ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Backend pro objednávkový/rezervační systém
Student:	Michal Kluzáček
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Cílem práce je revize stávajícího backend projektu „Webový objednávkový systém na pronájem zařízení/věcí" řešeného v SP1, SP2 a jeho rozšíření o API pro připojení externích aplikací.

- Na základě existující implementace vytvořte analytickou dokumentaci backend ve stavu "as is", tedy odpovídající současné implementaci.

- Na základě dokumentace refaktorujte stávající kód s ohledem na jeho budoucí rozšíření.

- Navrhněte a realizujte vhodné API umožňující napojení externích aplikací.

- Připravte vhodné prostředí pro budoucí rozvoj systému (verzovací nástroj, tiketovací nástroj, autodeploy, testovací prostředí např. Vagrant).

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 6. prosince 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra Softwarové inženýrství

Bakalářská práce

Backend pro objednávkový/rezervační systém

Michal Kluzáček

Vedoucí práce: Jiří Hunka

16. května 2016

Poděkování

Rád bych poděkoval Michalu Kváčkovi za skvělé vedení týmu v předmětech BI-SP1 a BI-SP2, díky němu jsme vypracovali projekt „Webový objednávkový systém na pronájem zařízení/věcí“ do stavu v jakém je, což mi umožnilo vytvořit tuto bakalářskou práci. Dále bych chtěl poděkovat Martinu Židovi za bezproblémovou spolupráci při vytváření API pro připojení externích aplikací k našemu systému. A v neposlední řadě děkuji Ing. Jiřímu Hunkovi za vedení této práce a za rady, které mi dal při její tvorbě.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 16. května 2016 .....

Odkaz na tuto práci

Kluzáček, Michal. *Backend pro objednávkový/rezervační systém.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

V práci se zabývám backendem webového portálu na pronájem a půjčování věcí. Zaměřil jsem se na jeho analýzu, dokumentaci a také jsem provedl refaktor daného kódu. Dále jsem vytvořil API pro připojení externích aplikací k tomuto portálu a testovací prostředí pro budoucí vývoj.

Klíčová slova webový portál, backend, dokumentace, API, testovací prostředí, Nette, PHP, MySql, UML

Abstract

This thesis is about backend of website for leases and rents. I focused on analysis, documentation and refactoring of the code. Furthermore I created API for connection of external applications and testing enviroment for future development.

Keywords web portal, backend, documentation, API, testing enviroment, Nette, PHP, MySql, UML

Obsah

Seznam obrázků

Úvod

Každý z nás má doma mnoho věcí, které téměř nepoužívá. Proto jsem se rozhodl zapojit se do práce na projektu, jehož cílem bylo vytvořit webový portál, na kterém mohou lidé tyto věci jednoduše pronajímat, nebo si naopak vypůjčit věc, kterou potřebují, ale nechtějí si ji kupovat.

V rámci předmětů BI-SP1 a BI-SP2 jsme začali tento projekt realizovat, a to v týmu složeného z šesti lidí, jmenovitě: Michal Kváček, Jaroslav Veselý, Martin Žid, Marek Dostál, Nguyen Thi Tuyet Trang, Michal Kluzáček.

V předmětu BI-SP1 jsme provedli analýzu systému a vytvořili příslušné UML diagramy. Tyto diagramy však neodpovídají stávajícímu řešení systému a byly spíše obecným návrhem webového portálu bez promítnutí použitých technologií. Proto ve své práci tyto diagramy vytvořím tak, aby odpovídaly současné implementaci.

Samotná implementace systému probíhala v rámci předmětu BI-SP2, kde jsme vzhledem k rozsáhlosti projektu vytvořili funkční prototyp, který splňuje základní nároky na daný systém.

Téma jsem si vybral, protože mě zaujala myšlenka daného portálu a jeho využití v praxi. Dalším důvodem byla práce, kterou jsem již strávil na daném projektu, a proto bych rád svojí prací pomohl případným vývojářům, kteří by se rozhodli tento projekt dokončit, v jejich snažení a proměnil tak hodiny strávené celým našim týmem v něco, co by mohla spousta lidí využít.

Kapitola 1

Cíl práce

Cílem práce je vytvořit analytickou dokumentaci stávajícího projektu „Webový objednávkový systém na pronájem zařízení/věcí“ řešeného v BI-SP1, BI-SP2. Dokumentace obsahuje tyto diagramy:

- případy užití,
- doménový model,
- diagram tříd,
- diagram aktivit,
- databázový model,
- model nasazení.

Na základě této dokumentace poté provedu refaktor kódu, který se bude skládat především ze smazání nepoužívaných tříd a funkcí, odstranění duplicitního kódu, popisu jednotlivých funkcí a udržení konzistence kódu.

Dále navrhnu a zrealizuji API pro připojení externích aplikací. Důraz budu klást na mobilní aplikaci, na které pracuje Martin Žid, se kterým v této části práce spolupracuji.

V poslední řadě vytvořím prostředí pro budoucí rozvoj systému. To obsahuje:

- verzovací nástroj,
- ticketovací nástroj,
- autodeploy,
- testovací prostředí.

Kapitola 2

Analýza systému

V této kapitole se zaměřím na analýzu backendu systému ve stavu „as-is“, tedy stavu, který odpovídá současné implementaci systému. Podíváme se na použité technologie, poté na požadavky na systém a na strukturu jednotlivých částí systému, kde využijeme UML diagramů.

2.1 Použité technologie

2.1.1 PHP

PHP je velice populární jazyk používaný na vývoj webových aplikací. Co odlišuje PHP od ostatních jazyků, je to, že kód je prováděný na straně serveru, kde vytvoří HTML, které je poté posláno klientovi. Klient tedy dostane výsledek daného skriptu, ale neví, jaký kód byl k dosažení výsledku použitý. Největší výhodou PHP je jeho jednoduchost pro nováčka, ale zároveň obsahuje spoustu nástrojů i pro profesionálního programátora. [1]

2.1.2 MySQL

MySQL je relační databáze typu DBMS (database managment system) a vychází z deklarativního programovacího jazyka SQL (Structured Query Language). Je šířen jako Open Source. [2]

Díky své licenci a rychlosti je v poslední době téměř nejoblíbenějším systémem. MySQL je malý, rychlý a jednoduchý databázový systém. [2]

2.1.3 Nette

Nette je kompletní framework pro PHP, který výrazně zjednodušuje tvorbu webových aplikací. Jeho autorem je český vývojář David Grudl. Framework je kvalitně objektově navržený a v Čechách velmi rozšířený. Fungují na něm velké projekty jako např. GE Money, Slevomat, ČSFD a také obrovská spousta menších webů a e-shopů. [3]

Nette je framework postavený na architektuře MVP, což je zkratka pro model-view-presenter. Aplikace stojí na komponentách třech typů, které se v aplikaci dělí o 3 základní úlohy – řízení, logiku a výstup. [3]

1. **Presentery (presenters), řízení** – Presenter je komponenta, se kterou komunikuje uživatel. Předá jí parametry a ona mu vrátí HTML stránku. Presenter typicky parametry předá modelům, od kterých získá data. Tato data předá pohledům (šablonám), které data začlení do nějakého HTML kódu. Tento HTML kód pošle presenter uživateli do prohlížeče. Funguje tedy jako takový prostředník. [3]

2. **Modely (models), logika** – Obsahují logiku aplikace, jako např. práci s databází nebo výpočty. Každá datová entita má většinou svůj model (uživatel, článek, komentář,...). [3].

3. **Pohledy (templates, česky šablony), výstup** – Obsahují Latte šablony s HTML kódem. Latte je šablonovací jazyk, který do HTML šablon umožňuje vkládat data z PHP pomocí speciálních značek. [3]

2.2 Funkční požadavky

V této části se budu zabývat funkčními požadavky. Nejdříve zmíním, jaké požadavky byly vytvořeny během návrhu systému, a poté stav požadavků v současné době.

2.2.1 Evidence položek

- **Návrh** – Každý registrovaný uživatel má možnost vkládat na web nabídky položek. Položky budeme třídit do kategorií. U položek budeme evidovat: název položky, detailní popis položky, počet, zařazení do kategorie, očekávaná cena za pronájem na den, výše vratné zálohy, termíny, kdy je položka dostupná, lokalita, ve které je možné položku převzít a vrátit. Nepovinnými položkami jsou: doplňkové soubory (např. smlouva řídící pronájem, návod k vkládané položce), fotografie, klíčová slova. Uživatel bude mít možnost svoji nabídku upravit nebo smazat.

- **Současný stav** – Současný stav odpovídá návrhu.

2.2.2 Evidence uživatelů

- **Návrh** – Systém bude umožňovat správu uživatelských účtů. Správou účtu rozumíme přidávání, mazání (deaktivace) a upravování. Účty budou dvojího typu – osoba a firma. U osob budeme evidovat jejich jméno, příjmení, adresu, e-mail a telefon. U firemních účtů budeme ukládat IČ firmy, jméno, telefon a e-mail. Uživatel bude mít možnost změnit si

heslo a telefon. Administrátor může jakýkoli účet i zablokovat (např. za opakované porušování podmínek serveru).

- **Současný stav** – Správa firem v současné době umožňuje přidat firmu k vybranému uživateli pomocí IČ. K této firmě je poté možné přidávat uživatele jako zaměstnance. Neexistuje však možnost dostat se na stránku, která by zobrazovala položky nabízené danou firmou nebo informace o dané firmě. U uživatele je možné nastavit stav zablokovaný, ale tento stav momentálně nemá žádný vliv na chování účtu. Je ale možné vybraný účet skrýt, položky skrytého uživatele pak nebudou zobrazovány na stránce a nebude možné si je vypůjčit.

2.2.3 Evidence výpůjček

- **Návrh** – V systému budeme uchovávat informace o historii půjčované položky. Klíčová je informace o uživateli, který si danou věc půjčil, a o tom, kdy si ji půjčil a kdy ji vrátil.

- **Současný stav** – Systém uchovává pouze informaci o tom, zda byla položka vrácena, nikoli datum kdy. Vše ostatní odpovídá návrhu.

2.2.4 Hodnocení položek

- **Návrh** – Hodnocení položek bude sloužit k hodnocení položek, které si uživatel vypůjčil.

- **Současný stav** – Návrh odpovídá současnému stavu.

2.2.5 Hodnocení uživatelů

- **Návrh** – Systém bude umožňovat hodnocení uživatelů. Každý uživatel bude mít dvě hodnocení: hodnocení nájemce a hodnocení pronajímatele.

- **Současný stav** – Hodnocení uživatelů v současném řešení chybí. Je pouze možné zobrazit všechna hodnocení, která daný uživatel provedl.

2.2.6 Integrace sociálních sítí

- **Návrh** – Do systému je zapotřebí vhodně integrovat sociální sítě kvůli snadnějšímu šíření mezi lidi. Podporovány budou tyto sítě: Facebook, Google+ a Twitter. Integrací se rozumí vložení tlačítka „To se mi líbí“ (a jeho obdoby), tlačítka pro sdílení obsahu mezi své přátele/kruhy.

- **Současný stav** – Současný stav odpovídá návrhu.

2.2.7 Interní komunikace

- **Návrh** – Do systému bude implementován jednoduchý komunikační kanál. Bude založený na prostém odesílání zpráv mezi uživateli. Zprávy po odeslání nebudou umožňovat smazání ani úpravu.
- **Současný stav** – Interní komunikace nebyla implementována. Systém nabízí pouze odeslání e-mailu danému uživateli.

2.2.8 Kalendář

- **Návrh** – Data rezervací jednotlivých předmětů budeme zobrazovat v kalendáři.
- **Současný stav** – Současný stav odpovídá návrhu.

2.2.9 Kategorizace položek

- **Návrh** – Vkládané položky budou organizovány v kategoriích. Systém bude nabízet rozhraní pro správu kategorií. U kategorie budeme ukládat její jméno a popis položek, které do kategorie patří. Kategorie musí mít možnost vytvoření hierarchické struktury (např. Nářadí: Kladiva, Kleště, Šroubováky). Kategorie budou spravovány v administraci.
- **Současný stav** – Současný stav odpovídá návrhu.

2.2.10 Mapy

- **Návrh** – U každé položky bude možnost zobrazení mapy, na které bude vyznačena lokalita nabídky. Zároveň se mapa bude zobrazovat i v přehledu kategorií a při vyhledávání. Na té budou zobrazeny nalezené položky.
- **Současný stav** – Mapa se zobrazuje pouze u detailu položky s vyznačeným místem k pronájmu.

2.2.11 Moderování položek

- **Návrh** – Administrátor bude mít možnost mazat nevhodné položky.
- **Současný stav** – Současný stav odpovídá návrhu. Navíc může moderátor položku upravit stejně jako její majitel.

2.2.12 Nahlášení nevhodného obsahu

- **Návrh**– V systému bude funkce pro nahlášení nevhodného obsahu (či chyby) libovolným uživatelem webu. Pro nahlášení nevhodného obsahu bude využíván reportovací formulář.

- **Současný stav** – Současný stav odpovídá návrhu. Uživatel může nahlásit položku za spam, nevhodný obsah a „ostatní“.

2.2.13 Oblíbené

- **Návrh** – Uživatelé budou mít možnost uložit si položku i uživatele do „oblíbených“. K takto uloženým položkám (a uživatelům) tak získají rychlejší přístup.

- **Současný stav** – Tato funkce není v současné době implementována.

2.2.14 Ověřování uživatele

- **Návrh** – Pro systém je klíčová integrace ověřování totožnosti uživatele. Stupně oveření budou tři – prostá registrace (i přes Facebook), ověření účtu MojeID pomocí telefonu, ověření účtu MojeID pomocí pošty. Uživatel musí mít možnost dodatečně spojit svůj účet s profilem na Facebooku, případně s MojeID.

- **Současný stav** – Stupeň ověření je pouze jeden, a to pomocí SMS s unikátním kódem, který uživatel zadá do systému.

2.2.15 Upozorňování

- **Návrh** – Systém bude upozorňovat uživatele na blížící se termín jejich rezervace pomocí e-mailu. Uživatel si bude nastavovat, kolik dní před termínem jeho výpůjčky má obdržet upozornění

- **Současný stav** – Současný stav odpovídá návrhu. Toto upozornění je možné také zaslat pomocí interní notifikace, která se zobrazí uživateli po přihlášení.

2.2.16 Vícejazyčné verze

- **Návrh** – Systém bude vyvíjen ve dvou jazykových mutacích – česky a anglicky. Zároveň bude podporovat jednoduchou správu překladu (např. Gettext) a jednoduché přidání nového jazykového balíčku.

- **Současný stav** – Systém je připraven na vícejazyčné verze. V současné době však není překlad hotový. Pro překlad do vícejazyčných verzí je použito rozšíření kdyby/translation.

2.2.17 Vyhledávání věcí

- **Návrh** – Systém bude umožňovat vyhledávání podle kategorií (s hierarchickou strukturou), lokality, klíčových slov a slov obsažených v popisu

a nadpisu. Výsledky budou řazeny podle relevance výsledků, lokality, hodnocení položky a hodnocení uživatele.

- **Současný stav** – Vyhledávání v současné době hledá zadaný výraz ve jméně položky, popisu, jméně a příjmení uživatele. Položky jsou řazeny podle data vložení.

2.3 Nefunkční požadavky

Nefunkční požadavky vymezují hranice ve smyslu nároků na systém. Ne tedy co bude systém umět, ale jak. Mezi nefunkční požadavky patří například:

- platforma,
- výkon,
- dostupnost.

Současný stav požadavků byl testován na poslední stabilní verzi systému, která je dostupná na `http://www.bagrujemeshunkou.cz/`.

2.3.1 Podpora prohlížečů

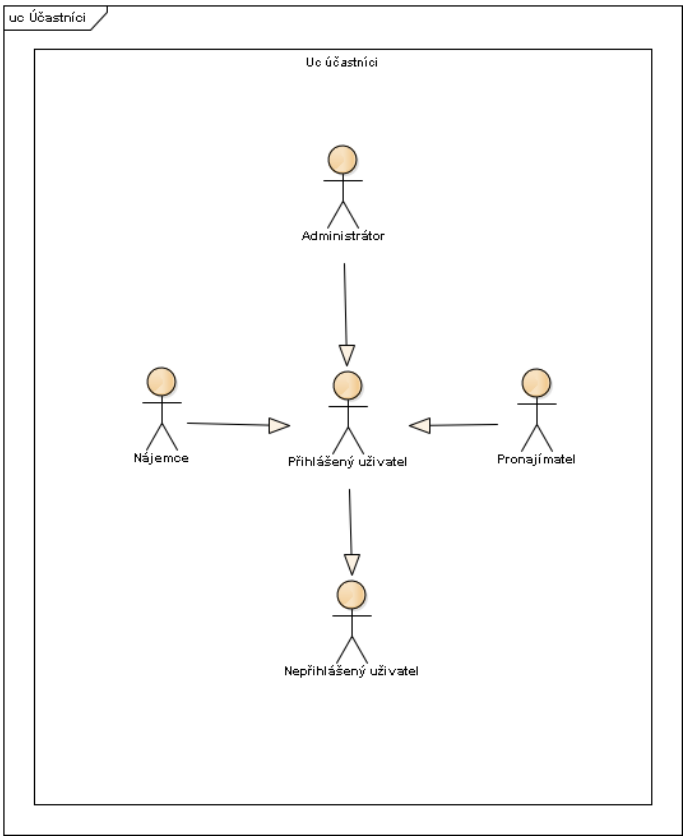
- **Návrh** – Webová služba se bude zobrazovat správně v prohlížečích Internet Explorer, Mozilla Firefox, Google Chrome a Opera. U uvedených prohlížečů budou podporovány poslední dvě verze (aktuálně stabilní) v době nasazení.
- **Současný stav** – Současný stav splňuje tento požadavek.

2.3.2 Unikátnost registrací

- **Návrh** – Vlastní registrace do systému bude vyžadovat unikátnost registračního e-mailu nebo Facebook účtu.
- **Současný stav** – Současný stav splňuje tento požadavek.

2.3.3 Webová služba

- **Návrh** – Systém bude dostupný přes webové prohlížeče.
- **Současný stav** – Současný stav splňuje tento požadavek.



Obrázek 2.1: Seznam účastníků.

2.4 Případy užití

Tyto diagramy byly vytvořeny ve spolupráci s Martinem Židem a Nguyen Thi Tuyet Trang. Původní návrh případů užití lze nalézt v příloze. Z původního návrhu byly odstraněny funkce, které nebyly implementovány. Dále byl ke každému případu přidán krátký popis, který obsahuje i některé minoritní funkce s ním spojené, což napomáhá k přehlednosti diagramů.

2.4.1 Seznam účastníků

Diagram účastníků viz 2.1.

- Nepřihlášený uživatel
  - zobrazuje a vyhledává položky, které jsou v systému uloženy. Může se registrovat a stát se přihlášeným uživatelem.
- Přihlášený uživatel

  - může nabízet své vlastní položky a půjčovat si věci ostatních uživatelů. Má svůj uživatelský profil, který může editovat.

- Nájemce
  - si půjčuje položky od pronajímatele. Vytváří tedy výpůjčky a objednávky, které po jejich skončení hodnotí.

- Pronajímatel
  - půjčuje své položky nájemcům. Výpůjčky schvaluje nebo zamítá a po jejich skončení je hodnotí.

- Administrátor
  - má přístup ke všem funkcím na administraci portálu.

2.4.2 Evidence položek

Diagram evidence položek viz 2.2.

2.4.2.1 Nabídnutí položky k pronájmu

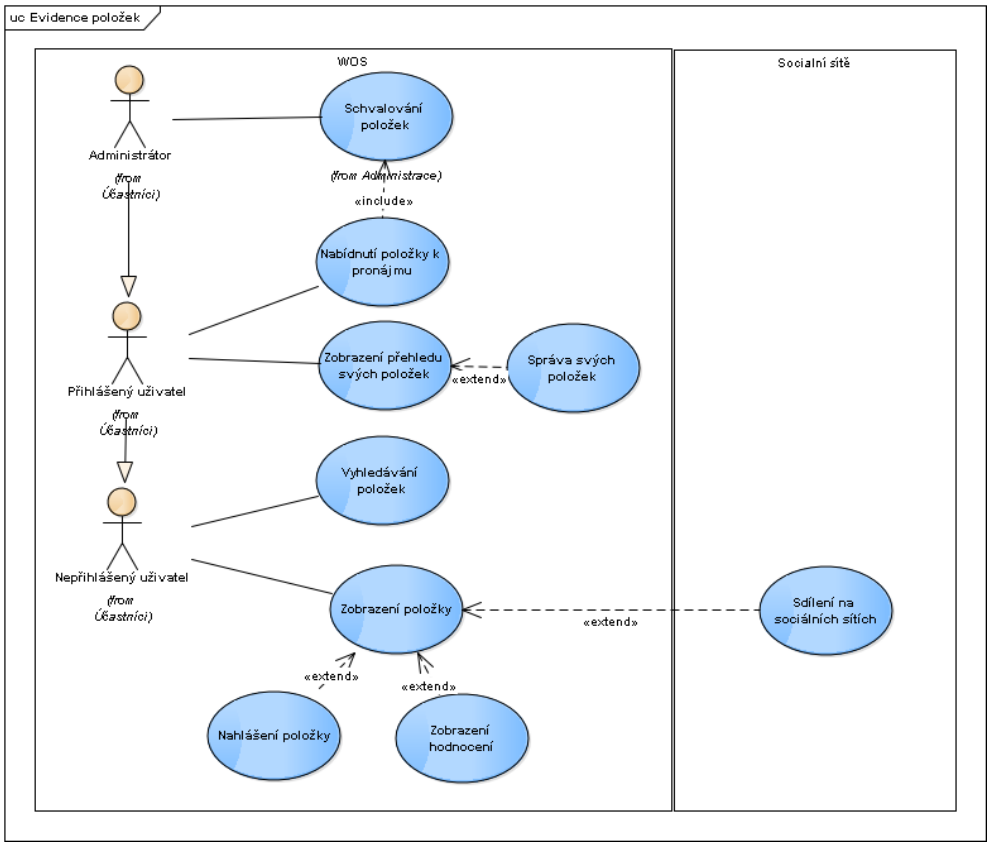
Uživatel může nabízet své položky k pronájmu. K vložení položky je třeba vyplnit formulář, ve kterém musí nahrát hlavní fotografii, vybrat kategorii, zadat název, popis, ceny a dostupnost, kdy je možné si danou věc vypůjčit. Dále také může zvolit klíčová slova, zvolit si, zda chce půjčovat více položek/instancí, nahrát soubory související s danou položkou a přidat služby, které lze poskytnout při vypůjčení.

1. Případ užití začíná, když chce uživatel na portálu nabízet svoji věc.
2. Uživatel vyplní formulář.
3. Systém uloží informace o položce do databáze.
4. Include (Schválení položky).
5. Zobrazení položky na stránce.

2.4.2.2 Zobrazení přehledu svých položek

Zobrazení přehledu položek umožňuje uživateli zobrazit jeho položky a dále s nimi pracovat.

1. Systém zobrazí seznam uživatelových položek.
2. Uživatel si zvolí jednu položku.



Obrázek 2.2: Evidence položek.

2.4.2.3 Správa svých položek

Správa položek obsahuje funkce upravení, skrytí a smazání položky.

- Upravení položky umožňuje uživateli měnit základní informace o dané položce.
- Smazání položky umožňuje uživateli smazat svou položku.
- Skrytí položky umožňuje uživateli skrýt svoji položku. Skrytá položka se nezobrazuje ostatním uživatelům a není možné si ji vypůjčit.

2.4.2.4 Zobrazení položky

Zobrazení položky zobrazí uživateli veškeré informace o dané položce. Mezi tyto informace patří: popis, lokality, kde si lze položku vypůjčit, časová dostupnost, hodnocení, podobné položky, mapa s vyznačenými lokalitami, fotka a jméno uživatele, který položku vložil, kdy byla položka vložena a soubory související s danou položkou. Po zobrazení položky je také možné položku

nahlásit například kvůli nevhodnému obsahu, zobrazit si všechna hodnocení nebo ji sdílet na socialních sítích.

1. Případ užití začíná, když si uživatel chce o položce zobrazit podrobné informace.
2. Systém zobrazí podrobný popis dané položky.

2.4.2.5 Nahlášení položky

Nahlášení položky slouží k upozornění administrátora na položku, která porušuje pravidla portálu, tj. jedná se například o spam nebo nevhodný obsah.

1. Případ užití začíná, pokud uživatel nalezl položku, která porušuje pravidla portálu, a rozhodl se ji nahlásit.
2. Systém zobrazí nahlašovací formulář, který nabízí možnosti: spam, nevhodný obsah nebo jiné s povinným popisem.
3. Uživatel formulář vyplní.
4. Systém uloží informace o nahlášení a odešle upozornění administrátorovi.

2.4.2.6 Zobrazení hodnocení

Zobrazení hodnocení slouží k zobrazení všech hodnocení dané položky. Hodnocení obsahuje popis, datum a dané hodnocení v podobě 1–5 hvězd.

2.4.2.7 Sdílení na sociálních sítích

Tato funkce umožňuje uživateli sdílet danou položku na sociální síti Facebook, Twitter nebo Google+.

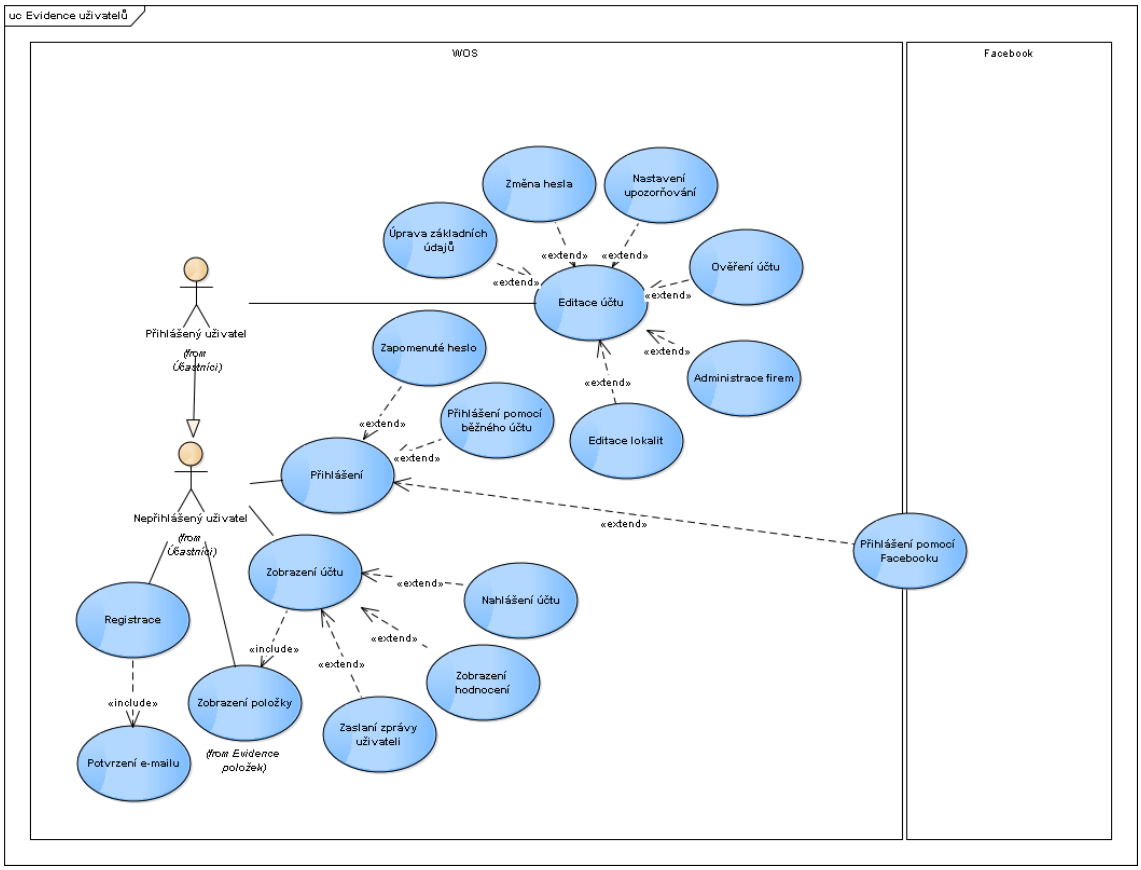
2.4.2.8 Vyhledávání položek

Vyhledávání položek umožňuje uživateli vyhledat specifickou položku.

1. Uživatel chce vyhledat specifickou položku.
2. Uživatel vyplní do vyhledávacího pole jméno nebo klíčové slovo dané položky.
3. Systém zobrazí seznam nalezených položek.

2.4.3 Evidence uživatelů

Diagram evidence uživatelů viz 2.3.



Obrázek 2.3: Evidence uživatelů.

2.4.3.1 Registrace

Registrace slouží k vytvoření nového účtu pomocí e-mailové adresy. Dalšími povinnými údaji jsou jméno, příjmení a heslo. K dokončení registrace je třeba ověřit zadanou e-mailovou adresu.

1. Případ užití začíná, pokud se uživatel rozhodne se zaregistrovat.
2. Systém zobrazí registrační formulář.
3. Uživatel vyplní své jméno, příjmení, e-mailovou adresu a heslo.
4. Pokud na zadanou e-mailovou adresu ještě není vázaný účet, systém vytvoří nový účet a odešle na zadanou e-mailovou adresu e-mail s odkazem k dokončení registrace. V opačném případě pokračuje 2. krokem tohoto scénáře.
5. Include (Potvrzení e-mailu).

2.4.3.2 Potvrzení e-mailu

Poté, co se uživatel zaregistruje pomocí e-mailové adresy, bude vyzván k potvrzení této adresy kliknutím na odkaz, který mu byl zaslán na zadanou e-mailovou adresu. Tím se aktivuje jeho účet a bude se moci přihlásit.

2.4.3.3 Přihlášení

Tato funkce nabízí uživateli dvě možnosti přihlášení: pomocí běžného účtu nebo přes Facebook.

2.4.3.4 Přihlášení pomocí běžného účtu

Tato funkce umožňuje uživateli se přihlásit pomocí své e-mailové adresy a hesla.

2.4.3.5 Přihlášení pomocí Facebooku

Tato funkce umožňuje uživateli se přihlásit pomocí svého Facebookového účtu. Pokud se uživatel přes Facebook přihlašuje poprvé, je mu rovnou vytvořen účet.

2.4.3.6 Zapomenuté heslo

Tato funkce slouží k možnosti resetovat zapomenuté heslo. Uživatel zadá svou e-mailovou adresu, na kterou je mu zaslán odkaz k obnově hesla.

1. Systém zobrazí formulář umožňující zadat e-mailovou adresu.
2. Uživatel vyplní svou e-mailovou adresu.
3. Systém na zadanou e-mailovou adresu pošle odkaz k obnově hesla.
4. Uživatel klikne na odkaz a je následně přesměrován na stránku určenou ke změně hesla.
5. Systém zobrazí fomulář umožňující zadat nové heslo.
6. Uživatel vyplní nové heslo.
7. Systém uloží nové heslo.

2.4.3.7 Zobrazení účtu

Zobrazení účtu umožňuje uživateli si zobrazit profil vybraného uživatele. Kromě jména, příjmení a e-mailové adresy si lze prohlédnout i nabízené položky. Dále je možné si zobrazit hodnocení úživatele, poslat mu zprávu na e-mail nebo nahlásit účet kvůli porušování pravidel portálu.

2.4.3.8 Nahlášení účtu

Nahlášení účtu slouží k upozornění administrátora na uživatele, který porušuje pravidla portálu, buď nabízením nevhodných položek, nebo jinak.

1. Případ užití začíná, pokud uživatel nalezne uživatele, který porušuje pravidla portálu, a rozhodl se ho nahlásit.
2. Systém zobrazí nahlašovací formulář, který nabízí možnosti: spam, nevhodný obsah nebo jiné s povinným popisem.
3. Uživatel formulář vyplní.
4. Systém uloží informace o nahlášení a odešle upozornění administrátorovi.

2.4.3.9 Zobrazení hodnocení

Tato funkce zobrazí všechna hodnocení, která daný uživatel dostal od ostatních uživatelů. Tato hodnocení se vážou k uskutečněným výpůjčkám položek, které uživatel nabízí. V jednotlivých hodnoceních je uvedeno, kdo a kdy hodnocení udělil, a samotné bodové a slovní ohodnocení.

2.4.3.10 Zaslání zprávy uživateli

Tato funkce umožňuje návštěvníkům portálu, například v případě dotazu, zaslat uživateli zprávu na e-mail. Nepřihlášení uživatelé musí ve formuláři vyplnit svoji e-mailovou adresu, na kterou chtějí, aby jim případně přišla odpověď.

2.4.3.11 Editace účtu

Editace účtu umožňuje uživateli spravovat svůj účet.

2.4.3.12 Úprava základních údajů

Tato funkce umožňuje uživateli upravit své jméno, příjmení, adresu a telefonní číslo.

2.4.3.13 Změna hesla

Tato funkce umožňuje uživateli si změnit své heslo.

2.4.3.14 Editace lokalit

Tato funkce umožňuje uživateli upravovat a přidávat lokality, ve kterých nabízí zapůjčení svých položek.

2.4.3.15 Ověření účtu

Ověření účtu umožňuje uživateli s vyplněným telefonním číslem ověřit svůj účet pomocí SMS zprávy. Tímto získá status „Ověřený uživatel“, což se projeví i na jeho profilové stránce.

1. Uživatel klikne na tlačítko „Ověřit účet pomocí SMS“.
2. Systém odešle na uživatelovo telefonní číslo SMS zprávu s unikátním kódem.
3. Systém vyzve uživatele, aby opsal kód z SMS zprávy.
4. Uživatel opíše kód.
5. Uživatel je ověřen a systém uloží změny.

2.4.3.16 Administrace firem

Administrace firem umožňuje uživateli spojit svůj účet se svými firmami a ty dále spravovat. Lze tedy přidat nebo odstranit firmu, dále je možné k firmě přidat nebo odebrat jiného uživatele jako prostředníka. K přidání firmy je nutné zadat její IČ, k přidání prostředníka jeho e-mailovou adresu.

2.4.3.17 Nastavení upozorňování

Tato funkce umožňuje uživateli nastavit si, na co, jak a popřípadě kdy chce být upozorněn. Způsob upozornění může být e-mailem nebo interními notifikacemi. Dále je možné si nastavit, jak dlouho dopředu chce uživatel být upozorněn na blížící se výpůjčku.

2.4.4 Evidence výpůjček

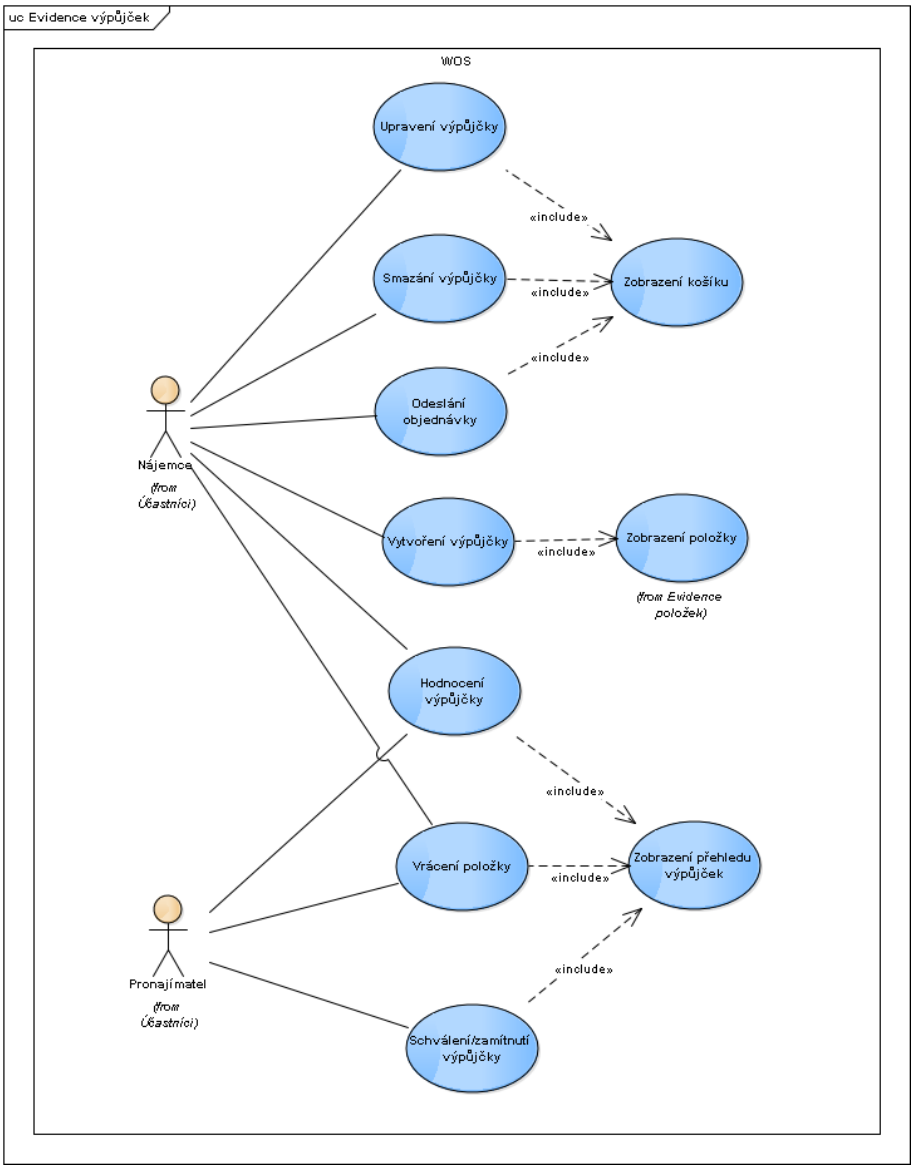
Diagram evidence výpůjček viz 2.4.

Oproti návrhu došlo k zásadním změnám. Původní diagram zobrazoval celý proces vypůjčení položky, nyní je zaměřený pouze na část spojenou s naším systémem. K velké změně také došlo díky rozdělení uživatele na nájemce a pronajímatele, což pomáhá k upřesnění kroků, které musí / může každá strana provést.

2.4.4.1 Vytvoření výpůjčky

Vytvoření výpůjčky umožňuje nájemci vytvořit výpůjčku na zvolené datum a s možným výběrem místa převzetí.

1. Případ užití začíná, jestliže si uživatel chce vypůjčit zvolenou položku.
2. Include (Zobrazení položky z Evidence položek).



Obrázek 2.4: Evidence výpůjček.

3. Systém zobrazí formulář umožňující zvolení místa převzetí položky a výběru data vypůjčení.
4. Systém vypočte cenu podle zvolených parametrů.
5. Uživatel potvrdí výpůjčku.
6. Systém výpůjčku uloží a přesměruje uživatele do košíku.

2.4.4.2 Zobrazení přehledu výpůjček

Zobrazení přehledu výpůjček umožňuje uživateli zobrazit výpůjčky i pronájmy.

1. Systém zobrazí seznam výpůjček.
2. Uživatel si zvolí jednu výpůjčku.

2.4.4.3 Hodnocení výpůjčky

Hodnocení výpůjčky umožňuje jak pronajímateli, tak i nájemci hodnotit provedenou výpůjčku. Hodnocení je jak bodové, tak i textové.

1. Případ užití začíná, jestliže chce uživatel ohodnotit proběhlou výpůjčku.
2. Include (Zobrazení přehledu výpůjček).
3. Systém zobrazí formulář pro bodové hodnocení a popis.
4. Uživatel povinně zvolí bodové hodnocení a volitelně může vyplnit popis.
5. Systém uloží informace o hodnocení.

Uživatel může hodnotit pouze ukončené výpůjčky. Každou výpůjčku může hodnotit pouze jednou.

2.4.4.4 Vrácení položky

Vrácení položky umožňuje nahlásit vrácení dané položky pronajímateli. Tuto akci může provést pronajímatel i nájemce.

1. Případ užití začíná, když byla položka vrácena a uživatel chce výpůjčku dokončit.
2. Include (Zobrazení přehledu výpůjček).
3. Uživatel potvrdí vrácení položky.
4. Systém uloží informace o výpůjčce.

2.4.4.5 Schválení/zamítnutí výpůjčky

Schválení/zamítnutí výpůjčky nabízí možnost pronajímateli zamítnout nebo schválit, a tím uskutečnit danou výpůjčku.

1. Případ užití začíná, když si nájemce objednal položku pronajímatele a pronajímatel chce výpůjčku schválit nebo zamítnout.
2. Include (Zobrazení přehledu výpůjček).
3. Pronajímatel si zvolí možnost potvrzení nebo zamítnutí výpůjčky.
4. Systém uloží informace o výpůjčce.

Schválit nebo zamítnout výpůjčku je možné pouze jednou a rozhodnutí nelze změnit.

2.4.4.6 Zobrazení košíku

Zobrazení košíku umožňuje zobrazit nájemcem vytvořené výpůjčky. Je zde také viditelná celková cena a košík obsahuje možnost zaplatit celou objednávku (všechny výpůjčky).

1. Systém zobrazí košík společně s celkovou cenou za objednávku.

2.4.4.7 Odeslání objednávky

Odeslání objednávky umožňuje nájemci odeslat a zaplatit výpůjčky, které si vložil do košíku.

1. Případ užití začíná, když si chce uživatel objednat jednu nebo více položek.
2. Include (Zobrazení košíku).
3. Uživatel si zvolí pokračovat.
4. Systém zobrazí simulaci platební brány.
5. Uživatel odsimuluje zaplacení objednávky.
6. Systém uloží informace o objednávce.

V košíku musí být vložená alespoň jedna výpůjčka.

2.4.4.8 Smazání výpůjčky

Smazání výpůjčky je možnost v košíku, která umožňuje nájemci smazat výpůjčku, kterou si vložil do košíku.

1. Případ užití začíná, když chce uživatel smazat nezaplacenou výpůjčku z košíku.
2. Include (Zobrazení košíku).
3. Uživatel si zvolí výpůjčku, kterou chce smazat.
4. Systém odstraní výpůjčku z košíku a nabídne uživateli možnost vrácení výpůjčky zpět.

2.4.4.9 Upravení výpůjčky

Upravení výpůjčky umožňuje upravit všechny složky výpůjčky.

1. Případ užití začíná, když chce uživatel upravit nezaplacenou výpůjčku.
2. Include (Zobrazení košíku).
3. Uživatel si zvolí výpůjčku, kterou chce upravit.
4. Scénář pokračuje 3. krokem scénáře Vytvoření výpůjčky.

2.4.5 Administrace

Diagram administrace viz 2.5.

Část administrace nebyla obsažena v původním návrhu.

2.4.5.1 Schvalování položek

Schválení položky je funkce, která slouží administrátorovi k filtraci nevhodného obsahu. Probíhá pouze při vkládání nové položky. Položky, které nejsou schváleny, nejsou zobrazeny na portálu.

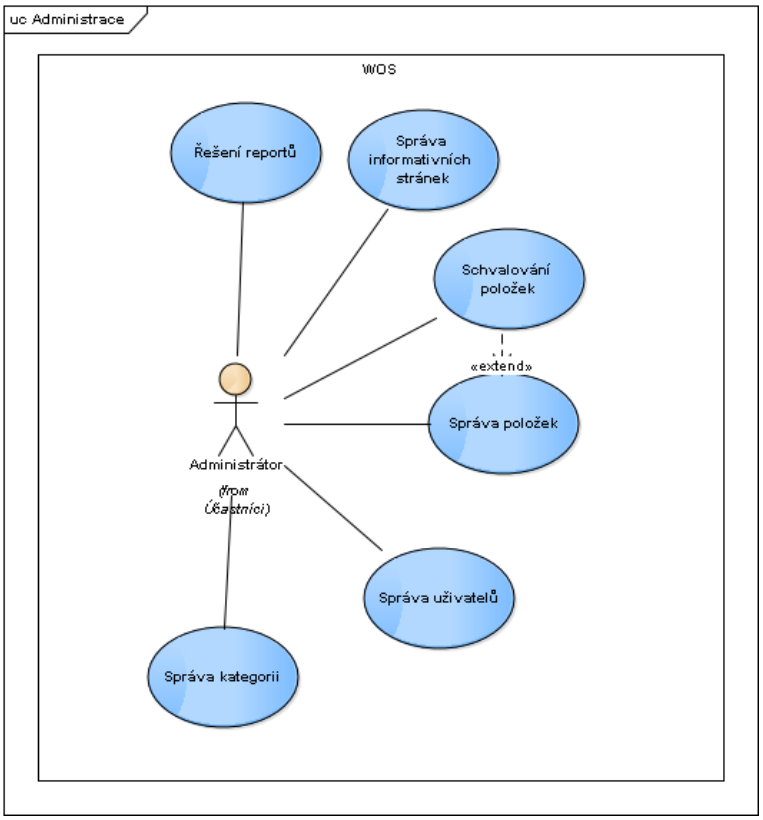
2.4.5.2 Správa informativních stránek

Správa informativních stránek umožňuje administrátorovi editovat a vytvářet nové statické stránky.

2.4.5.3 Správa kategorií

Správa kategorií slouží ke správě kategorií na stránce.

Administrátor může vytvářet nové podkategorie ke všem kategoriím, přidávat k nim atributy nebo již existující atributy upravovat.



Obrázek 2.5: Administrace systému.

2.4.5.4 Správa položek

Správa položek je soubor funkcí sloužících ke správě položek.

Administrátor může měnit základní informace o položce nebo může vybranou položku smazat.

2.4.5.5 Správa uživatelů

Správa uživatelů je soubor funkcí sloužících ke správě registrovaných uživatelů. Administrátor může uživatele skrýt (položky, které patří skrytému uživateli nejsou zobrazovány na stránce) nebo mu udělit administrátorská práva.

2.4.5.6 Řešení reportů

Administrátor může řešit reporty, které byly odeslány uživateli.

2.5 Diagramy aktivit

Diagram aktivit slouží k popisu průběhu funkcí. Používají se především k popisu funkcí, kde probíhá zároveň několik operací nebo kde se průběh dělí na několik větví vzhledem k rozhodnutí uživatele.

Tyto diagramy byly vytvořeny ve spolupráci s Martinem Židem a Nguyen Thi Tuyet Trang.

Rozhodli jsme se vytvořit diagram popisující přihlášení / registraci a průběh objednávky. Popisy zbylých funkcí jsou jednoduché a pro jejich pochopení postačí scénáře a popis u případu užití viz 2.4.

Diagramy vytvořené během návrhu lze nalézt v příloze.

2.5.1 Přihlášení / registrace

Diagram přihlášení / registrace viz 2.6.

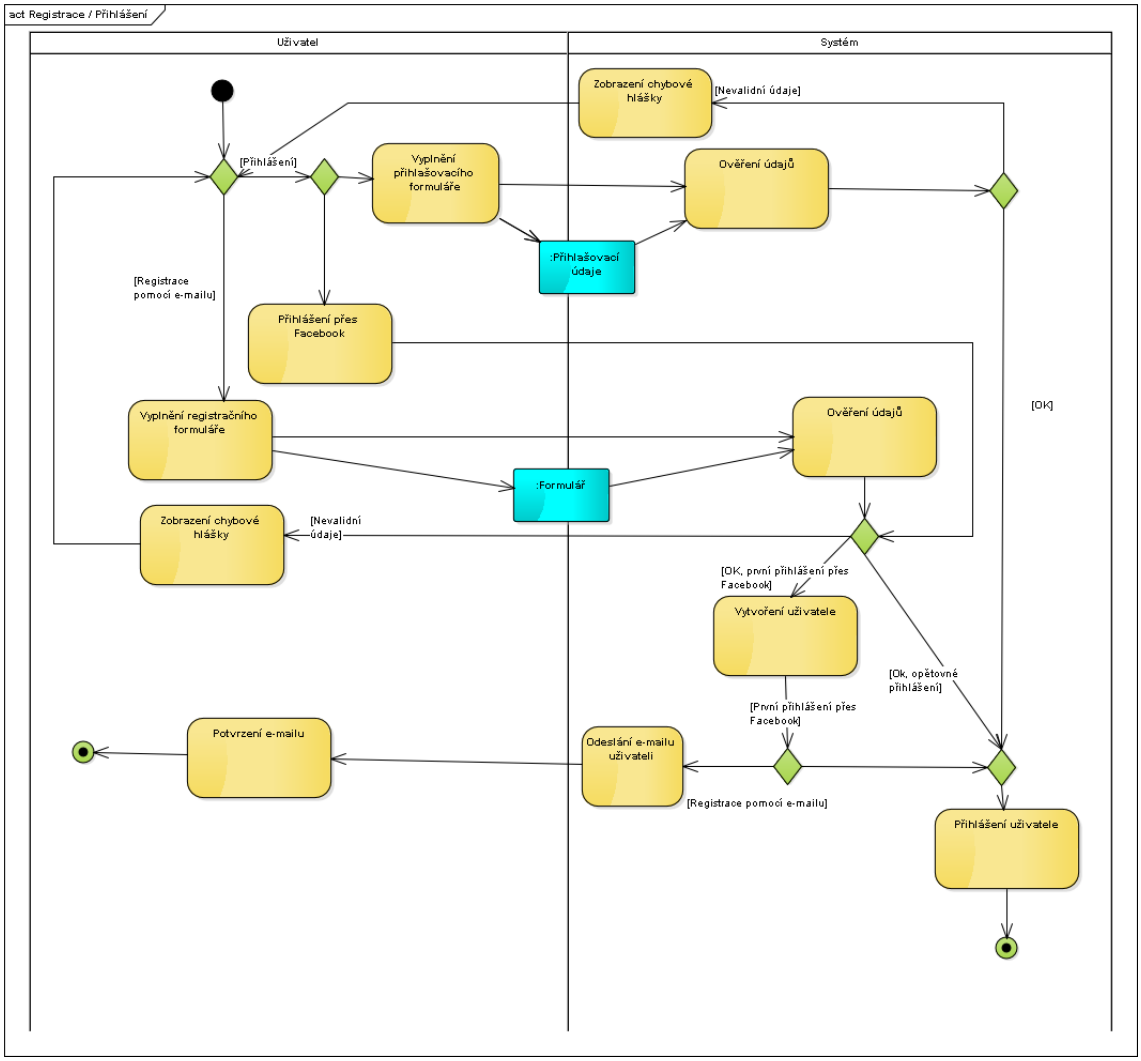
Diagram popisuje průběh přihlášení a registrace uživatele. Při přihlášení má uživatel dvě možnosti:

- Běžné přihlášení
  - uživatele pomocí účtu na našem portálu. Po vyplnění formuláře vyžadujícího přihlašovací e-mail a heslo se provede kontrola těchto údajů. Při validních údajích se uživatel přihlásí na svůj účet, v opačném případě je upozorněn chybovou hlášku na neplatné údaje a přesměrován na domovskou stránku.
- Přihlášení přes Facebook
  - umožňuje uživateli přihlásit se na našem portálu pomocí svého účtu na Facebooku. V případě nevalidních údajů je uživatel přesměrován na domovskou stránkou a upozorněn na neplatné přihlášení chybovou hláškou. Pokud jsou údaje validní rozlišujeme mezi prvním a opakovaným přihlášením. Pokud se uživatel přihlásil přes Facebook poprvé, je mu na serveru vytvořen uživatelský účet a poté je přihlášen k tomuto účtu. Pokud už se někdy přes Facebook přihlašoval, tedy už má účet založený, tak ho systém pouze přihlásí.

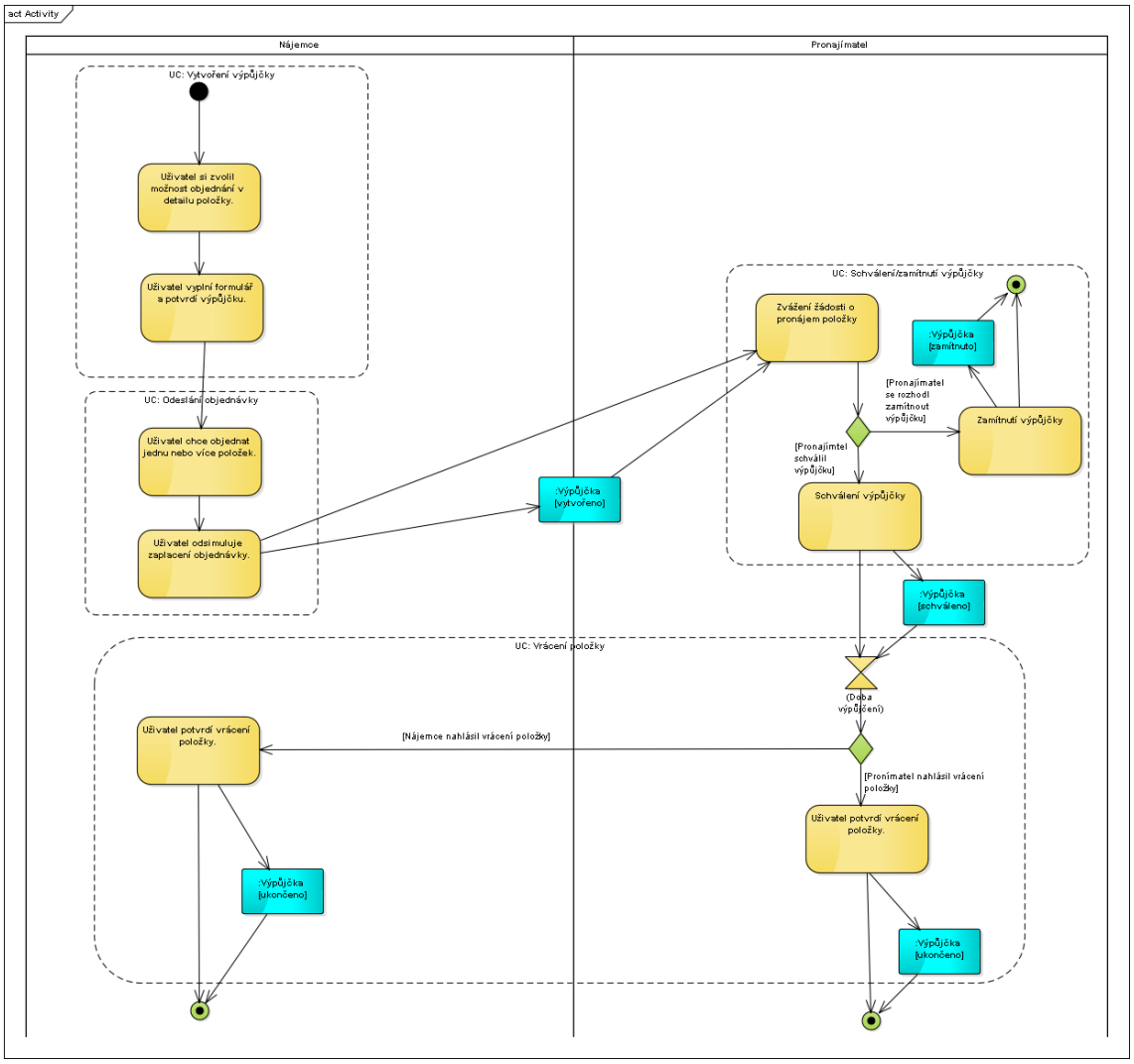
2.5.2 Vytvoření výpůjčky

Diagram vytvoření výpůjčky viz 2.7.

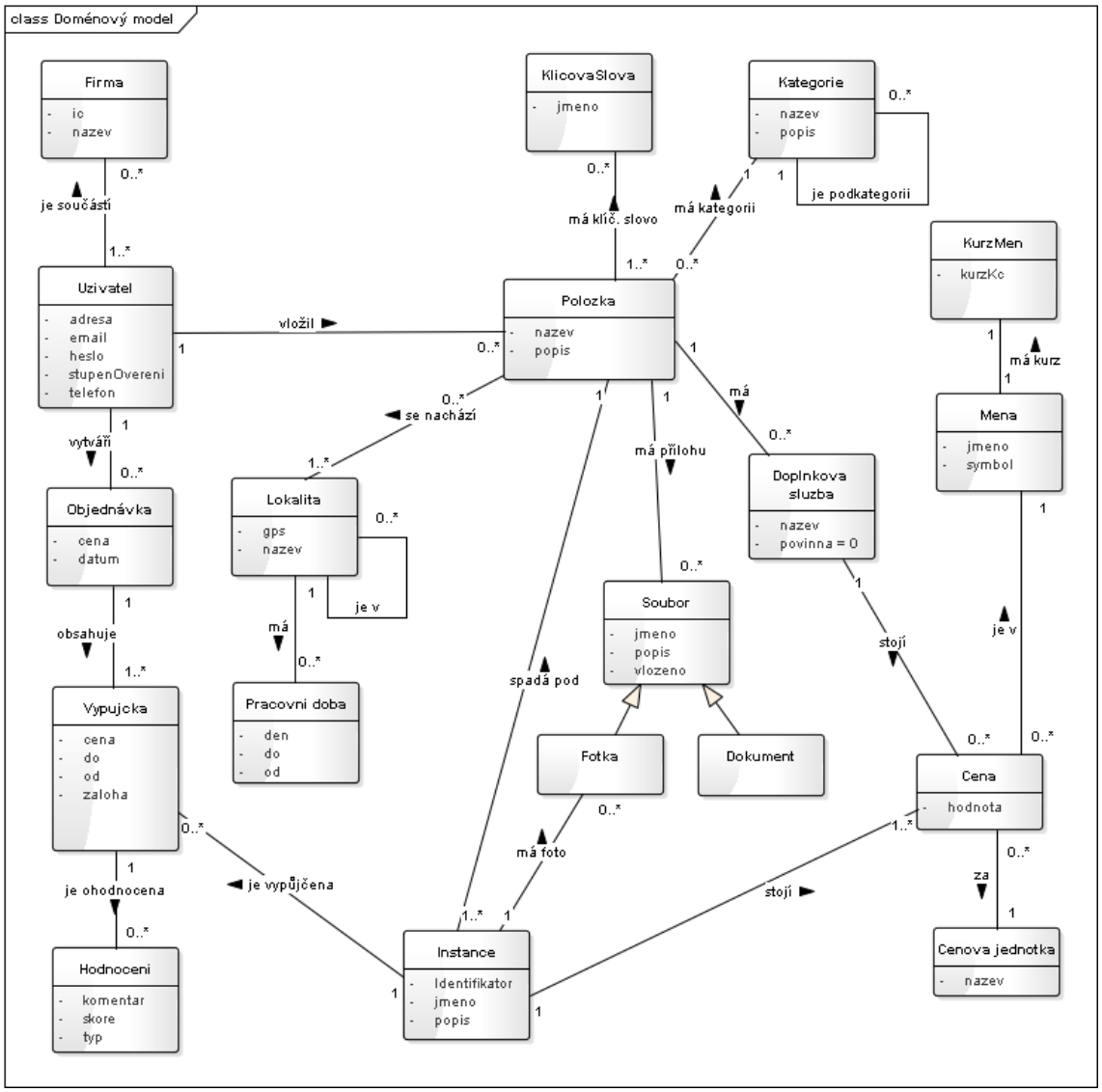
Diagram popisuje průběh vypůjčení položky jak z pohledu nájemce, tak pronajímatele. Jednotlivé aktivity jsou rozděleny do celků, náležících k případům užití popsaných v kapitole 2.4.



Obrázek 2.6: Diagram přihlášení / registrace.



Obrázek 2.7: Diagram vytvoření výpůjčky.



Obrázek 2.8: Doménový model.

2.6 Doménový model

Doménový model viz 2.8.

Doménový model popisuje základní logiku systému: jednotlivé entity, které v systému figurují, jejich atributy a vztahy mezi nimi.

Oproti návrhu došlo pouze k malým změnám, protože doménový model sloužil jako hlavní vzor při implementaci. Původní doménový model lze najít v příloze.

2.6.1 Uživatel

Uživatel je jedna z hlavních entit našeho systému. Uživatel vkládá na web svoje položky a může vytvářet objednávky.

2.6.2 Položka

Položka je základní entita. Jedná se o jakousi nehmotnou kategorii nad půjčovanou věcí. Např. pokud chce uživatel nabízet sekačku na trávu a má jich víc stejného typu, potom položkou bude sekačka na trávu daného typu a jednotlivé sekačky budou instance, které budou rozlišitelné například pomocí barvy nebo výrobního čísla. Díky tomuto postupu je možné rozlišovat mezi jednotlivými sekačkami a přiřazovat hodnocení ke správné věci.

2.6.3 Instance

Instance je přímo fyzická věc, kterou uživatel nabízí a kterou je možné si vypůjčit.

2.6.4 Firma

Uživatel má možnost spojit svůj účet se svojí firmou. Oproti návrhu není firma speciálním případem uživatele, ale samostatnou entitou.

2.6.5 Objednávka

Objednávka je entita, která přibyla oproti návrhu, jedná se o soubor výpůjček. Objednávku lze chápat jako obsah košíku.

2.6.6 Výpůjčka

Výpůjčka je záznamem o rezervaci jedné položky. K vypůjčení položky dojde po odeslání objednávky.

2.6.7 Klíčová slova

Klíčové slovo neboli tag je slovo nebo sousloví, které částečně identifikuje položku.

2.6.8 Kategorie

Kategorie složí k třídění položek do logických celků. Každá položka musí být součástí některé kategorie.

2.6.9 Soubor

Soubor může být dvojího typu: fotka, nebo soubor. Souborem může být například návod k použití nebo osvědčení o technickém stavu dané položky.

2.7 Diagram tříd

Diagram tříd je přesným obrazem současné implementace projektu. Obsahuje všechny třídy, jejich funkce a atributy.

Diagram tříd vytvořený při návrhu neodpovídá současnému stavu a je spíše obrazem doménového modelu doplněného o funkce. K tomuto došlo z důvodu neznalosti technologií použitých při implementaci.

Vzhledem k množství tříd v našem systému a s tím spojené velikosti daného diagramu neuvádím obrázek. Současný i původní diagram tříd lze nalézt v příloze.

Třídy v našem systému lze rozdělit na 3 typy:

1. modely,
2. továrny,
3. presentery.

2.7.1 Modely

Modely jsou v Nette hlavní třídou v backend části. Používají se k práci s databází, logickým výpočtům a dalším operacím s daty. Veškeré logické operace by měly probíhat v těchto třídách.

V našem systému jsou modely pojmenovány podle databázových tříd a jejich funkce slouží převážně k získávání a zpracování dat z těchto tříd nebo tříd s nimi spojených.

2.7.2 Továrny

Továrny slouží k tvorbě formulářů, které lze pak vložit do libovolné šablony a vyhnout se tak duplicitnímu kódu. Je zde možné vkládat do formuláře jednotlivé prvky (textová okna, radioboxy atd.), upravovat jejich vlastnosti a definovat funkce, které se zavolají po jeho odeslání.

2.7.3 Presentery

Presentery slouží k předávání dat získaných z modelů do šablon. V projektu jsou rozdělené do logických celků, a to na:

- Veřejné

  - pracují se šablonami ve veřejné části systému, to je části, do které má přístup i nepřihlášený uživatel.

- Uživatelské
  - pracují se šablonami přístupnými pouze přihlášenému uživateli.

- Administrace
  - pracují se šablonami přístupnými pouze administrátorovi systému.

2.8 Databázový model

Databázový model je vzhledem ke své velikosti pouze v příloze. Jedná se o běžnou MySQL databázi, proto zde zmíním pouze několik věcí o tabulkách, které jsou zajímavé.

Databázový model vytvořený při návrhu neodpovídá současně používané databázi, jelikož během implementace vznikaly nové nároky na ukládaná data.

Původní databázový model lze najít v příloze.

2.8.1 Kategorie

Hierarchii kategorií v databázi udržujeme pomocí tzv. nested set modelu. Tato metoda je založena na dvou parametrech: levý a pravý. Jedná se o číselné parametry a platí, že levý je vždy menší než pravý. Tato dvě čísla lze chápat jako interval, a pokud obě leží v intervalu jiné kategorie, jedná se o její podkategorii. V naší databázi si k těmto hodnotám ukládáme ještě hloubku dané kategorie, což značně usnadňuje hledání přímých podkategorií.

2.8.2 Kopie

Tyto tabulky vznikly kvůli potřebě uchovávat informace spojené s položkou v době jejího vypůjčení. Při každé výpůjčce se vytvoří kopie všech tabulek, které úzce souvisí s půjčovanou položkou, a to z důvodu možnosti dohledání potřebných informací v případě nesrovnalostí s danou výpůjčkou. Tímto lze zamezit podvodům, kdy by uživatel po potvrzení výpůjčky například zvýšil cenu a tvrdil, že tato cena byla platná již v době jejího potvrzení.

2.8.3 Nepoužívané tabulky

Tyto tabulky byly vytvořeny při návrhu systému, ale funkce s nimi spojené nebyly implementovány nebo se ukázaly jako nedůležité.

- Globální nastavení

  - Tato tabulka měla sloužit k jednoduché úpravě nastavení celé stránky. Například změně vzhledu nebo aktivaci slev na zvýhodněné položky.

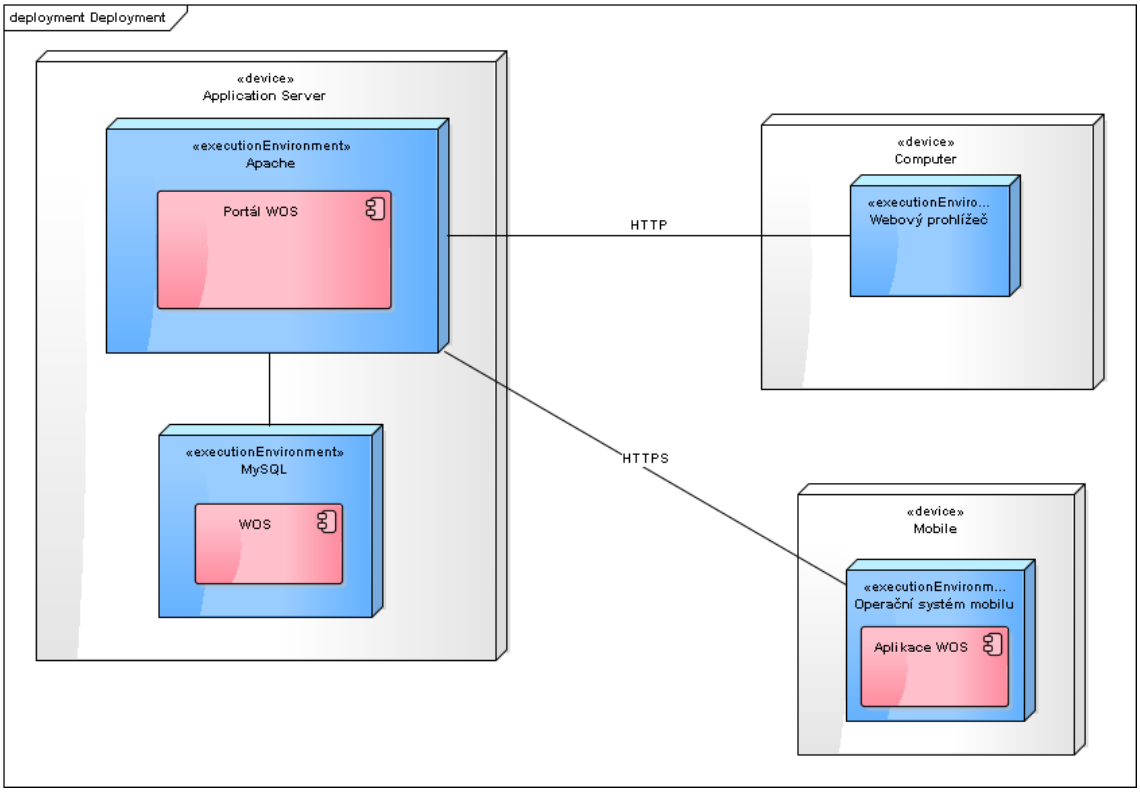
- Token
  - Tato tabulka měla sloužit k ukládání tokenů ze stránek jako je Facebook nebo MojeID. Od propojení se službou MojeID se nakonec odstoupilo a k přihlášení přes Facebook není ukládání tokenu nutné.

- Zvýhodnění
  - V tomto případě se jedná o soubor tabulek, které měly sloužit k ukládání informace o zvýhodnění, které si uživatele zakoupil. Zvýhodnění mohlo být spojeno s položkou nebo uživatelem. Zvýhodněné položky by se zobrazovaly na stránce nad normálními položkami a měly větší fotku. Zvýhodněný uživatel by mohl například přidávat více fotek ke svým položkám. Tyto funkce však nebyly implementovány.

2.9 Diagram nasazení

Diagram nasazení viz 2.9.

Diagram nasazení popisuje fyzickou strukturu systému a software používaný jednotlivými komponentami. U našeho systému se jedná o jednoduchou strukturu, kde na jednom serveru běží serverová část a MySQL databáze. K tomuto serveru se uživatelé připojují za pomocí webového prohlížeče přes protokol HTTP. Dále je zde také znázorněno připojení mobilní aplikace.



Obrázek 2.9: Diagram nasazení.

Kapitola 3

API

API (zkratka pro Application Programming Interface) označuje v informatice rozhraní pro programování aplikací. Tento termín používá softwarové inženýrství. Jde o sbírku procedur, funkcí, tříd či protokolů nějaké knihovny (ale třeba i jiného programu nebo jádra operačního systému), které může programátor využívat. API určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu. [4]

Já se v této práci zaměřil na vytvoření API pro připojení externích aplikací. Jelikož se jedná o webové API, probíhají jednotlivé dotazy pomocí HTTP a odpovědi systém vrací v JSON formátu.

Nejdříve se podíváme na nároky na tuto API, jak je to se zabezpečením a poté na strukturu jednotlivých dotazů.

3.1 Nároky na API

3.1.1 Obecné nároky

Hlavním cílem tohoto API je možnost připojení externích aplikací k našemu systému. Vzhledem k povaze systému budou tyto aplikace nejspíše vyžadovat získání dat, které se v systému vyskytují. Proto bylo třeba vytvořit funkce, které tyto informace získají a vrátí je v lehce zpracovatelném formátu.

3.1.2 Mobilní aplikace

Martin Žid ve své bakalářské práci vytváří mobilní aplikaci, která bude umožňovat připojení k našemu systému.

V aplikaci není použitý princip položka / instance (viz 2.6), ale jsou zobrazovány pouze instance. Díky této změně bylo třeba vytvořit funkce, které pracují s instancemi jako s položkami, a vracet tato data ve formátu používaném mobilní aplikací.

3.1.3 Rozšiřitelnost API

API by měla být snadno rozšiřitelná v případě vzniku nových nároků ze strany uživatelů. Proto je API realizovaná objektově a jeden dotaz zpracovává jedna funkce.

3.2 Zabezpečení

3.2.1 Metody zabezpečení

3.2.1.1 Základní autentizace

Jak už název napovídá, jedná se o nejjednodušší formu autentizace. Vše potřebné k implementaci základní autentizace lze většinou najít v použitém frameworku nebo v knihovnách použitého jazyka. Problémem základní autentizace je, že poskytuje minimální ochranu. Nejsou zde žádné rozšířené možnosti použití této metody, takže pouze odesíláte jméno a heslo zakódované pomocí Base64. Tato metoda by nikdy neměla být používána bez TLS (dříve známé jako SSL) šifrování, protože jinak může být jméno a heslo snadno dekódováno. [5]

3.2.1.2 OAuth 1.0a

OAuth 1.0a je nejbezpečnější ze zmiňovaných protokolů. Protokol využívá kryptografický podpis (většinou HMAC-SHA1), to je hodnotu, která kombinuje tajný token, nonce (náhodné číslo) a další základní informace spojené s dotazem. Velkou výhodou OAuth 1 je, že nikdy přímo nepředáváte po síti tajný token, což naprosto vylučuje možnost vidět heslo během přenosu. Toto je jediný ze zmiňovaných protokolů, který lze používat bez SSL. Pro použití tohoto protokolu je třeba využít specifických hashovacích algoritmů s přesně definovanými kroky. To však již často nebývá problém, jelikož většina programovacích jazyků obsahuje knihovny, které to udělají za vás. [5]

3.2.1.3 OAuth 2

OAth 2 je nástupcem OAuth 1, který se však vydal naprosto jinou cestou. Současná specifikace nepoužívá podpisy, proto již není nutné používat kryptografické algoritmy na jejich tvorbu a kontrolu. O veškeré šifrování se stará TLS, které se stalo nutností. Díky snížené bezpečnosti (žádný digitální podpis znamená, že není možné zjistit, zda byl obsah zfalšovaný před nebo po odeslání) doporučujeme používat OAuth 1. OAuth 2 lze použít v systémech, které neobsahují citlivá data. [5]

3.2.2 Zvolená metoda

Pro zabezpečení komunikace je použita upravená verze základní autentizace. Přihlašovací údaje jsou odesílány v hlavičce dotazu a zakódovány pomocí Base64. Po přihlášení se uživateli odešle API key (náhodný řetězec), který slouží jako identifikátor, používaný k ověření, zda má uživatel přistup k požadovaným datům. Po nasazení portálu do provozu bude nutné ke komunikaci použít protokol HTTPS namísto nešifrovaného HTTP, čímž se zajistí bezpečnost odesílaných dat.

3.3 Struktura dotazů

Přesný popis dotazů obsahující vstupní a výstupní parametry je přiložen v příloze. Zde je pouze stručný popis.

Položky zmiňované v této části jsou z pohledu webového portálu instancemi viz 3.1.2.

3.3.1 Všechny položky

Vrátí všechny položky, které jsou na stránce nabízené k pronájmu.

3.3.2 Detail položky

Vrátí detailní informace o vybrané položce. Kromě informací o položce také vrací informace o uživateli, který ji nabízí, hodnocení, lokality, kde je možné položku vypůjčit, a podobné položky.

3.3.3 Přihlášení

Přihlásí uživatele mobilní aplikace. Přihlašovací údaje (e-mail a heslo) jsou odesílány v hlavičce dotazu, zakódované pomocí Base64. Odesílá „api-key“, který slouží k ověření uživatele.

3.3.4 Registrace

Vytvoří nový účet. Odešle uživateli e-mail s odkazem na potvrzení registrace.

3.3.5 Vyhledávání

Vrací položky, které odpovídají hledanému výrazu.

3.3.6 Výpůjčky

Vrací informace o všech výpůjčkách vybraného uživatele. Obsahuje informace o položkách, které si on vypůjčil, i o těch, které si uživatelé půjčili od něj.

3.3.7 Profil uživatele

Vrací všechny informace potřebné k zobrazení profilu uživatele, včetně jeho položek.

3.3.8 Uživatelské notifikace

Vrací veškeré notifikace daného uživatele.

3.3.9 Zapomenuté heslo

Odešle na e-mail uživatele odkaz na stránku, na které si může změnit heslo ke svému účtu.

3.3.10 Lokality uživatele

Vrací všechny lokality spojené s vybraným uživatelem.

3.3.11 Přidání nové položky

Vloží novou položku do systému.

3.3.12 Schválení, zamítnutí a vrácení výpůjčky

Tyto tři funkce umožňují změnit stav výpůjčky na schválená, zamítnutá, nebo vrácená.

3.3.13 Hodnocení výpůjčky

Slouží k ohodnocení vybrané výpůjčky uživatelem.

3.3.14 Podkategorie vybrané kategorie

Vrací přímé podkategorie vybrané kategorie.

Kapitola 4

Refaktor kódu

V této kapitole se podíváme na refaktor kódu. Nejdříve zmíním, co to vlastně je a k čemu je to dobré. Poté co bylo hlavním cílem mé práce vzhledem k refaktoru kódu, na co jsem se zaměřil, a na některé příklady provedených změn.

4.1 Co je refaktor kódu

Refaktor je úprava kódu, která nepřidává žádnou novou funkčnost nebo optimalizuje kód z hlediska výkonu. Jedná se pouze o úpravy přispívající k přehlednosti a čitelnosti kódu. Mezi hlavní činnosti patří: odstranění duplicitního kódu, formátování, udržení konzistence, rozdělení kódu na logické celky, odstranění nepoužívaných částí kódu, popis kódu nebo logické pojmenování funkcí a parametrů.

4.2 Proč refaktorovat kód

Hlavním důvodem, proč kód refaktorovat, je usnadnění práce lidem, kteří s ním pracují. Čitelný kód je přívětivější pro nově příchozí programátory a je mnohem snazší se v něm orientovat a nalézt chyby.

4.3 Realizace

Většina z principů zmiňovaných výše již byla splněna díky dodržování programátorských konvencí během implementace. Při refaktoru jsem se tedy zaměřil na odstranění zbylých problémů, a to především na:

- odstranění nepoužívaných tříd a funkcí,
- odstranění duplicitního kódu,
- popis používaných funkcí,

- konzistenci kódu.

Veškeré změny lze dohledat v repozitáři na Redminu použitém při vývoji viz 5.2.1.

4.3.1 Odstranění nepoužívaných tříd a funkcí

4.3.1.1 Třídy

Během implementace vzniklo několik tříd vytvořených v souvislosti s návrhem systému, které se však během implementace nepoužily. Odstraněné třídy: AttributeCopy, AttributeType, Currency, GlobalSettings, ItemTag, Keyword, PriceUnit, AddItemFactory.

4.3.1.2 Funkce

Z kódu jsem odstranil funkce, které se nepoužívaly. Tyto funkce vznikly především jako funkce duplicitní, protože si někdo nevšiml, že daná funkce již byla implementována, a vytvořil si vlastní. Dalším typem byly funkce nahrazené optimalizovanými verzemi, které byly ponechány v případě problému s těmi novými.

4.3.2 Odstranění duplicitního kódu

Většina duplicitního kódu se skládala z duplicitních funkcí zmíněných v předchozí části. Dále by se za duplicitní daly považovat části MySQL dotazů nad stejnými tabulkami, které by bylo možné rozdělit na několik částí, ale to by dle mého názoru vedlo spíše ke zhoršení čitelnosti kódu než ke zlepšení, proto jsem se rozhodl nechat je v této podobě. Výjimkou byla pouze funkce *getCompaniesOfUser* ve třídě *Company*, která rozlišovala mezi dvěma případy a lišila se pouze v jednom řádku, proto jsem ji upravil do tvaru:

```
$users = $this->db->select("uzivatel.id_uzivatel,
    uzivatel.jmeno, uzivatel.prijmeni, uzivatel.jmeno,
    uzivatel.email")
                ->from(Tables::FIRMA)
                ->join(Tables::FIRMA_MA_UZIVATEL)
                ->on("firma.id_firma=firma_ma_uzivatel.
                    id_firma")
                ->naturalJoin(Tables::ROLE)
                ->join(Tables::UZIVATEL)
                ->on("uzivatel.id_uzivatel=
                    firma_ma_uzivatel.id_uzivatel")
                ->where("firma.id_firma = ?", $companyId);

if($roleId) $users->and("role.id_role = ?", $roleId);
```

```
$users = $users->fetchAll();
```

Další duplicitně vypadající kód lze nalézt v modelech *Item* a *Instance*. To je způsobeno velice podobnou strukturou těchto entit a funkcemi používanými v API, kvůli přístupu vysvětlenému v 3.1.2.

4.3.3 Popis používaných funkcí

U všech funkcí, u kterých chyběl popis, jsem jej doplnil. Těchto funkcí bylo mnoho, proto je zde nebudu uvádět.

4.3.4 Konzistence kódu

V rámci zachování konzistence kódu jsem databázové dotazy realizované pomocí příkazu *query* přepsal do formátu používaného ve většině kódu, to je do formátu využívajícího funkce poskytované dibi. Dibi je abstraktní databáze v jazyce PHP, kterou využívá framework Nette použitý v našem systému. Mezi změněné funkce patří například *deleteCompany*, *removeConnection* nebo *addLocalityForItem*.

Příklad změněného kódu:

```
$this->db->query(
'DELETE FROM '.Tables::
    FIRMA_MA_UZIVATEL.
' WHERE id_firma=' .$companyId.
' AND id_uzivatel=' .$userId.
' AND id_role=' .$roleId.';'
);
```

změněno na:

```
$this->db->delete(Tables::
    FIRMA_MA_UZIVATEL)
->where("id_firma = ?", $companyId)
->and("id_uzivatel = ?", $userId)
->and("id_role = ?", $roleId)
->execute();
```

Některé funkce bylo třeba ponechat v query formátu, protože složitější dotazy není možné napsat pomocí druhého postupu.

4.3.5 Ostatní změny

Mezi další minoritní změny patřily opravy překlepů, odstranění zakomentovaných částí kódu, změny názvů některých proměnných a funkcí.

Kapitola 5

Vývojářské prostředí

V této kapitole se budu zabývat nástroji, které pomohou případným vývojářům pokračujícím na našem projektu. Mezi tyto nástroje patří:

- verzovací nástroj,
- tiketovací nástroj,
- deployment,
- testovací prostředí.

5.1 Verzovací nástroj

Verzovací nástroj se používá k ukládání postupu na projektu a řeší problém, kdy se kvůli nějaké chybě stane projekt nefunkčním. Pokud se toto stane, máme možnost se vrátit zpět na funkční verzi projektu a bez přerušení či dlouhého odstraňování chyby dále pokračovat ve vývoji. Další výhodou je tvorba tzv. větví, které umožňují jednotlivým členům týmu pracovat na svém úkolu, aniž by jim někdo další zasahoval do jejich kódu. Tyto větve se poté dají spojit dohromady, a vytvořit tak jeden funkční celek. Mezi verzovací systémy patří například: GIT, GITHUB nebo Subversion.

5.1.1 Připravený nástroj

Jako verzovací nástroj jsem se rozhodl použít GIT, který jsme využívali během implementace. Popis, jak se k tomuto GITu připojit a jak s ním pracovat, naleznete v 5.5.

5.2 Tiketovací nástroj

Tiketovací nástroj umožňuje vytváření a správu tiketů. Tyto tikety mohou být různého typu. Mohou například nahlašovat objevený problém v projektu nebo

popisovat úkol, který je třeba splnit. Tikety mohou mít různé parametry, mezi které patří například: priorita, odhadovaný čas řešení, stav, doba, do které musí být tiket vyřešen, komu je problém přiřazen a tak dále.

Tento nástroj velice usnadňuje přiřazování úkolů a kontrolu průběhu práce na projektu. Snadno lze zjistit, které věci jsou hotové a které je nutné vyřešit. Také je možné sledovat aktivitu jednotlivých členů díky zapisování práce strávené na jednotlivých tiketech.

Mezi nepoužívanější nástroje patří například Redmine nebo Bugzilla.

5.2.1 Připravený nástroj

Stejně jako v případě verzovacího nástroje jsem se rozhodl použít již existující nástroj použitý při implementaci. Jedná se o Redmine a hlavním důvodem k této volbě bylo množství informací, které se na tomto Redminu nacházejí a mohly by být velkým přínosem při budoucí implementaci.

Jak se na tento Redmine připojit naleznete v 5.5.

5.3 Deployment

Deployment je nasazení aplikace do určitého prostředí. V našem projektu se deployment provádí při pushnutí do větve dev nebo master. Při deploymentu se využívá funkcionality GITu, tzv. hooků.

Autorem popisovaného deploymentu je Michal Kváček.

5.3.1 Hooky

GIT hooky jsou scripty, které GIT provádí před nebo po událostech jako jsou: *commit*, *push* a *receive*. [6]

V našem případě se používají dva typy hooků: pre-receive, který se volá po pushnutí, ale před začátkem provádění změn v repozitáři, a post-receive, který se volá po pushnutí včetně provedení změn v repozitáři.

5.3.2 Deployment větve dev

Deployment do větve dev je jednoduchý. Pre-receive hook vytvoří vývojové prostředí a inicializuje vývojové databáze. Post-receive hook nahraje změny do souborů a provede migrace databáze. Jednoduše by se dalo říct, že tato větev pouze obsahuje aktuální stav GIT repozitáře.

5.3.3 Deployment větve master

Vzhledem k tomu, že větev master se nahrává na produkční server, kde by měla být poslední stabilní verze projektu, je deployment složitější, aby nedošlo k nahrání nefunkčního systému.

5.3.3.1 Pre-receive

Postup při pre-receive hooku.

1. Vytvoření testovacího prostředí, pokud neexistuje.
2. Vytvoření produkčního prostředí, pokud neexistuje.
3. Vytvoření zálohy produkčních dat.
4. Vytvoření zálohy produkční databáze.

Pokud některá z částí skončí chybou, je deployment zastaven.

5.3.3.2 Post-receive

Postup při post-receive hooku.

- Testovací prostředí
  1. Nahrání nových dat.
  2. Migrace databáze.
  3. Spuštění testů.
- Produkční prostředí
  1. Zapnutí údržby portálu.
  2. Nahrání nových dat.
     a) Při neúspěchu – rollback dat.
  3. Migrace databáze.
     a) Při neúspěchu – rollback databáze a dat.
  4. Vypnutí údržby portálu.
  5. Deployment úspěšně dokončen.

Nejdříve se provede testovací deployment. Proběhne-li vše v pořádku, začne skutečný deployment produkčního prostředí. Pokud některá z dílčích částí skončí neúspěchem, končí celý deployment neúspěšně. Je-li neúspěšný některý z rollbacků, zůstane portál v režimu údržby. Údržba portálu pouze vypisuje na stránce hlášku o probíhající údržbě.

5.4 Testovací prostředí

Testovací prostředí je, jak už název napovídá, prostředí, ve kterém probíhá testování systému. V praxi to většinou bývá operační systém našeho zařízení. To však vede k problémům, protože v různých prostředích se může systém chovat odlišně. Tomuto problému se můžeme vyvarovat, pokud každý bude pracovat v naprosto totožném prostředí. Toho můžeme docílit například použitím virtuálního testovací prostředí, což je přívětivější varianta, než nutit programátory mít na svém zařízení to samé co ostatní. Jedním z nástrojů, který umožňuje tvorbu virtuálního prostředí, je Vagrant, využitý v mé práci.

5.4.1 Připravené prostředí

Vagrant umožňuje volbu z mnoha obrazů operačních systémů, kterým říká boxy. Box může být základní nebo již upravená verze operačního systému. Pro tvorbu testovacího prostředí jsem se rozhodl použít základní verzi 64bitového operačního systému Ubuntu 14.04. Na tento systém se při sestavení virtuálního obrazu nainstalují nástroje:

- Apache2,
- PHP 7.0,
- MySQL 5.6,
- phpunit.

Po této instalaci probíhá konfigurace Apache2 tak, aby fungoval s naším systémem. V poslední řadě se vytvoří databáze podle současného obrazu. K udržení aktuálnosti databáze slouží nástroj Phinx (bližší informace viz 5.5).

Jak pracovat s tímto prostředím naleznete v 5.5.

5.5 Programátorská příručka

V této sekci najdete veškeré informace spojené s přípravou a ovládáním připravených nástrojů.

5.5.1 GIT

5.5.1.1 Připojení

K práci s GITem budeme potřebovat mít nainstalovaný GIT (lze stáhnout například z `https://git-scm.com/`). Také doporučuji použít program PHPStorm, který budeme využívat i při práci s testovacím prostředí.

Dále budeme potřebovat k připojení k repozitáři SSH klíč. Ten můžeme na linuxových systémech vygenerovat jednoduše pomocí příkazu ssh-keygen.

Na systému Windows můžeme použít program PuTTYgen, který je volně ke stažení. Po spuštění programu vyberte v dolní části typ klíče SSH-2 DSA, poté klikněte na tlačítko GENERATE a náhodně hýbejte myší přes volnou oblast. Jakmile program dokončí generování klíče, zvolte si heslo (passphrase) a uložte oba klíče do `C:\Users\username\.ssh`. Veřejný klíč pojmenujte jako *id_rsa.pub* a privátní *id_rsa*.

Pokud máte klíče vygenerované, odešlete veřejný klíč na michal@binarity.cz s prosbou o přístup do repozitáře s projektem WOS a vyčkejte na potvrzení k přístupu.

Jakmile dostanete potvrzení k přístupu, otevřete program PHPStorm. Pro stažení repozitáře jděte do *VCS->Checkout from Version Control->GIT*. Do kolonky *Git Repository URL* vložte `binarity.cz:/opt/git/fit_cvut/wosnpz2015.git`. *Parent Directory* je cesta, kam chcete projekt stáhnout, a *Directory Name* název složky, do které se stáhne. Stiskněte tlačítko clone a budete požádáni o vyplnění hesla zvoleného při generování klíčů. Nyní máte projekt stažený ve vybrané složce.

5.5.1.2 Práce s GITem

Mezi základní a nejpoužívanější operace patří *Commit*, *Push* a *Pull*.

- Commit
  - slouží k lokálnímu uložení provedených změn. Při každém commitu se uvádí krátká zpráva popisující provedené změny. Jednotlivé commity lze dohledat v repozitáři a tím zjistit, jaké změny kdo provedl.
- Push
  - ukládá provedené lokální změny (uložené pomocí příkazu commit) do repozitáře tak, že si je můžou stáhnout ostatní pomocí příkazu pull.
- Pull
  - slouží ke stažení změn oproti lokální větvi. Měl by se použít před začátkem každé práce, abychom měli projekt aktuální.

Všechny tyto příkazy naleznete v programu PHPStorm v záložce *VCS->GIT*.

Další důležitou věcí při práci s GITem jsou větve. V současné době existují v připraveném repozitáři dvě hlavní větve: dev a master. Dev větev je používaná pro vývoj (dostupná na `http://sp2.binarity-testing.cz/`) a master je větev, na které běží poslední stabilní verze (dostupná na `http://www.bagrujemeshunkou.cz/`). Do master větve může pushnout pouze člověk, který k tomu má oprávnění (v současné době Michal Kváček).

Pokud pracujete na nové funkcionalitě, doporučuje se vytvořit si vlastní větev. To uděláme v PHPStormu tak, že v pravém dolním rohu klikneme na

Git a zvolíme možnost *New Branch*. Mezi větvemi lze libovolně přecházet po kliknutí na danou větev a vybrání možnosti *Checkout*. Chceme-li spojit dvě větve (například dev a naší větev) přepneme se do větve dev pomocí *Checkout* a poté u naší větve zvolíme možnost *Merge*. Všechny změny z naší větve jsou nyní ve větvi dev.

Toto jsou ve zkratce základní informace potřebné při práci s GITem. Pro další informace doporučuji použít stránky GITu a programu PHPStorm.

5.5.2 Redmine

Pro přístup do připraveného Redminu pošlete e-mail na michal@binarity.cz s prosbou o vytvoření účtu a přístupu k projektu WOS.

Redmine naleznete na stránce `https://io.binarity.cz/`. Vzhledem k tomu, že je daný Redmine v češtině, je práce s ním velice intuitivní, proto nepovažuji za nutné zde vypisovat základní operace při jeho používání. Podrobné informace o nástroji Redmine naleznete na `http://www.redmine.org/`.

5.5.3 Testovací prostředí

5.5.3.1 Instalace

Pro používání testovacího prostředí budete potřebovat programy Vagrant, VirtualBox a již dříve zmiňovaný PHPStorm.

Všechny ostatní soubory jsou již přiložené v repozitáři projektu WOS. Přístup k repozitáři a jeho stažení viz 5.5.1.1.

5.5.3.2 Práce s testovacím prostředím

Pro spuštění testovacího prostředí spusťte program PHPStorm a v záložce *Tools->Vagrant* zvolte příkaz *Up*. Vyčkejte, než se prostředí inicializuje[1].

Po dokončení inicializace běží váš server na IP adrese 192.168.33.101, na kterou můžete přistoupit přes váš webový prohlížeč.

K serveru se lze připojit pomocí SSH. V PHPStromu v záložce *Tools* zvolte možnost *Start SSH session* a vyberte Vagrant. Adresář s projektem je uložen v `~/../../var/www/`.

Pro ukončení testovacího prostředí existují dva příkazy:

- Halt
  - ukončí běh testovacího prostředí, ale zachová již nakonfigurovaný obraz v programu VirtualBox. Hlavní výhodou tohoto ukončení je

[1] Při prvním spuštění trvá inicializace déle, protože program stahuje obraz operačního systému Ubuntu 14.04.

rychlejší start při dalším spuštění. Nevýhodou je zachování databáze v době ukončení, což může vést k její neaktuálnosti. Při používání tohoto typu ukončení doporučuji při změnách databáze použít příkaz *Provision*, který databázi aktualizuje.

- Destroy
  - ukončí běh testovacího prostředí a uvolní veškeré místo alokované virtuálním prostředí. Tento typ ukončení zajistí, že při dalším spuštění budeme začínat s čistým obrazem a aktuální databází[2]. Nevýhodou je pomalejší spuštění, jelikož znovu probíhá konfigurace systému.

5.5.4 Migrace databáze

Ke změnám / migracím struktury databáze se v projektu využívá nástroj Phinx. Jedná se o sadu funkcí, které umožňují provádět změny databáze. Veškeré migrace v projektu se ukládají do souboru *migrations*. Migraci vytvoříte zavoláním příkazu „php -f ./vendor/robmorgan/phinx/bin/phinx“ v adresáři s projektem. Příkaz lze volat v příkazovém řádku programu PHPStorm nebo po připojení k testovacímu prostředí pomocí SSH.

Psaní migrací je velice jednoduché. Můžeme používat funkce typu *createTable*, *addColumn* atd. Nebo použít funkci *query()*, do které jako argument vložíme SQL dotaz. Veškeré příkazy pro tvorbu migrací naleznete na `http://docs.phinx.org/en/latest/migrations.html`.

5.5.5 Testování systému

Na testování systému se v projektu využívá unit testů a nástroje phpunit. Tento nástroj je nainstalovaný v testovacím prostředí viz 5.5.3. Testy se spouštějí příkazem „phpunit [adresář s testy nebo vybraný test]“.

V rámci udržení funkčnosti systému je dobré pokrýt testy všechny klíčové funkce. Takto pokrytý systém se snadno upravuje, protože můžeme rychle zjistit, zda naše změny nenarušily již existující funkce.

Všechny existující testy jsou uloženy v projektu v adresáři *tests*. Jak testy vytvářet naleznete na `https://phpunit.de/`.

[2] Za předpokladu používaní programu Phinx ke změnám databáze.

Kapitola 6

Shrnutí systému

Systém se v současné době nachází ve stavu funkčního prototypu. Vzhledem k objektové orientaci by jeho rozšíření nemělo být problémem.

6.1 Budoucí vývoj

Zde jsou uvedeny některé funkcionality, které považuji v rámci budoucího vývoje za prioritní.

- Kontrola dostupnosti položky při vypůjčení.
  - V současné době neexistuje kontrola dostupnosti položky v době vypůjčení, a to jak z hlediska již existujících výpůjček, tak z hlediska otevírací doby lokality, kde si lze položku vypůjčit. Jelikož se jedná o zásadní funkci vzhledem k povaze portálu, považoval bych ji za prioritu při budoucím vývoji.
- Parametry kategorií.
  - Pro usnadnění vyhledávání specifických položek by bylo dobré implementovat používání parametrů jako je například značka, typ nebo cena. Základní filtrování je již připraveno v modelu *Category* ve funkci *getItemsOfCategory*. Avšak přikládání parametrů při vkládání položky v současné době neexistuje.
- Chytré vyhledávání.
- Správa firem.
  - Dokončit správu firem. Umožnit uživatelům přidávat položky ke své firmě, vytvořit stránky firem, kde by se zobrazovaly položky firmou nabízené, a další funkce, které by mohly firmy na takovéto stránce využít.

- Platby na stránce.
  - Zpracovávat platby v rámci portálu.
- Chytré chybové hlášky
  - Při nasazení systému do provozu by bylo dobré vytvořit specifické chybové hlášky, které by pomohly se správou serveru a urychlily hledání problémů.

6.2 Optimalizace / revize

- Kopie údajů při provedení výpůjčky.
  - Osobně bych navrhoval předělat ukládání potřebných údajů při provedení výpůjčky. V současném návrhu se vytváří velké množství kopií tabulek, což je matoucí nejen v databázi, ale také při volaní funkcí na kopírovaní, které probíhá přes velké množství různých tříd.
- Princip položka / instance.
  - Během implementace a hlavně po jejím dokončení se ukázalo, že tento princip přináší pouze problémy a zmatení. Proto bych navrhoval předělat systém tak, aby pracoval pouze s jednou entitou. Současný princip byl zaveden pro ulehčení vkládání více podobných položek, čehož lze docílit úpravou formuláře pro vkládání tak, že bude možné zadané údaje zkopírovat a vytvořit totožnou položku, u které se poté změní pouze identifikátor (SPZ, barva, rok výroby atd.).

Dále je samozřejmě možné optimalizovat systém ať už úpravou kódu, nebo databázového modelu. Tyto změny bych však považoval za druhořadé ve srovnání s těmi výše uvedenými.

Závěr

Ve své práci jsem vytvořil dokumentaci backendu projektu „Webový objednávkový systém na pronájem zařízení/věcí“. Dokumentace obsahuje diagramy:

- případy užití,
- doménový model,
- diagram tříd,
- diagram aktivit,
- databázový model,
- model nasazení.

Provedl jsem refaktor kódu, který se skládal z odstranění duplicitního kódu, nepoužívaných tříd, funkcí, a upravil jsem některé části kódu tak, aby byl kód konzistentní v rámci celého projektu.

Dále jsem vytvořil API pro připojení externích aplikací, které umožňuje připojení mobilní aplikace, vytvořené Martinem Židem, k našemu portálu.

V poslední řadě jsem připravil vývojářské prostředí, které lze využít při budoucí práci na projektu. Toto prostředí obsahuje:

- verzovací nástroj GIT,
- ticketovací nástroj Redmine,
- autodeploy,
- testovací prostředí Vagrant.

Literatura

[1] What is PHP? php [online]. *The PHP Group*, [cit. 2016-04-21]. Dostupné z: http://php.net/manual/en/intro-whatis.php

[2] Databáze MySQL. Artic Studio [online]. *ARTIC STUDIO*, [cit. 2016-04-21]. Dostupné z: https://www.artic-studio.net/slovnicek-pojmu/databaze-mysql/

[3] Úvod do Nette frameworku. Itnetwork [online]. *David Čápka*, 2014, [cit. 2016-04-22]. Dostupné z: http://www.itnetwork.cz/php/nette/zaklady/uvod-do-php-frameworku-nette/

[4] API. Wikipedia: the free encyclopedia [online]. *Wikimedia Foundation*, 2001-2016, [cit. 2016-04-29]. Dostupné z: https://cs.wikipedia.org/wiki/API

[5] Secure Your REST API... The Right Way. Stormpath [online]. *Stormpath*, 2013, [cit. 2016-05-03]. Dostupné z: https://stormpath.com/blog/secure-your-rest-api-right-way/

[6] GIT Hooks. GIT Hooks [online]. *Matthew Hudson*, [cit. 2016-05-07]. Dostupné z: http://githooks.com/

Příloha **A**

Seznam použitých zkratek

IČ Identifikační číslo osoby

API Application Programming Interface

HTTP The Hypertext Transfer Protocol

JSON JavaScript Object Notation

HTTPS Hyper Text Transfer Protocol Secure

TLS Transport Layer Security

SSL Secure Sockets Layer

WOS „Webový objednávkový systém na pronájem zařízení/věcí“

SSH Secure Shell

SPZ Státní poznávací značka

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `wosnpz2015` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `documentation` ..... zdrojová forma dokumentace
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS
- `documentation` ..... dokumentace projektu
  - `code` ..... dokumentace kódu
  - `analytic` ..... analytická dokumentace
  - `api.pdf` ..... podrobný popis API
- `images` ..... UML diagramy ve formátu PNG