimi již nemá zájem cestovat.

Nastavení profilu

V záložce *nastavení profilu* se nachází všechny informace o uživateli a jeho profilu. Lze si zde měnit osobní informace, kontaktní údaje, jazyk a měna (měna, ve které se zadá cena jízdy), nastavení viditelnosti kontaktních údajů (přihlášení/nepřihlášení, ověření/neověření), fotografie a řidičské zkušenosti.

Nastavení profilu dále umožňuje ruční propojení účtu s Facebookem nebo s aplikací, která se zabývá ověřováním. Nastavit si lze také to, jestli uživatel musí nejprve potvrdit přidání uchazeče do jízdy, nebo jestli se uchazeč do jízdy přidá automaticky, po kliknutí na tlačítko *chci se svézt*, které se nachází v dané nabídce.

V profilu si lze měnit i heslo a email, nebo je možné vlastní profil zobrazit tak, jak ho vidí ostatní uživatelé.

Jízdomat je velmi propracovaná aplikace a nabízí spoustu dalších možností. Uživatelé se můžou navzájem po předchozích zkušenostech hodnotit, psát k nabídkám i poptávkám poznámky, nebo si přidávat ostatní uživatele do seznamu přátel. Na hlavní stránce se nachází pro lepší přehlednost boční uživatelský panel, kde jsou zobrazeny *moje jízdy*, *moje poptávky* a *vaši přátelé na jízdomatu*. Pod tímto panelem se nachází informativní tabulka, v té jsou aktuální informace o tom, kdo se právě s kým a kam nechal svézt, nebo kdo koho a jak hodnotil.

Jízdaspolu (www.jizdaspolu.cz)

Jízdaspolu je česká webová aplikace pro plánování spolujízdy autem. Je zde možnost registrace, která ovšem pro základní použití není nutná.

Hlavní menu

Hlavní menu nabízí položky *spolujízdy*, *nejnovější jízdy*, *vložit jízdu*, s*polucestování* a *informace*. Jako úvodní stránka zde slouží položka *spolujízdy*.

Spolujízdy

Záložka spolujízdy obsahuje výpis zveřejněných nabídek i poptávek spolujízd. Položky se automaticky řadí od nejstarších po nejnovější, dle data odjezdu. Na bočním panelu se nachází tabulka, která obsahuje seznam států a u každého státu počet nabízených či poptávaných spolujízd. Po kliknutí na stát se zobrazí pouze takové položky, které obsahují ať už výchozí či cílové město v daném státě.

Obsah konkrétní nabídky

- datum a čas odjezdu
- trasa
- jméno řidiče
- email
- telefon
- počet míst k dispozici v autě
- cena cesty z výchozího do cílového města

Poptávky jsou totožné s nabídkou s tím, že volná místa jsou poptávána a cena za cestu je navržena poptávajícím.

Nejnovější jízdy

V této záložce se vypsané položky seřadí od nejnověji přidaných po ty nejstarší.

Vložit jízdu

Tato sekce obsahuje formulář pro vložení spolujízdy. Tento formulář slouží pro vložení poptávky i pro vložení nabídky. Nejdříve si uživatel vybere, zda bude jízdu poptávat nebo nabízet a poté vyplní potřebné údaje. Těmi jsou výchozí stát, město, města, kterými bude řidič projíždět, cílový stát a cílové město.

Dále lze vybrat počet míst, které v případě řidiče nabízíte a v případě uchazeče poptáváte. Jako poslední uživatel vyplní datum a čas odjezdu. Je zde možnost vytvořit i zpáteční jízdu. Po stisknutí tlačítka *uložit* se nabídka respektive poptávka uloží do databáze a zveřejní v seznamu spolujízd. Uživatelé se poté mohou kontaktovat emailem nebo po telefonu.

Spolucestování

V záložce *spolucestování* se nachází vypsané inzeráty všeho druhu. Převážně jde však o to, seznámit se a sehnat spolucestovatele například na dovolenou, nebo na nějakou akci jako je například festival. Inzerát se skládá z názvu (název by měl být co nejvýstižnější), jména, věku inzeráta, kontaktu a popisu nabídky.

Uživatelský profil

Po zaregistrování a přihlášení získá uživatel přístup na svůj účet. V osobním účtu si lze upravit všechny osobní údaje, jako je email, jméno, telefon, skype nebo heslo.

V osobním profilu je také jednoduchá správa vlastních inzerátů, ať už spolujízdy nebo spolucestování. Tyto inzeráty lze buď měnit, nebo vymazat.

Jízdaspolu slouží převážně pro cestování do zahraničí. Tato aplikace je velmi jednoduchá, sama o sobě neposkytuje žádnou interakci mezi uživateli, vyhledávací formuláře, nebo síť důvěryhodných uživatelů. Nevyužívá jí také tolik uživatelů jako právě výše zmíněný Jízdomat. Jízdaspolu je spíše inzerční webová stránka pro spolucestování.

2.2 Zahraniční aplikace

Ride4cents (www.ride4cents.org)

Ride4cens, v českém překladu jedu za korunu je webová aplikace pro plánování společných jízd autem, vytvořená týmem, jehož členové pochází ze všech možných koutů Evropy. Tuto aplikaci lze přepnout do 18. světových jazyků a pro plnohodnotné využití vyžaduje registraci.

Hlavní menu

Hlavní menu tvoří záložky: *hlavní stránka, hledej, přidat nabídku, registrace, pomoc, bezpečnost*.

Hlavní stránka

Tato stránka se uživateli zobrazí jako úvodní a slouží k hledání spolujezdce, respektive řidiče. Místo vypsaných položek zde poptávající najde formulář, do kterého vyplní kritéria vyhledávání. Startovní místo určí uživatel tak, že vybere stát a město z nabízených možností a poté vyplní již vlastnoručně název města, které spadá pod vybraný stát a (většinou krajské) město. Tím samým způsobem se určí cílová lokalita. Další položky ve formuláři jsou datum odjezdu a tolerance data. Jako poslední si uživatel zvolí, v jakou denní dobu chce odcestovat. Tato možnost slouží jako alternativa přesného času. Po vyplnění celého formuláře lze vybrat buď, *hledám spolujezdce* nebo *hledám řidiče*. Pokud je nějaká položka dle zadaných kritérií uložena v databázi, vypíše se uživateli.

Obsah konkrétní nabídky

- přezdívka uživatele
- email a telefonní číslo (zobrazí se pouze přihlášeným)
- Výchozí a cílové město
- kompletní trasa

Funkční tlačítka

- Přidat k oblíbeným – uloží tuto jízdu do záložky oblíbené, která se nachází v uživatelském profilu.
- Uživatelský profil – přepne na profil nabízejícího/poptávajícího.
- Přeposlat příteli – možnost přeposlat tuto nabídku/poptávku na libovolný email.

Jelikož je formulář pro zadávání nabídky i poptávky stejný, neliší se ničím ani obsah zveřejněné nabídky respektive poptávky. Tato položka neobsahuje odkaz, kterým by se mohl uchazeč přidat automaticky do jízdy, záleží tak pouze na domluvě uživatelů pomocí kontaktů, které jsou v daných nabídkách/poptávkách k dispozici.

Přidat nabídku

Nabídku lze přidat třemi způsoby. V položce druh nabídky lze zvolit *hledám řidiče*, *hledám spolujezdce* nebo *hledám řidiče/mohu vzít spolujezdce*. Pro všechny druhy nabídek zůstává formulář totožný krom položky s názvem: *chci doplnit města, přes která pojedu*. Tato položka se neobjeví pouze u druhu nabídky *hledám řidiče*, která je spíše poptávka po odvozu a poptávající si zde tak zvolí pouze výchozí a cílové město.

Pokud uživatel zvolí druh nabídky *hledám řidiče/mohu vzít spolujezdce*, znamená to, že má k dispozici auto a nezáleží mu na tom, zda se případní cestující svezou s ním, nebo jestli se on sveze na té samé trase s jiným řidičem a jiným autem. Tato nabídka respektive poptávka se poté bude objevovat ve zveřejněných nabídkách a zároveň i poptávkách po spolujízdě.

Ve formuláři tedy uživatel nejprve zvolí o jaký druh nabídky/poptávky se jedná a dále vyplňuje výchozí a cílové město stejně tak, jako tomu bylo u vyhledávacího formuláře. Vybere tedy nejprve stát a krajské město a poté vyplní ručně konkrétní město, které spadá do této oblasti. Stejně tak je tomu v případě nabídky jízdy, kdy řidič hledá spolujezdce a zadává města, přes která bude do cílového města cestovat.

Pomoc a bezpečnost

V těchto záložkách uživatel najde užitečné informace o tom, jak s aplikací zacházet, jak zvýšit svou bezpečnost, nebo jak se stát důvěryhodným členem.

Spočítejte si cenu

Součástí webu je i jednoduchá informativní kalkulačka, určená pro výpočet nákladů, které cestující za cestu vynaloží. Tato kalkulačka má dvě přepínatelné podoby, jednu pro spolujezdce a druhou pro řidiče.

Pokud má v plánu uživatel někam cestovat jako spolujezdec, zvolí si nejprve, jestli se bude pohybovat v jižní a východní Evropě, nebo v severní a západní Evropě. Důvod tohoto rozdělení spočívá v rozdílných ekonomických situacích ve státech těchto částí Evropy, což se projeví i v nákladech na cesty. Dalšími kritérii pro výpočet ceny jsou vzdálenost (v km) a počet spolucestujících. Na základě těchto údajů kalkulačka spočte přibližnou cenu cesty pro jednoho spolujezdce. Výsledná částka je v eurech.

Kalkulačka určená pro řidiče obsahuje opět volbu, po jaké části Evropy se bude řidič pohybovat. Další kritéria pro výpočet ceny cesty jsou cena benzínu za litr, spotřeba auta na 100 kilometrů, vzdálenost a počet pasažérů. Aplikace si zjistí, v jakém jazyce ji uživatel zobrazuje a dle toho automaticky zvolí měnu, ve které zobrazí výslednou sumu nákladů na cestu.

Další důležité součásti

V horizontálním menu se nachází další řada položek, které slouží převážně pro jednodušší vyhledávání. Za zmínku stojí položky *města odjezdu* a *cílová města*. Tyto seznamy slouží pro vyhledávání spolujízdy pouze dle výchozích, respektive cílových měst.

Dále je zde na výběr záložka *nabídky pro nejbližší tři dny*, ve které jsou vypsány pouze ty nabídky a poptávky, ve kterých je datum odjezdu stanoveno maximálně na tři dny dopředu od aktuálního data. *Přidáno v poslední době* je poslední záložka, která slouží pro vyhledávání spolujízd a ve které je seznam nabídek i poptávek spolujízd, které byly přidány v posledních třech dnech (Zohledňuje se tedy datum přidání, nikoliv datum odjezdu).

Uživatelský profil

V profilu přihlášeného uživatele se nachází šest položek, těmi jsou: *zpráva*, *oblíbení*, *mé nabídky*, *profil*, *hodnocení uživatele* a *systém*.

Zpráva

Aplikace umožňuje rozesílání zpráv mezi registrovanými uživateli. V této záložce jsou přijaté i odeslané zprávy uživatele uložené. Odeslat zprávu lze pouze po navštívení profilu adresáta. Po rozkliknutí ikonky pro odeslání zprávy se odesílajícímu zobrazí formulář, kde vyplní předmět zprávy a obsah. Poté zprávu odešle a ta se adresátovi zobrazí právě v záložce *zpráva* v jeho osobním profilu.

Oblíbení

Zde se nachází seznam oblíbených nabídek a oblíbených uživatelů. Jak si přidat nabídku mezi oblíbené je popsáno výše v sekci *obsah konkrétní nabídky*. Stejně tak je tomu v případě uživatele.

Profil

Profil obsahuje všechny údaje o uživateli, které zadal při registraci. Tyto údaje mohou vidět i ostatní uživatelé. Lze si však nastavit, kdo jaké položky v profilu uvidí. Například kontaktní údaje se běžně zveřejňují jen pro oblíbené uživatele.

Hodnocení uživatele

V aplikaci Ride4cents lze cestující hodnotit. V této záložce se nachází formulář pro vyplnění uživatelského jména, které chceme hodnotit.

Po kliknutí na položku s názvem *najít uživatele* se zobrazí další formulář, který slouží k samotnému ohodnocení. Hodnotit lze záporně a pozitivně plusovými nebo mínusovými body. Další částí hodnocení je slovní komentář směřující k hodnocenému uživateli. Po té je hodnocení odesláno a zveřejněno, je možné kohokoliv hodnotit až za další tři dny. Hodnocení je veřejné a přístupné pro všechny. Najít ho lze na osobním profilu uživatele v záložce *přijaté komentáře*.

Systém

Systém je historie přihlašování ve formě informativní tabulky. Zobrazeno je datum a čas přihlášení, IP adresa zařízení a poskytovatel internetového připojení.

Aplikace Ride4cents obsahuje i fixní menu, která jsou cestovateli k dispozici nehledě na to, v jaké sekci právě je. V jednom z nich se nachází seznam nejnověji registrovaných uživatelů. V dalším menu nazvaným *quicklinks* jsou k dispozici odkazy na státy a města. Toto menu slouží k rychlejšímu vyhledání konkrétního města a nabídek nebo poptávek, ve kterých je dané město obsaženo.

3 Řešení

Tato kapitola obsahuje kompletní návrh programu, od databázové struktury, návrhu architektury, rozvržení tříd až po finální podobu aplikace.

3.1 Požadavky na výsledný software

Předmětem této práce je vytvoření aplikace pro plánování společných jízd autem. Aby má aplikace splňovala hlavní požadavek, kterým je efektivnost plánování společných jízd, stanovil jsem si dílčí prvky, kterými software bude disponovat:

- Přidávání nabídek a poptávek bude realizováno pomocí formulářů, kvůli jednotné formě všech položek a jejich snadnějšímu vyhledávání.
- Přidání nabídek i poptávek bude umožněno pouze zaregistrovaným a přihlášeným uživatelům.
- Výpis nabídek a poptávek bude mít formu tabulky, kde jeden řádek obsahuje nejdůležitější informace o konkrétní nabídce, respektive poptávce.
- Sekce s výpisem nabídek i poptávek budou obsahovat vyhledávací menu pro rychlejší nalezení požadované položky.
- U každé zveřejněné nabídky bude možno zobrazit její detail s doplňujícími informacemi.
- V detailu nabídky bude navíc možnost přihlásit se do jízdy. Uchazeč se poté zobrazí majiteli (řidiči) nabídky v administrační části. Řidič se poté může rozhodnout, zda uživatele do jízdy příjme, nebo ho odmítne.
- Každý zaregistrovaný a přihlášený uživatel bude mít k dispozici svůj osobní profil, ve kterém mu bude umožněno mimo jiné měnit si své kontaktní údaje
- Součástí profilu je i hodnocení uživatelů, kde budou vypsána vlastní obdržená hodnocení a také zde bude možnost hodnotit ostatní uživatele. Obdržená hodnocení budou veřejně zobrazena u každého zaregistrovaného uživatele.
- V osobním profilu se bude také nacházet výše zmíněná administrace vlastních nabídek, kde kromě přijímání a odmítání uchazečů bude možnost nabídky za určitých okolností měnit a mazat.
- Další součástí aplikace bude také zasílání soukromých zpráv mezi přihlášenými uživateli. Tyto zprávy bude uživatelům rozesílat i sama aplikace, například ohledně jízdy, do které byl uchazeč přijat.

3.2 Volba typu aplikace

V první fázi bylo potřeba zvolit, jakého typu software bude. Vybírat jsem mohl z desktopové, mobilní nebo webové aplikace. Vzhledem k pokročení techniky a zvyšujícímu se počtu „chytrých telefonů“ mezi lidmi jsem bral v potaz naprogramovat aplikaci pro mobilní zařízení. Problém je v tom, že pořád existují lidé, kteří „chytrý telefon“ nemají, a stolní počítač či notebook je pro ně jediný prostředek pro přístup k internetu. Tato aplikace by měla být k dispozici pro každého, nehledě na to, jakými zařízeními disponuje.

O desktopové verzi programu jsem neuvažoval příliš dlouho. V dnešní době, kdy je internet naprostou samozřejmostí a jeho rychlost více než dostatečná i pro přenos objemných dat, by desktopová aplikace tohoto typu neměla velký význam.

Přistoupil jsem tedy na webovou verzi, oproti desktopové je uživatelsky pohodlnější. Všechna data i se samotným programem jsou uložena na serveru, to znamená, že uživatel potřebuje mít ve svém zařízení ke zprovoznění aplikace k dispozici pouze připojení k internetu a webový prohlížeč. Nemusí se tedy zabývat instalováním žádného softwaru navíc, jako by tomu bylo v případě desktopové nebo mobilní aplikace. S webovými stránkami nemají velký problém ani dnešní mobilní zařízení, pokud jsou webové stránky dobře navrženy.

3.3 Použité prostředky

Pro programování této aplikace jsem zvolil programovací jazyk PHP. Tento jazyk je pro mou práci po všech stránkách plně dostačující. Samozřejmostí je objektově orientované programování (OOP), které je pro rozsáhlejší projekty nutností. PHP dále umí dobře pracovat s výjimkami, které jsou v mé práci důležité. O PHP je také k dispozici velké množství literatury, a jeho podpora je velice široká.

Aplikace bude provozována na webovém serveru Apache. Práci jsem realizoval na lokálním serveru pomocí konfiguračního balíku XAMPP, který umožňuje instalaci Apache, PHP, phpMyAdmin a databáze MySQL, kterou jsem pro svou práci zvolil.

3.4 Návrh architektury programu

Před tím, než jsem začal tvořit samotný kód, bylo třeba zvolit jeho architekturu. Hlavním kritériem pro výběr architektury bylo to, že se jedná o webovou aplikaci, kterou budu programovat v jazyce PHP. Pro tuto kombinaci se nabízí velice známá a osvědčená Model View Controler (dále též MVC) architektura a tu jsem pro svou práci také zvolil. Základní myšlenkou MVC architektury je oddělení logiky od výstupu. Tímto rozdělením lze dosáhnout přehledného a snadno rozšiřitelného zdrojového kódu. Častým problémem spojení logiky s výstupem je tzv. „špagetový“ kód, který je velice nepřehledný, často zbytečně dlouhý a ve většině případů se špatně rozšiřuje.

MVC architektura se skládá ze tří hlavních složek, těmi jsou model, pohled a kontroler.

3.4.1 Model

První složkou tohoto návrhového vzoru je model. Hlavním úkolem modelu je zpracovat všechna data, se kterými bude program operovat. Model se nestará o to, odkud data přišla, ani o to, jaký proces čeká zpracovaná data v dalším kroku.

3.4.2 Pohled

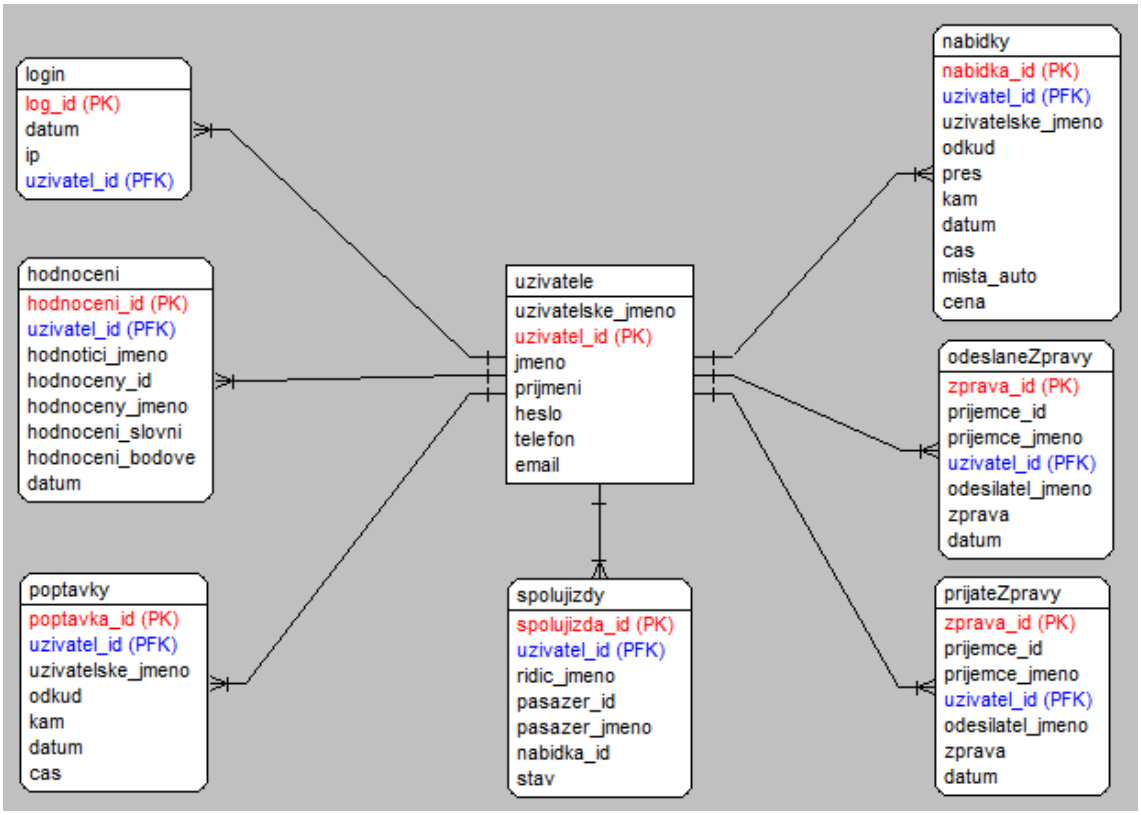
Pohled slouží k zobrazení výstupu uživateli. Jedná se většinou o *html* šablonu, ke které se později přidají data. Pohled se stejně jako model nestará, odkud se data vzala, pouze je zobrazí spolu se šablonou, která přísluší konkrétním datům. Pohled obsahuje minimální množství logiky pouze pro zobrazení šablony a dat.

3.4.3 Kontroler

Model a pohled pracují nezávisle na sobě. Aby byl program založený na této architektuře funkční, potřebuje poslední složku, která model i pohled propojí dohromady. Kontroler získá zpracovaná data od modelu a připraví pohled, který ke zpracovaným datům přísluší. Jedná se tedy o prostředníka, který šetrně spojí logiku s výstupem a umožní komunikaci mezi jednotlivými složkami.

3.5 Data programu

Všechny akce, které se v programu uskuteční, budou závislé na uživateli, tím pádem na uživateli budou závislá i veškerá data programu. Této skutečnosti jsem přizpůsobil i datovou strukturu. Všechny databázové tabulky s daty jsou závislé právě na tabulce, která slouží k evidenci uživatelů (viz. Obrázek 1). Z pohledu uživatele je vztah ke všem ostatním datům 1:N. Znamená to, že například jeden uživatel může přidat do systému více nabídek, ale jedna nabídka může mít pouze jednoho uživatele (vlastníka). Data programu jsou uložena v databázi *MySQL*.



Obrázek 1: Databázová struktura

3.6 Princip směrování a výpis šablon

Základním stavebním prvkem programu je směrování. Aplikace bude obsahovat mnoho sekcí a směrování mezi nimi by mělo být navrženo tak, aby vyhovovalo MVC architektuře. Program jsem rozdělil do tří adresářů dle návrhového vzoru. V adresáři *modely* budou uloženy *php* soubory, které se starají jen o data. Tyto soubory budou obsahovat převážně funkce pro práci s databází. Bude zde například správa uživatelů nebo nabídek spolujízdy. Do adresáře *pohledy* budu umisťovat pouze soubory s příponou *phtml*. *Phtml* se od klasického *html* liší tím, že má větší podporu programovacího jazyka

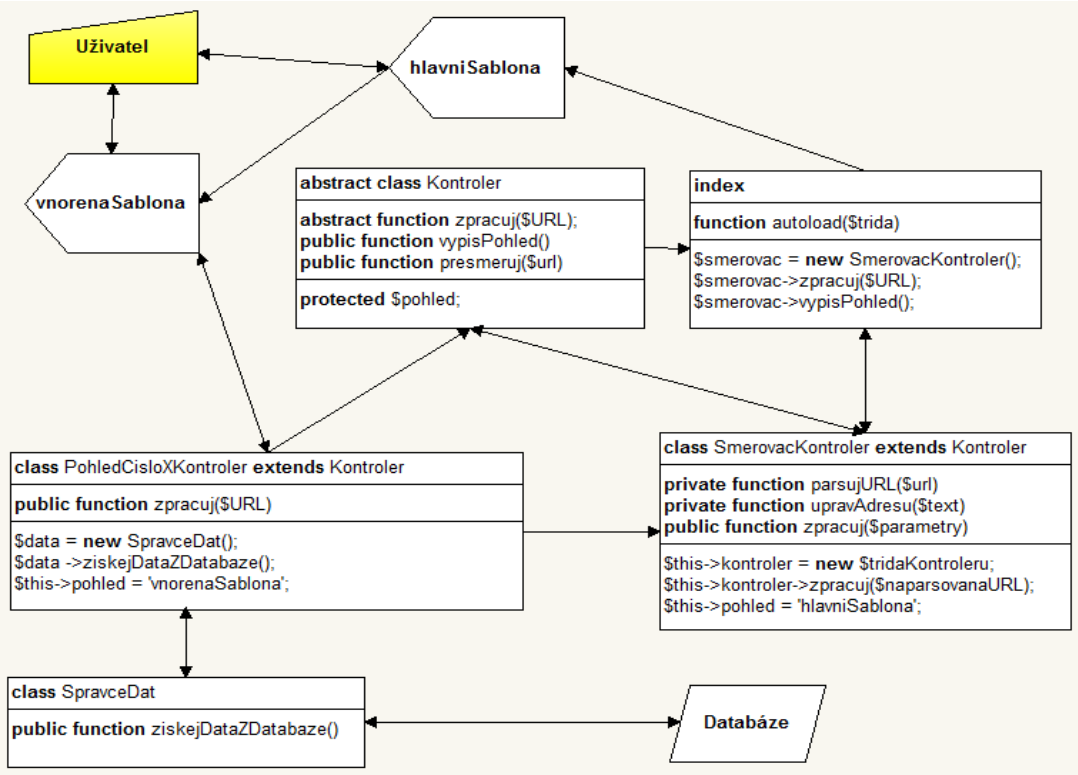
PHP, jako je například zvýrazňování syntaxe. Soubory v adresáři *pohledy* budou obsahovat pouze šablony.

Jelikož bude moje aplikace rozsáhlá, rozhodl jsem se kvůli přehlednosti kódu rozčlenit program tak, aby každá šablona obsahovala v pozadí vlastní třídu. Tato třída bude vždy obsahovat kompletní logiku potřebnou pro zobrazení dané šablony a příslušících dat. Po tom, co jsem zvolil tento princip, bylo potřeba vytvořit systém pro zavolání správné třídy po každé změně URL adresy. Potřeboval jsem tedy, aby všechny URL adresy směřovaly na jediné místo. Webový server *Apache* disponuje konfiguračním souborem *htacces*, díky jehož správnému nastavení jsem docílil nasměrování adres do souboru *index.php*. Pro zpracování URL adresy jsem vytvořil třídu s metodou, která bude adresu v *souboru index.php* přebírat jako parametr. Funkce se tedy bude volat při každé změně URL adresy. V této chvíli bylo třeba zvolit jednotný název pro třídy obsahující logiku, kvůli jejich snadnému zavolání. Jelikož metoda bude volat pouze třídy z adresáře *kontrolery,* rozhodl jsem se pro standardizaci názvu na `nazevTridyKontroler` (viz. Zdrojový kód 1). Pomocí pomocných metod jsem rozdělil adresu dle lomítek do pole a připravil funkce pro úpravu názvu v adrese na požadovaný tvar. V této chvíli je možné vytvořit instanci třídy, a spustit logiku, kterou volaná třída obsahuje. Na tomto principu je založená celá aplikace.

Pro to, abych mohl instanci třídy vytvořit, je třeba nejdříve vložit soubor, který volanou třídu obsahuje. Pro vkládání souboru slouží funkce `require`, která jako parametr přebírá cestu k adresáři s volaným souborem. Jelikož bude program obsahovat mnoho sekcí, kde bude potřeba soubor vkládat, bylo by neefektivní v každé třídě zvlášť tuto funkci volat. Programovací jazyk PHP disponuje funkcí, přímo určenou pro volání tříd. Tato metoda je spuštěna automaticky a přebírá jako parametr název volané třídy. V tomto případě budeme volat soubor, a tak je nutné, aby byly názvy tříd stejné, jako názvy souborů, ve kterých jsou třídy obsaženy. V těle této vkládací funkce jsem vytvořil podmínky ověřující existenci vkládaného souboru. V případě, že je podmínka vyhodnocena kladně, výše zmiňovaná metoda `require` soubor se třídou vloží. Nyní je potřeba zobrazit uživateli správnou šablonu. Jelikož slouží pro každý pohled jiný kontroler, je potřeba aby všechny třídy typu kontroler obsahovaly informace o tom, jaká šablona se má pro daný kontroler zobrazit. Vytvořil jsem si tedy obecnou třídu, ze které budou všechny tyto třídy dědit (viz. Obrázek 2). V obecné třídě jsem založil jednoduchou funkci pro výpis šablony,

přebírající jako parametr název šablony, který této funkci zděděné třídy pomocí proměnné předají.

Pro svou práci jsem potřeboval vytvořit dva druhy šablon. Prvním typem je šablona pro hlavní rozvržení stránky a tím druhým je vnořená šablona, která se bude měnit na základě toho, v jaké sekci se uživatel zrovna nachází. Kontrolery se zpracovávají a volají ve směrovací třídě popsané výše, ve které se zpracovává i URL. V této třídě je potřeba tedy předat funkci pro výpis data o tom, jaká hlavní šablona (rozvržení stránky) se zobrazí. Data o tom, jaká vnořená šablona se zobrazí, obsahují již konkrétní kontrolery volané ve směrovací třídě. Funkci pro výpis šablony jsou předána nejprve data o hlavní šabloně ze směrovací třídy a poté je tato funkce volána v souboru *index.php*. Ve chvíli, kdy je hlavní šablona načtena, jsou funkci určené pro výpis předána data o vnořené šabloně. Ta je vypsána pomocí stejné funkce v těle hlavní šablony.



Obrázek 2: Grafické znázornění rozvržení programu

V aplikaci bude potřeba často uživatele přesměrovat ručně, pokud například návštěvník nebude přihlášený a bude se dožadovat přístupu do sekce pro přihlášené. Pro přesměrování slouží již vytvořená funkce `header`, která na základě parametru (URL adresy), na požadovanou adresu uživatele přesměruje. Tuto funkci jsem vytvořil opět

v obecné třídě, jelikož obdrží data od zděděných tříd, stejně tak jako metoda pro výpis šablony.

```
27      // $parametry = URL adresa
28      public function zpracuj($parametry)
29      {
30      $naparsovanaURL = $this->parsujURL($parametry[0]);
31
32      /*Pokud URL adresa nemá žádné parametry,
33        uživatel bude přesměrován na hlavní stránku */
34      if (empty($naparsovanaURL[0]))
35        $this->presmeruj('hlavniStranka');
36
37      /* Z pole rozparsované adresy vybírám prvky,
38         z kterých potřebuji vytvořit existující kontroler */
39      if($naparsovanaURL[0] == 'profil')
40      {
41      if($naparsovanaURL[1] == 'zpravy')
42      {
43      $tridaKontroleru = $this->upravAdresu($naparsovanaURL[1]) . 'Kontroler';
44      // Do proměnné pohled ukládám šablonu hlavního rozložení
45      $this->pohled = 'profil/zpravy/rozlozeniProfilZpravy';
46      }else{
47      $tridaKontroleru = $this->upravAdresu($naparsovanaURL[1]) . 'Kontroler';
48      $this->pohled = 'profil/rozlozeniProfil'; }
49      }else{
50      $tridaKontroleru = $this->upravAdresu($naparsovanaURL[0]) . 'Kontroler';
51      $this->pohled = 'rozlozeni';
52      }
53      /* Pokud kontroler existuje, vytvořím instanci jeho třídy,
54         pokud ne, přesměruji na chybovou stránku */
55      if (file_exists('kontrolery/' . $tridaKontroleru . '.php'))
56        $this->kontroler = new $tridaKontroleru;
57      else
58        $this->presmeruj('chyba');
59      // volám logiku třídy, včetně vnořené šablony
60      $this->kontroler->zpracuj($naparsovanaURL);
61
62      $this->data['titulek'] = $this->kontroler->hlavicka['titulek'];
63      $this->data['zprava'] = $this->vypisZpravu();
64      }
```

Zdrojový kód 1: Funkce pro zpracování URL adresy

Tímto systémem volání tříd a směrování jsem se inspiroval z webových stránek *itnetwork*, které se věnují programování v mnoha programovacích jazycích.

3.7 Komunikace s databází

Jelikož se jedná pouze o práci s daty, vytvářel jsem veškeré třídy pro práci s databází v adresáři *modely*. Dotazy na databázi jsem musel před použitím nejprve upravit. Hlavním důvodem je bezpečnost zejména před *SQL injection* útokem. Tento útok spočívá

ve vkládání škodlivého kódu do formulářů aplikace za účelem získání cenných dat z databáze. Funkce, které dotazy připraví, jsem vytvářel v samostatné třídě spolu s metodou, která administrátora k databázi připojí.

Aby byla aplikace zabezpečená před *SQL injection* útokem musel jsem oddělit samotný dotaz na databázi od jeho parametrů. Princip této ochrany spočívá ve vložení zástupného znaku do dotazu na místo proměnné. Tento dotaz je nejprve připraven a až poté se spustí spolu s parametry, které nejdříve nahradil zástupný znak. Metoda pro komunikaci s databází bude přebírat tedy dva parametry. Jeden parametr obsahuje text dotazu a druhý parametry dotazu. Pro každý druh dotazu jsem připravil jednu funkci.

```
27      public static function vratJedenRadek($dotaz, $parametry = array()) {
28      /* V proměnné dotaz je uložený text dotazu spolu se zástupnými snaky,
29         metoda prepare tento dotaz připraví */
30      $vratDotaz = self::$spojeni->prepare($dotaz);
31      // Metoda execute připojí k dotazu parametry, a dotaz spustí
32      $vratDotaz->execute($parametry);
33      // Metoda fetch vrátí 1 řádek z databáze, který poté navrátíme
34      return $vratDotaz->fetch();
35    }
```

Zdrojový kód 2: Funkce pro komunikaci s databází

V adresáři *modely* jsem vytvořil další třídy pro práci s daty, ty budou pouze využívat těchto funkcí a předávat jim odděleně data od dotazu. Například třída pro správu poptávek a nabídek bude obsahovat pouze takové dotazy na databázi, které se týkají nabídek nebo poptávek.

```
111     public function vratNabidku($nabidka_id)
112     {
113       return Databaze::vratJedenRadek('
114       SELECT `nabidka_id`, `uzivatel_id`, `uzivatelske_jmeno`, `odkud`, `pres`,
115       `kam`, `pres`, `datum`, `cas`, `mista_auto`, `cena`
116       FROM `nabidky`
117       WHERE nabidka_id = ?
118      ',array($nabidka_id));
119     }
```

Zdrojový kód 3: Funkce pro vrácení konkrétní nabídky spolujízdy

Funkce pro vrácení konkrétní nabídky spolujízdy přebírá parametr od uživatele, dle kterého vyhledá konkrétní nabídku spolujízdy v databázi. Za zmínku stojí příkaz `where`, který slouží k porovnávání vyhledávaného prvku s prvky uloženými v databázi. Na místo

parametru, který zadal uživatel, je zde výše popsaný zástupný znak (otazník). Proměnná, ve které jsou obsažena data od uživatele, je až jako druhý parametr funkce.

3.8 Tvorba šablon, evidence uživatelů a mechanismus zpráv

Dalším krokem je návrh hlavního rozložení stránky. Hlavním rozložením rozumíme šablonu, která obsahuje neměnné prvky bez ohledu na to, v jaké části aplikace se uživatel v dané chvíli nachází. Hlavní šablona pro tuto aplikaci bude obsahovat v první řadě navigační menu a tělo dokumentu s funkcí pro výpis vnořených šablon. Dalšími prvky jsou název aplikace a patička. Navigační menu jsem vytvořil pomocí seznamu a jeho položky jsou: *hlavní stránka*, *přidat nabídku*, *přidat poptávku*, *poptávky* a *profil*.

Záložka *profil* je přístupná pouze pro zaregistrovaného a přihlášeného uživatele. Administrace profilu nabízí řadu dalších prvků, kterými jsou: *moje údaje*, *moje hodnocení*, *moje nabídky*, *historie přihlašování* a *zprávy*. Vytvořil jsem druhou hlavní šablonu, která se aktivuje ve chvíli, kdy uživatel zobrazí administraci profilu. Jediný rozdíl mezi těmito šablonami je v tom, že rozložení pro uživatelský profil obsahuje navíc již zmíněné menu pro navigaci v rámci profilu.

3.8.1 Evidence uživatelů

Poslední viditelnou položkou v hlavních šablonách je odkaz *přihlásit se*. Tento odkaz, který přesměruje návštěvníka do sekce s přihlašovacími formuláři, jsem umístil do pravého horního rohu. Pro přihlášení uživatele do aplikace je nutná nejprve registrace.

Pro realizaci registrace jsem jako první vytvořil šablonu s kontaktním formulářem a uložil stejně tak jako všechny šablony do adresáře *pohledy*. Aby registrace proběhla úspěšně, musí návštěvník vyplnit všechny položky formuláře, kterými jsou: *uživatelské jméno*, *jméno*, *příjmení*, *heslo*, *telefon*, *email* a *příklad*. Políčko příklad slouží jako ochrana proti spamu. Po návštěvníkovi je požadován výsledek jednoduchého matematického příkladu. Systém disponuje výjimkami, které uživatele upozorní, pokud není formulář vyplněn správně.

Kontroler registrace bude používat dva modely (třídy pro práci s daty), jeden pro správu uživatelů a druhý pro správu emailu. V modelu pro správu uživatelů jsem si připravil funkci pro vložení uživatele do databáze. Tato funkce s názvem `registruj`, přebírá jako parametry proměnné, které zájemce o registraci zadal do formulářů. Jako první

jsem musel ověřit unikátnost uživatelského jména, které v některých případech slouží jako identifikace uživatele. Pro toto ověření jsem založil metodu `overUnikatnost`, která se též nachází ve třídě pro správu uživatelů. Ověřovací funkce přebírá jako parametr proměnnou, která bude obsahovat uživatelské jméno uchazeče o zaregistrování. V tělu této metody jsem vytvořil SQL dotaz, který prohledá tabulku se zaregistrovanými uživateli. Pokud funkce zjistí, že ověřované jméno se v databázi uživatelů již nachází, vrátí hodnotu „0“, v opačném případě vrátí „1“.

Aby byl uživatelský učet bezpečný, připravil jsem ještě jednu metodu, s názvem `zasifruj`. Jako parametr bude přebírat heslo, které uchazeč o registraci zadal do formuláře a toho heslo zašifruje. PHP disponuje funkcí `hash`. Parametry této funkce jsou druh šifry, heslo a náhodné znaky (sůl), které funkce do hesla vloží. Výsledek šifrování je otisk hesla (*hash*), z něhož prakticky nelze rekonstruovat původní heslo.

Tímto je vše důležité pro registraci připraveno a registrační funkci je možné dokončit. Kvůli jednoduššímu uložení převzatých údajů do databáze se parametry nejprve uloží do pole. Všechny parametry kromě hesla jsou v poli uložené beze změny. Heslo se nejprve pomocí připravené funkce zašifruje a až poté je do pole uloženo.

Před tím, než budou údaje uloženy do databáze, je potřeba přidat několik podmínek. První, hlavní podmínka ověří, zda uchazeč o registraci zadal do formulářů všechny hodnoty. Pokud je tato podmínka splněna, začnou se ověřovat vnitřní podmínky. První z nich ověří, zda se od sebe neliší zadaná hesla (musí být stejná), druhá podmínka ověří, zda byla zadána správná hodnota do antispamové ochrany a poslední podmínka ověří unikátnost uživatele. V případě, že je jedna z podmínek vyhodnocena negativně, je vyvolána chybová zpráva (výjimka). Pokud jsou tyto podmínky vyhodnoceny kladně, uživatel je vložen pomocí bezpečnostně upraveného dotazu do databáze.

Ve třídě kontroleru příslušící registraci jsem nejprve vytvořil příkaz, který zakládá instanci třídy pro správu uživatelů, abych mohl registrační metodu v této třídě použít. Funkce příjme parametry pomocí metody `$_POST`, která slouží k přenosu dat z vyplněného formuláře. Tímto je registrace kompletní. Po zaregistrování je odeslán uživateli informativní email.

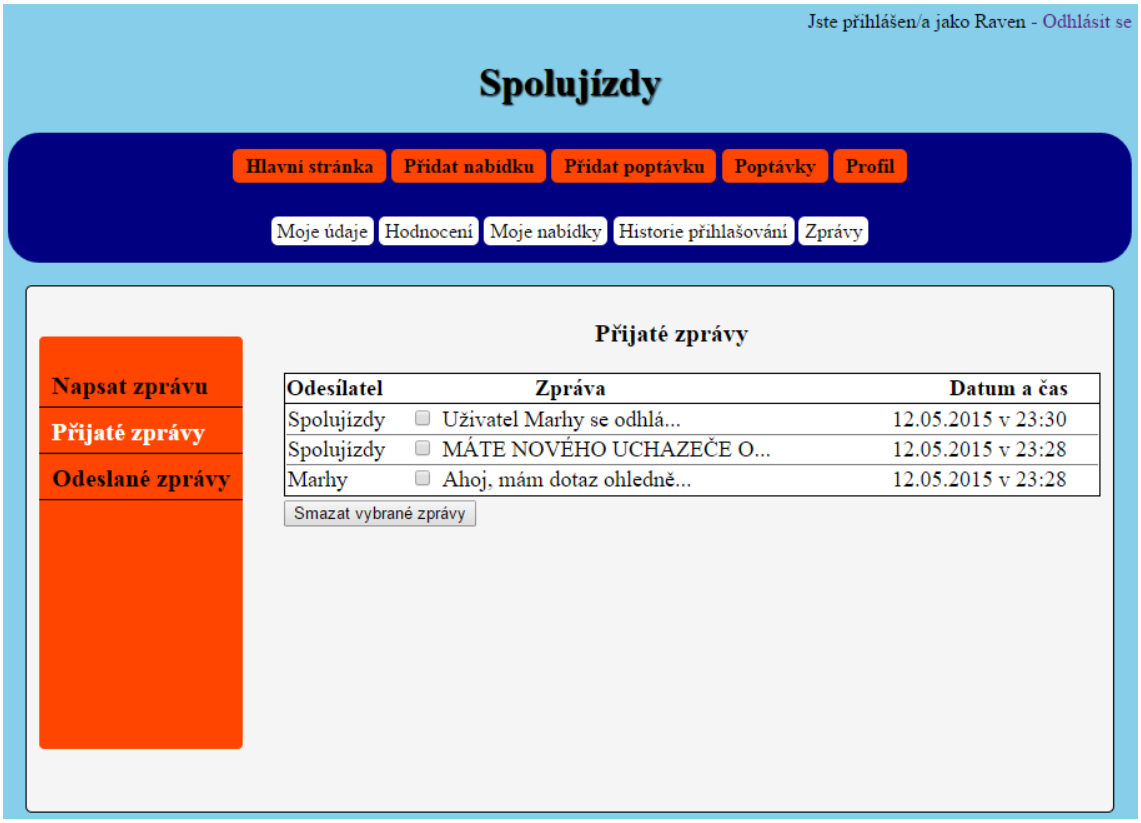
Do proměnné `pohled`, zděděné z obecného kontroleru je zde vkládán název šablony s registračním formulářem. Pohled s touto šablonou se v této chvíli vypíše v těle hlavního rozložení stránky. Jestliže registrace proběhne úspěšně, zavolá se funkce s názvem `presmeruj` a přesměruje uživatele do přihlašovací sekce odkud je již uživateli umožněno se přihlásit.

Přihlášení je stejně tak, jako všechny ostatní prvky aplikace, založeno na podobném principu, jako zaregistrování uživatele. Ve směrovací třídě, kde se zpracovávají URL adresy je načtena třída kontroleru, která přísluší přihlašování. Tato třída opět pracuje s modelem pro správu uživatelů, ze kterého volá funkci pro přihlášení. Tato metoda přebírá jako parametr přihlašovací jméno a heslo. V jejím těle se heslo zašifruje pomocí stejné metody jako při registraci. Poté vyhledá v databázi uživatele na základě jména a zašifrovaného hesla. Pokud neexistuje shoda, systém vypíše výjimku s informací o neplatných přihlašovacích údajích. V opačném případě je uživatel přihlášen. Aby bylo možné s uživatelem pracovat v rámci celé aplikace, vytvořil jsem ve správci uživatelů funkci, která ukládá data o uživateli do globální proměnné `$_SESSION`. Ve třídě kontroleru pro přihlášení je tato metoda volána. Parametry metody jsou opět data z přihlašovacího formuláře předané pomocí proměnné `$_POST`. Název šablony s přihlašovacím formulářem je předán proměnné `pohled` a tento pohled se v hlavní šabloně vypíše. V hlavní šabloně je také vypsaný uživatel, který je právě přihlášen. Pro odhlášení uživatele jsem použil funkci `unset`, která vymaže data o uživateli ze `$_SSESSION`.

3.8.2 Mechanismus zpráv

Zaregistrovanému uživateli se zpřístupní možnost rozesílání soukromých zpráv ostatním zaregistrovaným uživatelům. Administrace zpráv se nachází v osobním profilu uživatele pod záložkou *zprávy*. Jedná se o třetí a poslední hlavní šablonu, obsahuje navíc vertikální navigační menu, pro práci se zprávami. Menu se skládá ze tří položek, těmi jsou: *napsat zprávu*, *přijaté zprávy* a *odeslané zprávy*. Šablona určena k vytvoření a odeslání zprávy obsahuje formulář s textovým polem pro obsah a s kolonkou pro adresáta. Jelikož je při registraci uživatele ošetřena unikátnost uživatelského jména, je možné jméno použít pro identifikaci příjemce. Po správném vyplnění obou položek lze pomocí potvrzovacího tlačítka zprávu odeslat. Odeslaná zpráva se adresátovi zobrazí v záložce *přijaté zprávy* (viz. Obrázek 3). Šablona přijatých zpráv má formu tabulky. Každý řádek tabulky obsahuje čas, kdy byla zpráva odeslána, uživatele, který zprávu odeslal

a část obsahu zprávy. Uživateli je umožněno přijaté i odeslané zprávy také mazat. Tímto způsobem jsem vytvořil výpis všech přijatých zpráv uživatele. Pro zobrazení celého obsahu konkrétní zprávy je třeba kliknout na konkrétní položku tabulky, která je zároveň odkazem. Na tomto principu jsou založené i odeslané zprávy uživatele.



Obrázek 3: Výpis přijatých zpráv

Třída kontroleru, která obsluhuje zprávy je podobná třídě, která zpracovává URL adresy. Na základě URL zjistí, v které části se uživatel právě nachází a dle toho zavolá vnořenou šablonu a data. V kontroleru zpráv potřebuji pracovat s uživateli a se zprávami, vytvořil jsem tedy instance tříd pro správu uživatelů a správu zpráv. V detailu zprávy je třeba zobrazit jednu konkrétní zprávu a naopak ve výpisu přijatých respektive odeslaných zpráv potřebuji zobrazit všechny zprávy přihlášeného uživatele. V modelu pro správu zpráv jsem založil SQL dotazy pro výpis jedné zprávy a výpis všech zpráv. Tyto funkce jsou volány ve třídě kontroleru zpráv a přebírají jediný parametr, kterým je identifikační číslo právě přihlášeného uživatele. Metoda pro výpis konkrétní zprávy přebírá jako parametr identifikační číslo vypisované zprávy, které je obsaženo v URL adrese.

Pro odeslání zprávy slouží metoda `odesliZpravu`. K realizaci tohoto procesu funkce potřebuje znát informace o příjemci a odesílateli, obsah zprávy a datum odeslání.

Před samotným odesláním zprávy potřebuji ověřit, zda adresát vůbec existuje. K tomu jsem použil pomocnou metodu, fungující na podobném principu, jako metoda pro ověření unikátnosti uživatelského jména při registraci. Funkce pro odeslání zprávy uloží do pole opět všechny přijaté parametry a poté ověří, zda jsou vyplněny všechny údaje pro odeslání zprávy, nebo zda je adresát evidován v databázi. Pokud jsou všechny podmínky vyhodnoceny kladně, zpráva se uloží do databáze.

V kontroleru zpráv, kde je funkce pro odeslání zprávy volána, je třeba zjistit aktuální datum. K tomu jsem použil předdefinovanou metodu `StrSFTime`. Návratovou hodnotou této funkce je aktuální datum a čas. Tyto hodnoty jsou pomocí proměnné předány jako parametr funkce pro odeslání zprávy. Dalšími parametry jsou identifikační číslo a uživatelské jméno příjemce a odesílatele. Uživatelské jméno příjemce je k dispozici prostřednictvím metody `$_POST`, odeslané z formuláře. Ostatní údaje o adresátovi a odesílateli jsou zjištěny prostřednictvím modelu pro správu uživatelů, který obsahuje funkce pro výpis údajů o uživateli. Poslední parametr, nutný ke správnému spuštění funkce je obsah zprávy. Ten jsem opět získal z formuláře pomocí proměnné `$_POST`. V této chvíli metoda obsahuje všechny potřebné parametry a může uložit zprávu do databáze. Příjemce ji nalezne v přijatých zprávách a odesílatel v odeslaných.

Dalším druhem zpráv je e-mail. Ten bude sloužit pouze jako poskytovatel informací o důležitých událostech v rámci aplikace. Rozesílat e-mail bude tedy pouze systém, nikoliv uživatelé. Kvůli přehlednosti jsem vytvořil pro práci s emailem samostatnou třídu s názvem `spravceEmailu`. Jedná se o práci s daty, tím pádem soubor s touto třídou patří do adresáře *modely*. Pro odesílání emailu jsem použil funkci `mb_send_mail`. Metoda příjme jako parametr adresu příjemce, předmět emailu, obsah zprávy a hlavičku. V hlavičce mohou být nejrůznější informace ohledně e-mailu, včetně kódování nebo adresy odesílatele. Pokud jsou všechny údaje platné, email je odeslán adresátovi. O to, jakým způsobem odesílání e-mailu funguje, se v této chvíli není potřeba starat, to je již součástí funkce.

Funkci je volána například ve výše popisovaném kontroleru registrace. Zde je do parametrů metody pro odeslání e-mailu vložen příjemce, tím je právě registrovaný uživatel. Odesílatelem je aplikace (spolecne-jizdy.cz) a do předmětu a obsahu zprávy jsem zde

vložil informace o tom, že byl uživatel úspěšně zaregistrován. Součástí zprávy jsou i přihlašovací údaje uživatele, kvůli jejich případnému zapomenutí. E-mail je odeslán, pouze pokud zaregistrování uživatele proběhne úspěšně. Celý mechanismus je obsažen v bloku pro zachycování výjimek `try` a `catch`.

3.9 Tvorba systému pro plánování společných jízd

První mou myšlenkou bylo, zda zpřístupnit plánování jízd i nezaregistrovaným, tím pádem nepřihlášeným uživatelům. Výhody tohoto řešení by spočívaly například v praktičtějším využití aplikace pro uživatele, který by chtěl naplánovat jízdu pouze jednorázově. Tím by se ale snížila úroveň bezpečnosti a na základě již známé pohodlnosti uživatelů internetu by se do aplikace zaregistroval málokdo. Díky tomu by postrádaly smysl další prvky aplikace pro registrovaného člena. Přistoupil jsem tedy na verzi s registrací. Poté, co se uživatel přihlásí, nezpřístupní se mu jen jeho osobní profil, ale i záložky v hlavním navigačním menu, určené pro přidávání poptávek a nabídek. Přidání nabídky i poptávky jsem realizoval opět pomocí formulářů. Řidič nabízející svezení vyplní trasu jízdy, datum a čas odjezdu, počet volných míst v autě a cenu za osobu. Trasa jízdy se skládá z povinných a nepovinných položek. Povinnými položkami jsou výchozí a cílové místo. Údaje, které uživatel není povinen zadat, jsou města, která leží na trase mezi výchozím a cílovým městem. Formulář umožňuje zadat celkem tři města ležící na trase. Pokud tato města uživatel zadá, jsou poté zohledněna ve vyhledávání nabídek spolujízdy. Pokud jsou všechny údaje vyplněny správně, vloží se nabídka do systému. Ta je poté vypsána na hlavní stránce společně s ostatními již dříve přidanými nabídkami.

Příklad: Uživatel vkládající nabídku do systému, zadá výchozí místo A a cílové místo B. Dále vloží města C a D nacházející se na trase. Uživatel, který hledá spolujízdu z místa C do místa D, tuto informaci zadá do vyhledávacích formulářů a zobrazí se mu mimo jiné i spolujízda z místa A do místa B.

Vypsané položky mají formu tabulky, kde jeden řádek obsahuje jednu nabídku (viz. Obrázek 4). Na hlavní stránce je k dispozici také formulář pro vyhledávání, pro případ, že nabídek bude v systému uloženo mnoho. Pro vyhledání nabídky je možné zadat čtyři různá kritéria, těmi jsou výchozí místo, cílové místo, datum odjezdu a čas odjezdu. Vyhledávání není ničím omezeno, tím pádem stačí zadat například jen výchozí město, a uživateli budou zobrazeny jen ty jízdy, které mají stejné výchozí město. Stejně tak je

tomu i s cílovým městem, datem a časem. Nevypisují se nabídky, jejichž datum je starší než aktuální datum. Stejně tak je tomu i u výpisu poptávek. Vyhledávání na základě času jsem upravil tak, aby nebylo nutné zadávat přesný čas hledané nabídky. Pokud uživatel zadá například čas 18:00, vypíší se mu všechny nabídky, s časem v rozmezí 18:00 – 18:59.

Jak je vidět z obrázku, u každé nabídky je možné zobrazit v novém okně její detail. Oproti základním údajům o jízdě, vypsaným v seznamu nabídek, detail disponuje ještě informacemi o řidiči a funkčními prvky. Pokud není uživatel, který prohlíží detail nabídky v aplikaci přihlášený, nejsou mu tyto informace navíc zobrazeny. V opačném případě je možno řidiče kontaktovat, buď zasláním soukromé zprávy v rámci aplikace, nebo pomocí zveřejněných kontaktních údajů.

Vyhledej nabídku

Odkud | Kam | Datum (dd.mm.rrrr) | Čas (--:--) | vyhledat

Nabídky spolujízdy

Odkud	Kam	Uživatel	Datum	Čas odjezdu	Volná místa	
Praha	Pardubice	Raven	13.05.2015	20:00	3	detail
Olomouc	Třebíč	Dita	14.05.2015	16:00	3	detail
Velvary	Beroun	Kuba_007	15.05.2015	15:00	2	detail
Nový Bor	Liberec	Raven	15.05.2015	12:00	3	detail
Mladá Bolesl	Poděbrady	Dita	16.05.2015	15:30	2	detail
Česká Lípa	Mělník	Kuba_007	17.05.2015	18:45	3	detail
Jičín	Jablonec nad	Raven	18.05.2015	14:00	1	detail
Chrudim	Svitavy	Dita	18.05.2015	20:20	1	detail
Karlovy Vary	Příbram	Kuba_007	19.05.2015	11:20	2	detail
Nový Bor	Liberec	Raven	23.05.2015	01:01	3	detail
Nový Bor	Liberec	Raven	23.05.2015	01:58	3	detail
Nový Bor	Liberec	Raven	27.05.2015	00:00	3	detail
Nový Bor	Liberec	Raven	29.05.2015	01:58	4	detail

Obrázek 4: Výpis nabídek spolujízdy

Detail nabídky disponuje ještě jedním prvkem, tím je tlačítko *chci se svézt*. Po jeho použití se uživatel stane uchazečem o svezení, nikoliv pasažérem. Každý zaregistrovaný uživatel má k dispozici v osobním profilu administraci vlastních zveřejněných nabídek pro spolujízdu. Zde lze uchazeče do jízdy přijmout, v tuto chvíli se z uchazeče stane pasažér a počet volných míst v autě se automaticky zmenší o 1.

Uchazeče o jízdu lze i odmítnout, například kvůli nedostatečnému místu v autě, nebo na základě uchazečova hodnocení. V administraci je dále také možné nabídku smazat, nebo změnit datum a čas odjezdu. Tyto úkony je možné provést pouze v tom případě, pokud v jízdě není evidovaný žádný uchazeč a také pouze pokud není datum odjezdu v nabídce starší než aktuální datum. V případě staršího data taktéž nelze manipulovat s přijatými ani s nepřijatými uchazeči o tuto jízdu.

Uživatel má možnost se z jízdy také odhlásit, ať už jako uchazeč, nebo i poté, co byl již přidán do jízdy jako pasažér. O všech změnách ohledně konkrétní jízdy jsou zainteresovaní uživatelé informováni jak interními zprávami, tak prostřednictvím e-mailu.

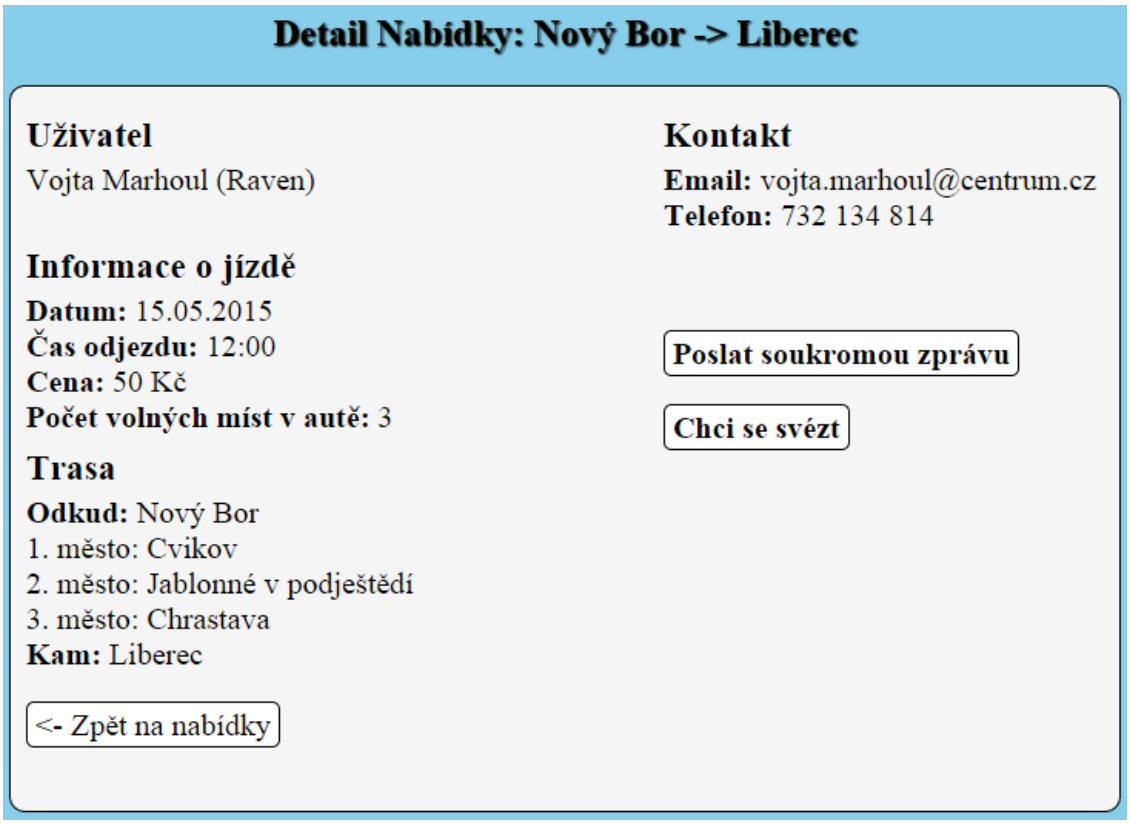
Přidání poptávky spolujízdy je založeno na stejném principu jako vložení nabídky. Pomocí formuláře poptávající do systému vloží informace o tom, odkud kam a v jakém čase potřebuje cestovat. Tato data jsou poté společně se jménem uživatele vypsána v záložce *poptávky*, stejným způsobem jako nabídky spolujízdy. Záložka *poptávky* je tedy přínosná pro řidiče hledajícího pasažéry do svého auta. Vyhledávání v poptávkách probíhá stejným způsobem jako vyhledávání v nabídkách. Pokud by řidič nalezl jemu vyhovující poptávku, může poptávajícího kontaktovat a domluvit se s ním na společné jízdě.

3.9.1 Programová část

V kontroleru nabídky potřebuji pracovat s uživatelem a nabídkou, vytvořil jsem zde tedy instance pro volání tříd pro správu uživatelů a správu nabídek. V modelu správce nabídek jsem si připravil funkci pro uložení nabídky do databáze. Parametry převzaté z formuláře zde jsou uloženy do pole. Pokud bude podmínka, ověřující zdali jsou vyplněny všechny údaje, vyhodnocena kladně, vloží se data do databáze. Funkce je volána opět v kontroleru pro přidání nabídky, kde metoda přebírá data z formuláře. Přidat nabídku je umožněno jen přihlášenému uživateli. V kontroleru jsem proto vytvořil podmínku, která ověří, zda je globální proměnná `$_SESSION` prázdná nebo ne. Pokud není, znamená to, že jsou v ní uloženy údaje o aktuálně přihlášeném členu. V případě, že je podmínka vyhodnocena negativně, uživatel je přesměrován pomocí funkce `presmeruj` do přihlašovací sekce. Tímto způsobem jsem zrealizoval i přidání poptávky spolujízdy.

V kontroleru pro výpis nabídek opět použiji modely pro práci s uživateli a nabídkami. Funkce pro vrácení nabídky přebírá parametry z vyhledávacího formuláře. Pokud

v těchto proměnných nejsou žádná data, metoda pro vrácení nabídky zavolá dotaz na celou tabulku s nabídkami a vypíše ji. Pokud uživatel zadal nějaké vyhledávací kritérium, funkce pro výpis nabídek pomocí podmínek ověří, jaký parametr obsahuje data a na základě tohoto parametru vyhledá v databázi uživatelem požadovanou položku. Šablona pro výpis nabídek se skládá z tabulky, do které jsou prostřednictvím třídy kontroleru vrá-



Obrázek 5: Detail nabídky spolujízdy

cená data vypsána. Pro vztah mezi uživateli a konkrétní nabídkou spolujízdy jsem vytvořil v databázi tabulku *spolujízdy*. Tato tabulka obsahuje informace o řidiči, jenž nabídku zveřejnil, uchazeči o tuto spolujízdu, dále potom identifikační číslo nabídky a stav. Stav znázorňuje pozici uchazeče. V této kolonce je hodnota „0“, pokud použil uchazeč v detailu nabídky možnost zaevidování se do jízdy. Ve chvíli, kdy řidič v administraci dané nabídky uchazeče příjme, změní se stav na hodnotu „1“. Pokud by řidič uživatele odmítl, tato položka se z tabulky spolujízdy vymaže.

Do odkazu na detail nabídky jsem přidal identifikační číslo konkrétní nabídky, které je uchováno v proměnné, aby bylo možné na základě tohoto čísla nabídku v databázi vyhledat a vypsat všechny podstatné informace. Pomocí SQL dotazů zjišťuji informace o tom, v jakém postavení je uživatel vůči dané nabídce.

Následně je pomocí podmínek ověřeno to, zda není přihlášený uživatel zároveň majitelem dané nabídky, pokud ne, je uživateli zobrazena pravá část detailu nabídky. V kontroleru, který obstarává detail této nabídky, jsou volány funkce z modelu pro správu uživatelů a nabídek. Tyto funkce zjistí z databáze informace o tom, zda je uživatel v dané jízdě již evidován. Pokud není, zobrazí se uživateli možnost pro zaevidování se do jízdy, v opačném případě je mu předložen odkaz, kterým se lze z jízdy opět odhlásit. Tímto způsobem jsou zjišťovány informace o uchazeči z pohledu majitele nabídky v administraci.

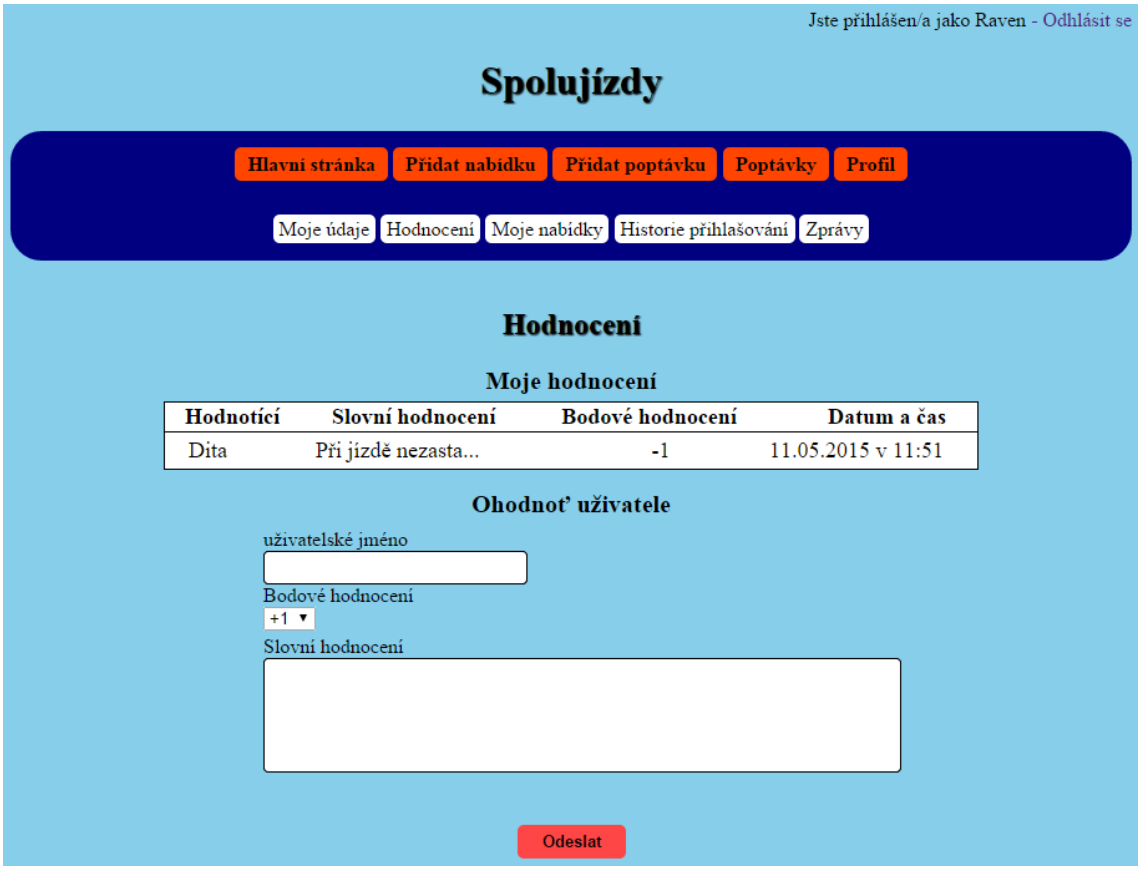
3.10 Hodnocení uživatelů

Předtím, než jsem začal tvořit systém pro hodnocení uživatelů, bylo potřeba rozmyslet se, kdo koho může hodnotit. Stejně tak, jako u plánování společných jízd jsem se rozhodl i zde, že hodnocení i hodnotící budou pouze zaregistrovaní a přihlášení uživatelé. Anonymita hodnotících uživatelů by způsobila menší věrohodnost hodnocení. Hlavním důvodem je však zachovat hlavní myšlenku aplikace, spočívající v tom, že buď je uživatel přihlášený a může vše, nebo není a je mu dovoleno jen prohlížet. Uživatel může hodnotit řidiče pouze na základě předchozích zkušeností s hodnoceným. To znamená, že pokud hodnotící s hodnoceným za poslední půl rok (kvůli relevanci hodnocení) neabsolvoval žádnou jízdu, nebude mu umožněno řidiče ohodnotit. V případě, že uživatel s řidičem již absolvoval jízdu, může ho ohodnotit. Pokud by chtěl jeden a ten samý uživatel hodnotit řidiče dvakrát za sebou, stanovil jsem časový interval mezi hodnoceními na 31 dní.

Začal jsem vytvořením šablony, která se nachází pod odkazem pro hodnocení v osobním profilu uživatele (viz. Obrázek 6). Rozhodl jsem se kvůli pohodlnějšímu používání pro jednu šablonu, ve které budou zobrazena vlastní hodnocení a zároveň zde bude možnost hodnotit i ostatní řidiče. Pro výpis vlastních hodnocení jsem vytvořil tabulku. V jednom řádku tabulky bude jedno hodnocení, stejně tak, jako je tomu například u přijatých zpráv. Pod tabulku s výpisem vlastních hodnocení jsem přidal formulář, prostřednictvím kterého uživatel bude hodnotit. Rozhodl jsem se pro hodnocení skládajícího se ze dvou částí, těmi jsou slovní a bodové. Ve formuláři jsem tedy vytvořil rozbalovací menu s možnostmi kladného a záporného hodnocení, ve formě plusových a mínusových bodů. Pro zadání slovního hodnocení jsem použil obyčejné textové pole. Poslední částí formuláře je okénko pro vyplnění uživatelského jména hodnoceného a dále tlačítko, prostřednictvím kterého se data odešlou třídě kontroleru pro hodnocení.

Další šablonou, která se týká hodnocení, je šablona detailu hodnocení. Detail jsem potřeboval vytvořit, kvůli zobrazení celého textu slovního hodnocení. V tabulce pro výpis všech hodnocení jsem tedy vytvořil odkaz na detail. Do parametru odkazu jsem přidal pomocí proměnné identifikační číslo daného hodnocení, aby měla šablona s detailem informace o tom, jaké konkrétní hodnocení je třeba vypsat.

Poslední šablona, kterou jsem vytvořil, slouží pro zobrazení hodnocení jiného uživatele. Použil jsem stejnou šablonu, jako tu pro výpis vlastních hodnocení (bez formuláře).



Obrázek 6: Šablona pro hodnocení

Programové řešení jsem opět postavil na třídách s kontrolery. Tyto třídy volají připravené funkce z modelů pro výpis nebo vkládání hodnocení. Větší problém nastal při řešení samotného udělení hodnocení, díky tomu, že hodnotící již musel absolvovat jízdu s hodnoceným, aby mohl hodnotit. Potřeboval jsem tedy umožnit hodnotit pouze těm, kteří byli přijati řidičem do jízdy a zároveň se tato jízda již uskutečnila. Ošetřit bylo potřeba ještě několik dalších situací, jako například to, že hodnotící mohl být přijat do jízdy,

která ovšem ještě neproběhla, nebo uživatel mohl být v jízdě, která se již uskutečnila evidován pouze jako uchazeč. V těchto případech nelze řidiče hodnotit.

Pro všechny tyto případy jsem v modelu správce uživatelů vytvořil ověřovací funkci. Jako parametr je funkci předáváno uživatelské jméno hodnoceného, tedy řidiče. Použil jsem připravenou funkci pro výpis všech nabídek hodnoceného uživatele, na základě jeho uživatelského jména. Nyní bylo třeba porovnat aktuální datum s datem všech vrácených nabídek. Pro zjištění aktuálního data jsem použil funkci `date`. Tato funkce vrátí aktuální datum ve formátu YYYY-MM-DD. V tomto formátu jsou uložena také data nabídek v databázi, tím pádem lze takto formátovaná data v jazyce PHP porovnat relačními operátory <, > nebo =. V první podmínce je tedy porovnáváno aktuální datum s datem nabídky hodnoceného řidiče. Tuto podmínku jsem vložil do cyklu, ve kterém jsou procházena data nalezených nabíde. Pokud je podmínka vyhodnocena kladně (nalezne nabídku, která má starší datum než je aktuální), zavolá se funkce pro výpis spolujízdy. Spolujízda je hledána na základě identifikačního čísla nalezené nabídky a identifikačního čísla hodnotícího uživatele. Pokud takový záznam v tabulce spolujízdy existuje, znamená to, že hodnotící byl buď v dané jízdě pouze zaevidován jako uchazeč, nebo byl do spolujízdy přijat.

Vytvořil jsem tedy vnořené podmínky. První ověří, zda nějaký záznam v tabulce spolujízd vůbec existuje, pokud ne, funkce vrátí hodnotu „0“ a pokračuje se dalším cyklem. Pokud je podmínka vyhodnocena kladně, další vnořená podmínka ověří, zda má stav nalezené položky v tabulce spolujízd hodnotu „1“, pokud ne, funkce navrátí hodnotu „0“ a pokračuje se dalším cyklem. V opačném případě funkce nalezla uživatele, který má oprávnění hodnotit tohoto řidiče. Další záznamy tedy není nutné procházet, proto je v těle této podmínky vrácena hodnota „1“ a cyklus může být ukončen.

Pomocí této funkce je ověřováno v již připravené vkládací metodě to, zda je možné hodnocení do databáze uložit nebo ne. Funkce pro vložení hodnocení používá ještě dvě pomocné metody. První funkce vyhodnotí, zda hodnocený uživatel v databázi vůbec existuje a druhá ověří, zda hodnotící uživatel řidiče v minulosti již nehodnotil. Pokud ano, ověřuje se dále, zda od té doby uplynulo již 31 dní. V případě, že ano, lze uživatele znovu ohodnotit.

3.11 Dokončení osobního profilu a popis dalších součástí

Aby měl uživatel přehled o tom, kdy a odkud se do aplikace přihlásil, vytvořil jsem jednoduchou historii přihlašování. Šablonu jsem navrhl jako tabulku, s datem přihlášení a IP adresou. Ve třídě kontroleru pro přihlášení jsem použil funkci `StrFTime`, pro zjištění aktuálního data a času. Pro zjištění IP adresy přihlašovaného jsem použil předdefinovanou funkci `$_SERVER` s parametrem `REMOTE_ADDR`. Tyto údaje se ve chvíli přihlášení ukládají do databáze společně s údaji o uživateli, jemuž hodnoty náleží.

Poslední záložka, která se nachází v uživatelském profilu, má název *moje údaje* a obsahuje osobní informace přihlášeného uživatele. Ve třídě kontroleru této sekce jsou volány funkce pro výpis informací o aktuálně přihlášeném členu. Informace jsou získány z tabulky zaregistrovaných uživatelů. V šabloně příslušící této třídě jsem vytvořil formulář, pomocí kterého lze měnit vlastní e-mail a telefonní číslo. Šablonu jsem vytvořil i pro zobrazení cizího profilu. Jedná se o sekci s osobními informacemi, která obsahuje křížové odkazy pro interakci s daným uživatelem. V nabídce je možnost zaslání soukromé zprávy, udělení hodnocení, anebo výpis obdržených hodnocení. Jména uživatelů slouží v rámci celé aplikace zároveň jako odkazy pro zobrazení osobního profilu.

3.11.1 Archivace dat, výjimky a ošetření formulářů

Aby se nabídky a poptávky spolujízdy nearchivovaly v databázi zbytečně dlouho, vytvořil jsem funkce pro jejich vymazání, pokud jsou staršího data nežli půl roku. Ostatní data jsou pro funkci aplikace potřebná, nebo má uživatel možnost vymazat tato data ručně.

Aplikace disponuje také systémem pro práci s výjimkami. Pro jejich použití jsem vytvořil v adresáři *modely* třídu, která je prázdná a pouze dědí ze třídy `exeption`. Díky výjimkám je uživatel informován o všem, co se podařilo anebo nepodařilo, jako například zaevidování se do jízdy, zaslání zprávy a jiné.

V aplikaci bylo nutné ošetřit formulář pro registraci uživatele proti zadání nerelevantních údajů. Vytvořil jsem proto funkce, které ověřují správný tvar e-mailové adresy a telefonního čísla. Dále je po uživateli požadováno zadání hesla o délce minimálně šesti znaků.

4 Závěr

Cílem mé práce bylo vytvořit efektivnější systém pro plánování společných jízd autem než je ten, kde spolujízdu plánují studenti, tedy Facebook. Výsledkem mé práce je aplikace pro plánování společných jízd autem, která disponuje přehledným vyhledáváním nabídek a poptávek, interakcí mezi uživatelem a aplikací, hodnocením uživatelů a správou vlastních nabídek. Software dále umožňuje správu osobního profilu nebo rozesílání soukromých zpráv mezi uživateli.

Aplikaci jsem měl v plánu již během vytváření zveřejnit a poskytnout ji studentům k testování. Dalším plánovaným krokem bylo do programu implementovat mapový systém. Tyto kroky se mi zatím nepodařilo uskutečnit, jelikož systém obsahuje mnoho vstupů a jejich ošetření mi zabralo více času, než jsem předpokládal. V době odevzdání práce je aplikace již veřejně dostupná na adrese *www.spolecne-jizdy.cz* a její testování spolu s implementací mapového systému plánuji v nejbližší době uskutečnit.

Software disponuje velkým potenciálem pro postupné rozšiřování. V první řadě bude třeba upravit vzhled, který zatím nepůsobí příliš profesionálním dojmem. Dalšími prvky pro vylepšení může být rozšíření nabídek spolujízdy o možnost zadat libovolný počet měst na trase, nebo o možnost naplánovat pravidelné spolujízdy.

Tato práce mě velice bavila a rozšířila mi obzory, co se týká objektově orientovaného programování v jazyce PHP. Aplikaci mám v plánu dále zdokonalovat a uvést do plnohodnotného provozu.

5 Seznam použité literatury a dalších zdrojů

[1] Andi Guymans, Derick Rethans, Stig Saether Bakken: Mistrovství v PHP 5. COMPUTER PRESS, 2008, 656 stran. EAN: 9788025115190

[2] W. Jason Gilmore: Velká kniha PHP 5 a MySQL. Zoner Press, 2011, 736 stran. ISBN 978-80-7413-163-9

[3] Nicholas Z. Zakas: JavaScript pro webové vývojáře. COMPUTER PRESS, 2009, 832 stran. EAN: 9788025125090

[4] Jednoduchý redakční systém v PHP objektově (MVC). *itnetwork.* [online]. 2015 [cit. 2015-04-07]. Dostupné z: *http://www.itnetwork.cz/objektovy-mvc-redakcni-system-v-php*

[5] Přehled vlastností CSS. *jakpsatweb.* [online]. 16.12.2014 [cit. 2015-03-05]. Dostupné z: *http://www.jakpsatweb.cz/css/css-vlastnosti-hodnoty-prehled.html*

[6] PHP 5 Tutorial. *w3schools.* [online]. 1999-2015 [cit. 2015-04-13]. Dostupné z: *http://www.w3schools.com/php/*