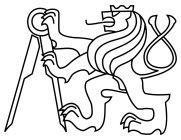
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Portál ITStudentHelp
Student:	Jan Kabela
Vedoucí:	doc. RNDr. Josef Kolář, CSc.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

Tématem práce je analýza, návrh a vytvoření back-end části webové aplikace v rámci širšího zadání, jehož úkolem je vytvořit portál, který bude sloužit jako místo, na které se budou obracet žadatelé o drobné služby související především s osobním využíváním výpočetní techniky, tabletů, mobilních telefonů apod. Vzorové řešení takového portálu představuje https://www.studentaanhuis.nl/. Back-end bude pokrývat všechny základní funkce portálu, tj. registraci klientů a jejich žádosti o službu, registraci studentů pro poskytování služeb, výběr studenta, evidenci a hodnocení poskytnuté služby, tvorba podkladů pro účtování služeb a výplatu studentů, apod. Ve spolupráci s tvůrcem front-endové části též proveďte výběr vhodných technologií a navrhněte vhodnou formu vzájemné komunikace mezi back- a front-end částmi portálu. Vytvořené řešení řádně otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 1. března 2016

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 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Kabela, Jan. *ITStudentsHelp Portál*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato bakalářská práce je součástí projektu zabývajícího se realizací webového portálu ITStudentsHelp. Tento portál by měl zprostředkovávat pomoc zákazníkům s problémy v oblasti IT. Tato pomoc bude realizována z větší části studenty univerzity ČVUT Fakulty informačních technologií a dále pak i učiteli této fakulty (případně jiných škol). Cílem této bakalářské práce je vytvořit pro daný webový portál backend, skládající se z databáze a funkcionalit backendu. Funkcionalitami se rozumí vytváření uživatelských účtů, vytváření zakázek, nakonec pak funkcionality pro posílání dat na frontend. Pro řešení databáze jsem zvolil PostgreSQL, pro backend server Node.js. Pro komunikaci mezi frontendem a backendem je zvolen protokol HTTPS a pro přesnost dat se používá struktura JSON. Výsledkem této práce je funkční a otestovaný backend server s databází, připravený na propojení s ostatními částmi webového portálu, vyvíjenými v projektu ITStudentsHelp portál.

Klíčová slova PostgreSQL, Node.JS, backend, databáze, moduly

Abstract

This bachelor thesis is part of project called ITStudentsHelp portal. This portal should arrange help for customers with IT problems. This help will be done mostly by students of Czech technical University in Prague, Faculty of Information Technology rarely then by teachers of this faculty (eventually other universities). The goal of this bachelor thesis is to develop a database and backend server for this web portal. Backend server means for example creating user accounts, creating tasks and sending data to frontend. For creating database I chose PostgreSQL, for backend was chosen Node.js. For communication between frontend and backend was chosen HTTPS and for transferring data between frontend and backend was chosen JSON structure. Result of this bachelor thesis is functional and tested backend server with database. This part of the project ITStudentsHelp portal is ready to be connected with the rest of ITStudentsHelp portal parts.

Keywords PostgreSQL, Node.JS, backend, database, modules

Obsah

Seznam obrázků

Úvod

V současné době se v České republice nevyskytuje mnoho portálů zaměřených na pomoc zákazníkům v oblasti IT. Pokud se zde nějaké portály vyskytují, neposkytují komplexní pomoc. Jsou to například autorizované servisy, které sice nabízejí pomoc, ale pouze s výrobky od některých výrobců.

V případě webového portálu ITStudentsHelp, vyvíjeného v rámci této bakalářské práce, by měla tato mezera být zaplněna. Cílem tohoto portálu je nabídnout zákazníkům co nejrozsáhlejší pomoc s problémy z oblasti informačních technologií. Tohoto cíle by se mělo dosáhnout tím, že jako zaměstnanci služby v rámci portálu ITStudentsHelp budou vystupovat především studenti Fakulty informačních technologií ČVUT (případně jiných fakult či škol) a v omezené míře i vyučující. Jelikož se na fakultě můžeme setkat s velkým rozsahem znalostí v rámci IT, měla by tedy být možnost pomoci zákazníkům s většinou problémů, s kterými se mohou setkat.

Zároveň toto řešení přináší problém. Zaměstnanci nebudou pouze pracovat jako pracovníci pro portál ITStudentsHelp, ale zároveň jsou vyučujícími nebo studenty na fakultě, což má za následek, že nebude možné zákazníkům pomoci okamžitě po nastání problému, ale například v rámci následujících dvou dnů. Toto přináší nevýhodu oproti webovým portálům, které nabízejí okamžitou pomoc. Takováto pomoct je ovšem kvůli okamžitému řešení dražší z důvodu, že pracovníci musejí být neustále k dispozici. Webový portál ITStudentsHelp budou tedy využívat zákazníci, kteří zvládnou s daným problémem fungovat za vidiny nižších poplatků.

Vytvoření webového portálu ITStudentsHelp je projekt, který je v rámci realizace rozdělen do tří částí. První částí je frontend, tuto část vytváří Igor Kulka, a další částí je backend. Backend je rozdělený do dvou částí. Částí backendu je vytvoření administrátorského zázemí. Tuto část realizuje Štefán

Töltési. Poslední složkou projektu a druhou částí backendu je vytvoření databáze a funkcionalit pro získávání dat z databáze, vytváření uživatelských účtů a zakázek. Tomuto úkolu jsem se věnoval já a jeho řešení je popsáno v této bakalářské práci.

Kapitola 1

Cíl práce

Cílem celého projektu ITStudentsHelp je vytvořit funkční portál pro poskytování pomoci v oblasti IT takovým zákazníkům, kteří patří spíše mezi začátečníky nebo méně zkušené uživatele. Cílem této bakalářské práce je vytvořit část backendu skládající se z následujících částí.

1. Vytvoření databáze
2. Vytvoření následujících funkcionalit
   a) Vytváření uživatelských účtů a to pro zákazníky i zaměstnance
   b) Editace uživatelských účtů
   c) Vytváření jednotlivých zakázek od zákazníků
   d) Editace zakázek od zákazníků
   e) Poskytování zpětné vazby od zákazníků
   f) Přidělování jednotlivých zakázek zaměstnancům
   g) Tvorba podkladů pro výplaty zaměstnanců
   h) Selekce dat z databáze pro zobrazení na frontendu
   i) Změna a obnovení hesla
      i. Selekce detailních informací o zaměstnanci
      ii. Selekce seznamu všech zaměstnanců
      iii. Selekce informací o jednotlivých kategoriích
3. Všechny tyto části řádně otestovat

Kapitola 2

Analýza a návrh

2.1 Současná řešení

V současné době existují webové portály, které poskytují zákazníkům pomoc v rámci IT, ale povětšinou nejsou komplexní a na jejich službách se nepodílejí v rozhodující míře studenti.

Portály jako bude ITStudentsHelp existují pouze v zahraničí. Jeden takovýto webový portál je v Nizozemsku [1]. Druhý takovýto webový portál existuje ve Velké Británii [2]. Oba tyto webové portály poskytují zákazníkům komplexní pomoc v oboru IT. Tato pomoc je realizovaná studenty, kteří za zákazníky dojíždějí a s jejich problémy jim pomáhají. Tyto dva portály jsou úspěšné, proto je výhodné je využít jako vzor pro realizaci webového portálu ITStudentsHelp.

V této bakalářské práci se ze zmiňovaných portálů můžu inspirovat ve dvou oblastech. První oblastí je vymezení rozsahu dat, která bude nutné ukládat do databáze, a druhá oblast se týká hlavně toho, jaké jsou a jak fungují jednotlivé funkcionality.

2.2 Možnosti řešení

Pro realizaci backendu je třeba se nejprve věnovat dvěma základním otázkám, a to výběru databáze a způsobu implementace funkcionalit backendu.

2.2.1 Databáze

Pro databázi je zde možnost volby databázi objektově orientované nebo relační. Oba typy se v dnešní době využívají.

Relační databáze je postavená na jednotlivých tabulkách, obsahujících sloupce a řádky, každý sloupec má své jméno a datový typ a multidimenzionální objekty jsou reprezentovány spojenými tabulkami.[3]

Literatura [4] uvádí následující charakteristiku:

"Objektové databáze kombinují prvky objektově orientovaného programování s databázovými schopnostmi. Rozšiřují funkčnost objektových programovacích jazyků (C++, Smalltalk, Java) a poskytují plnou schopnost programování databáze. Datový model aplikace a datový model databáze se ve výsledku hodně shodují a výsledný kód se dá mnohem efektivněji udržovat."

Další autoři uvádějí (viz např. [5]), že relační databáze s velkým objemem uložených dat (např. 10^8 záznamů) může být výrazně pomalejší než objektově orientovaná databáze.

2.2.2 Tvorba backendu

U tvorby backendu pro webový portál je mnoho možností. Základem je určit si požadavky pro backend, například[6]:

1. *Kolik uživatelů bude celkově webový portál resp. backend využívat*
2. *Kolik uživatelů bude přistupovat za sekundu*
3. *Je potřeba aby backend komunikoval se serverem v reálném čase*

Podle těchto požadavků si pak můžeme vybrat například mezi: Java, Python, Ruby, PHP nebo Node.js. Všechny tyto možnosti by posloužily k tvorbě backendu k webovému portálu. Je tedy na preferencích daného člověka, co si z daného seznamu vybere.

2.3 Analýza požadavků

Webový portál budou využívat dva typy uživatelů. Prvním typem budou zákazníci, kteří budou zadávat zakázky. Zakázkou je myšlen problém, který u zákazníka nastal a je potřeba vyřešit.

Druhým typem uživatelů jsou zaměstnanci. Zaměstnanci se dále budou dělit na specialisty, jejichž úkolem bude vyřizování jednotlivých zakázek, tedy navštíví zákazníka a pomohou mu vyřešit problém, který se vyskytl. Jiní zaměstnanci budou pracovat na telefonu a budou se snažit se zákazníkem specifikovat co nejpřesněji jeho problém, a to z důvodu přidělení správného specialisty. Posledním typem zaměstnanců budou administrátoři, jejichž náplní práce bude péče o správné fungování webového portálu.

Pro využívání webového portálu bude pro každého uživatele nutností se zaregistrovat. Registrace bude obnášet vyplnění osobních údajů a volbu hesla pro přihlášení. Poté co si uživatel vytvoří uživatelský účet, bude moci po přihlášení využívat funkcí webového portálu.

Zákazník bude mít možnost si zvolit, zda bude platit roční členské poplatky, v tom případě bude sazba za jednotlivé zakázky snížena. Tuto možnost si vybere zákazník, jenž bude tyto služby využívat často. V případě že zákazník využije službu nepravidelně jen několikrát za rok, nebude muset platit členské poplatky a každá zakázka mu bude účtována plnou sazbou. Bude tedy nutné u zákazníků uvádět, zda mají uhrazený členský poplatek, případně kdy jim členství vyprší.

Pro zjednodušení přiřazování zaměstnanců k jednotlivým zakázkám bude u každého zaměstnance uvedená specializace popř. více specializací. Každá specializace bude ještě mít podspecializace. U zaměstnance bude potřebné pouze ukládat, jaké podspecializace ovládá, jelikož u každé podspecializace bude jasně dáno, do které specializace patří.

Jelikož je pro využívání webového portálu ITStudentsHelp nutné přihlášení, je důležité myslet na to, že uživatel si může přát změnit heslo. Je tedy nutné mu tuto možnost poskytnout. Z důvodu že dnes uživatelé mají své účty na mnoha webových portálech, je možné, že uživatel své heslo zapomene. V tomto případě je zapotřebí umožnit uživateli vygenerovat nové dočasné heslo.

Může se stát, že uživatelé, zejména zákazníci, si nebudou s něčím vědět rady. Toto povede k situaci, že zákazníci budou pokládat otázky pomocí e-mailu. Vyřešení takovýchto e-mailů nějakou dobu trvá. Bude tedy praktické tyto otázky zaznamenávat a následně, pokud se některé otázky budou opakovat častěji, zobrazovat je jako FAQ. FAQ urychlí zodpovězení otázek zákazníků.

Důležité je také zamyslet se nad bezpečností. Bezpečnost je důležitá, jelikož se v rámci komunikace frontendu a backendu budou přenášet osobní data. Je tedy potřeba, aby nebylo možné tyto informace nějak odposlouchávat nebo je jinak zneužívat. Jelikož se uživatelé budou přihlašovat, bude jasné, kdo jednotlivé akce na backendu provádí. Pokud by komunikace nebyla zabezpečena, mohl by pak kdokoli získat přihlašovací údaje a vystupovat za jiného uživatele. Odposlech přihlašovacích údajů by mohl způsobit největší, škody pokud by se jednalo o údaje administrátora.

Nakonec je nutné ošetřit data, jež se budou vkládat do databáze, a to z důvodu sqlinjection (zmíněno v [7])

"Přinejlepším se útočník dostane tam, kam nemá, ale neprovede nic s vaší aplikací, v horším případě se dostane třeba k uživatelským heslům (spíše hashům hesel) a v nejhorším případě vám smaže tabulky nebo upraví jejich obsah."

2.4 Zvolená řešení

Po zvážení jednotlivých možností jsem si zvolil pro implementaci databáze relační databázi. Tuto volbu jsem učinil zejména z důvodu mé znalosti relačních databází, a to konkrétně předmětů zaměřených na databáze. Při volbě systému, ve kterém databázi vytvořit, jsem vzal v potaz, že již ovládám MySQL a ORACLE databáze. Jelikož bych si rád vyzkoušel i nějaký nový systém, jež je podobný zmiňovaným, padla volba na PostgreSQL.

Z mnoha zdrojů vyplývá, že vývojáři stále nejsou zajedno, když padne otázka výběru jedné konkrétní technologie pro tvorbu relační databáze. Např. v [8] se autor zmiňuje o výhodách a nevýhodách PostgreSQL. "Z hlediska programátora je výborná možnost přístupu z mnoha jazyků C/C++ (knihovny libpq a libpq++), scriptovacích jazyků Perl, Python, ale i třeba javy (jdbc) a PHP. Já sám jsem použil 'pouze' poslední tři přístupy. Vím, že teď asi vyvolám nevoli a možná i divoký flame, ale PostgreSQL se mi jeví jako mnohem lepší databázové řešení, než MySQL, protože nabízí spoustu vymožeností, které MySQL nezvládá. Naproti tomu MySQL je nepoměrně rychlejší a na menší webové aplikace více než dostatečná. Jako vždy je to především o aplikaci."Spíše se ale na internetu setkáme s články, které jsou jednoznačně proti PostgreSQL nebo naopak pro PostgreSQL. Abych si vytvořil svůj vlastní názor na PostgreSQL, zvolil jsem tuto technologii pro svou práci.

Při volbě technologií pro tvorbu backendu je nutné myslet na to, že pro databázi byl zvolen PostgreSQL. Je tedy zapotřebí zvolit pro backend technologii, která PostgreSQL podporuje. Z předešlého odstavce vyplývá, že PostgreSQL je podporová všemi technologiemi zmiňovanými v možnostech řešení. Pro tvorbu backendu jsem zvolil Node.js. Jak zde [9] zmiňuje autor:

V porovnání Node.js s tradičními webovými technologiemi, kde každý požadavek vytvoří nové vlákno, což následně vede k velkému využití RAM, Node.js pracuje na jednom vlákně pomocí neblokujících I/O volání. Toto umožňuje uskutečňovat desítky tisíc paralelních připojení.

Node.js navíc obsahuje mnoho modulů, které se dají díky NPM nástroji jednoduše instalovat z online repozitářů. Tyto moduly pak zjednoduší práci při vytváření backendu.[9]

Co se týče bezpečnosti komunikace mezi backendem a frontendem, postačí zde volba https na místo http. HTTPS je Hypertext transfer protokol secure, znamená to tedy, že data se neposílají jako plain text, ale jsou zašifrované pomocí dvou klíčů. Jedním klíčem je veřejný klíč serveru (ten šifruje) a druhým klíčem je soukromý klíč serveru (ten dešifruje). Pro testování backendu postačí využít selfsigned certifikáty.

Ohledně sqlinjection se počítá s tím, že frontend bude zasílat na server pouze validní data. Ovšem mohlo by se stát, že se někdo pokusí zaslat data na server bez využití frontendu, se záměrem smazání databáze nebo výběru osobních dat z databáze. V tomto případě je nutné u funkcionalit, které budou komunikovat s databází na základě přijatých parametrů, tyto parametry validovat. Validace proběhne tak, že se v řetězcích odstraní speciální znaky a u číselných hodnot se zkontroluje, zda se jedná opravdu o číslo.

Z mého osobního hlediska přínos bakalářské práce nebude jen vytvoření backend serveru a databáze pro webový portál ITStudentsHelp, ale také doplnění PostgreSQL a Node.js k mým znalostem.

Kapitola 3

Realizace

3.1 Databáze

3.1.1 Uživatelé

Z analýzy vyplývá, že je nutné ukládat informace o uživatelích tvořených zaměstnanci a zákazníky. Pro zákazníky i zaměstnance budou některé údaje stejné. U každého zaměstnance i zákazníka bude uvedeno:

- Křestní jméno (Firstname)
- Příjmení (Surname)
- Adresa bydliště
  - Ulice (Street)
  - Číslo popisné (HouseNumber)
  - Město (City)
  - PSČ (Zip)
- Telefonní číslo (Telephone)
- E-mail (E-mail)
- Přihlašovací heslo (Password)
- Fotografie (Photo)
- Token (Token)

Bude-li se jednat o zaměstnance, připojí se k údajům ještě následující informace:

- Specializace (Specialization)
- Podspecializace (SubSpecialization)
- Role (Role)
- Univerzita (School)
- Fakulta (Faculty)
- Obor (Department)
- Stupeň (Level)
- Ročník (Grade)
- Datum narození (DateOfBirth)
- Hodnocení (Rating)
- Odpracováno tento měsíc (WorkedThisMonth)

V případě zákazníka bude k údajům přidáno ještě členství a to:

- Člen od (PrepaidFrom)
- Člen do (PrepaidTill)

3.1.2 Uživatel

Jelikož jsou některé údaje u zaměstnanců a zákazníků stejné, bude praktické je ukládat do stejné tabulky nazvané uživatel (User). V této tabulce budou uloženy údaje o uživateli, které jsou pro každého uživatele jedinečné. Takovéto údaje jsou: křestní jméno, příjmení, telefonní číslo, e-mailová adresa, přihlašovací heslo a fotografie. Fotografie je nepovinná část. Pokud uživatel nenahraje fotografii, bude se u jeho profilu zobrazovat defaultní avatar.

3.1.3 Adresa

Adresa nemusí být pro každého uživatele unikátní, a to z důvodu např. jedná-li se o bytový či panelový dům. Může se tedy stát, že více uživatelů bude bydlet na téže adrese. Z tohoto důvodu je nutné vytvořit samostatnou tabulku adresa (Address), obsahující adresu bydliště, tedy ulici, číslo popisné, město, PSČ. Tato tabulka bude napojena z výše zmiňovaných důvodů k tabulce uživatel vazbou M:N.

3.1.4 Zaměstnanec

Pro zaměstnance je potřebné vytvořit novou tabulku, neboť se pro něj zaznamenává více informací než pro zákazníka. Jedná se o následující údaje: specializace, podspecializace, role, univerzita, fakulta, obor, ročník a datum narození. Tabulka zaměstnanec (Employee) bude napojena na tabulku uživatel (User) vazbou 1:1.

Pro zaznamenávání specializace resp. podspecializace je nutné vzít v potaz, že více zaměstnanců může ovládat stejné specializace resp. podspecializace. Stejně jako u adres je nutné separovat specializace a podspecializace do samostatné tabulky. Nově vzniklou tabulku spojíme opět s tabulkou zaměstnanec ve vztahu M:N.

Při pohledu na tabulku specializace a podspecializace je zjevné, že tyto dvě množiny není možné udržovat v jedné tabulce. A to z důvodu, že některé specializace mohou mít více podspecializací, je tedy věcnější vytvořit dvě separátní tabulky, a to specializace (Specialization) a podspecializace (Subspecialization). Tabulka podspecializace (Subspecialization) bude napojena na tabulku uživatel se vztahem M:N. Tabulka specializace (Specialization) bude spojena s tabulkou podspecializace (Subspecialization) N:1.

Specializace (Specialization) i podspecializace (Subspecialization) budou obsahovat:

- Název (Title)
- Typ (Type)

Bylo by možné uvádět pouze název, jelikož typ bude pouze zkrácený název. Zavedení typu ovšem přinese rychlejší porovnávání v databázi. Zároveň zavedení typu způsobí, že v rámci přijímání dat z frontendu bude stačit přijímat pouze typ, který bude kratší než název.

Dále se u zaměstnanců zaznamenává informace o roli popř. rolích, které zastávají. Jak bylo dříve zmíněno, počítá se se třemi rolemi a to specialista, telefonista a admin. Opět je zde možné nebo spíše jisté, že více zaměstnanců bude zastávat stejnou roli popř. role. Z tohoto důvodu je jako dříve nutné vytvořit novou entitu role (Role), jenž bude na tabulku zaměstnanec (Employee) napojena vztahem N:M. U tabulky role (Role) stačí, aby obsahovala pouze název role.

Zaměstnanec bude dostávat průběžně od zákazníků hodnocení jeho práce. Je tedy potřeba toto hodnocení zaznamenávat. Jelikož zaměstnanec průběžně nastřádá mnoho hodnocení a každé u něj musí být uvedeno, je pro uložení hodnocení praktické zvolit novou tabulku hodnocení (Rating), obsahující:

- Hodnocení (Rating)
- Komentář (Comment)

Hodnocení bude v rozsahu 1 až 5 hvězdiček (* až *****), 5 nejlepší, 1 nejhorší. Komentář bude sloužit pro upřesnění daného hodnocení. Tato tabulka bude napojena na tabulku zaměstnanec N:1.

Bude praktické u jednotlivých zaměstnanců mít uvedeno jejich průměrné hodnocení. Pro průměrné hodnocení bude vytvořena nová tabulka průměrné hodnocení (AverageRating) obsahující:

- Průměrné hodnocení (Average)
- Počet hodnocení (NumberOfRat)

Pro počítání průměrného hodnocení je zapotřebí vytvořit trigger, který vždy při přidání hodnocení zákazníkovi přepočítá jeho průměrné hodnocení a aktualizuje ho. Pro tento účel je v této tabulce kromě průměrného hodnocení také uveden počet hodnocení. Průměrné hodnocení by se dalo počítat z jednotlivých tabulek vždy, když je potřeba, ale bylo by to časově náročné. Tabulka průměrné hodnocení bude napojena na tabulku zákazník vazbou 1:1.

3.1.5 Zákazník

Pro zákazníka je nutné, stejně jako pro zaměstnance, vytvořit samostatnou tabulku a to ze stejných důvodů. Tabulka zákazník bude propojena s tabulkou uživatel vazbou 1:1.

Pro zákazníka je nutné ukládat od kdy do kdy je členem. Ovšem je také důležité zaznamenávat, od kdy do kdy byl členem. Toto vede na nutnost vytvoření nové tabulky bývalá členství (PreviousMembership). V této tabulce budou uloženy informace o bývalých členstvích (PreviousMembership) a bude napojena na tabulku zákazník (Customer) vztahem N:1.

Zákazník musí být svázán se všemi hodnoceními, která udělil, aby bylo možné zpětně pro hodnocení nalézt autora. Jelikož už byla dříve vytvořena tabulka hodnocení, stačí pouze propojit hodnocení (Rating) se zákazníkem (Customer) vazbou 1:N.

3.1.6 Zakázka

Kromě jednotlivých uživatelů bude nutné ukládat zakázky (Task), které zákazníci zadali do systému. U každé zakázky bude uvedeno:

- Jméno (Name)
- Popis (Description)
- Adresa plnění zakázky
  - Ulice (Street)
  - Číslo popisné (HouseNumber)
  - Město (City)
  - PSČ (ZIP)
- Specializace (Specialization)
- Podspecializace (SubSpecialization)
- Hodnocení od zákazníka (Rating)
- Přiřazeno zaměstnancem (Customer)
- Vytvořeno zákazníkem (Employee)
- Doba plnění (Duration)
- Vytvořeno (Added)
- Přiřazeno zaměstnanci (Assigned)
- Dokončeno (Done)

Jméno zakázky by mělo v krátkosti vystihovat předmět zakázky např. instalace MS Office na operační systém Windows. Popis zakázky bude obsahovat podrobný popis problému, který u zákazníka nastal.

Adresa u zakázek bude uvedena z důvodu, že adresa plnění nemusí vždy být shodná s adresou bydliště zákazníka např. bude-li zákazníkem živnostník, kterému se vyskytl problém v kanceláři mimo domov. Jelikož už dříve byla vytvořena tabulka adresa pro zaznamenávání bydliště uživatelů, stačí stejně jako u uživatelů propojit tabulku zakázka (Task) s tabulkou adresa (Address) N:1.

Každá zakázka bude obsahovat specializaci a podspecializaci problému, který je v jejím rámci potřeba vyřešit. Specializace a podspecializace bude uvedena zejména z důvodu přiřazení kvalifikovaného zaměstnance. Jak už bylo zmíněno u zaměstnance, každá podspecializace je svázána se specializací a stačí tedy uvádět pouze podspecializaci. Toho se docílí přidání vazby mezi zakázkou a podspecializací N:1.

Pro každou zakázku je důležité vědět, jaké hodnocení získala resp. jak dobře ji zaměstnanec vypracoval. To zaručí provázání tabulky zakázka s tabulkou hodnocení (Rating) vazbou N:1.

Ke každé zakázce bude přiřazen zaměstnanec specialista, který ovládá specializaci, jež je u zakázky uvedena. Ten pomůže zákazníkovi s vyřešením problému. Toto bude realizováno spojením zakázky (Task) a zaměstnance (Employee) vazbou N:1.

Aby bylo jasné, kdo zakázku vytvořil popř. zjistil kontaktní informace na zákazníka, je zapotřebí uvést jméno i zákazníka, pro kterého se bude objednávka plnit. Z tohoto důvodu se tabulka Zakázka (Task) propojí s tabulkou Zákazník (Customer) vazbou N:1.

3.1.7 Otázky

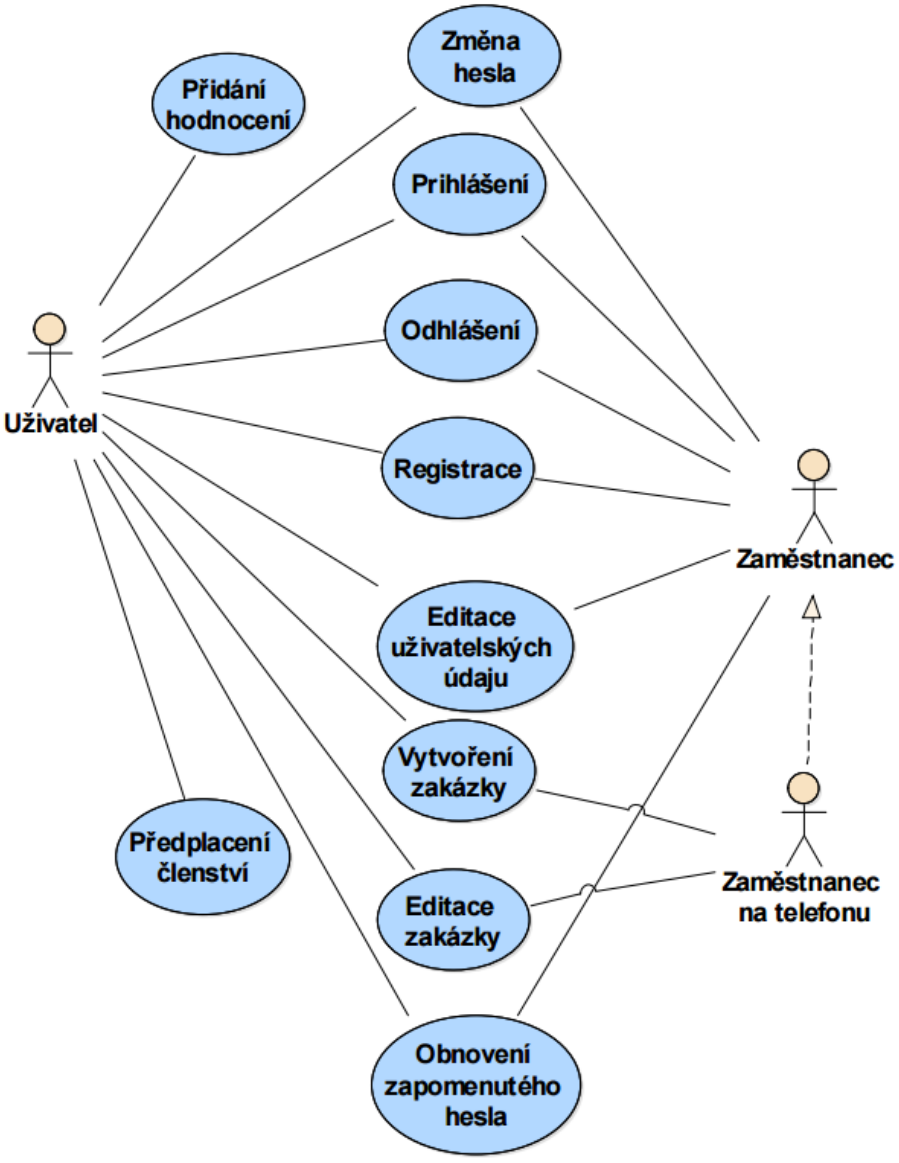
Jak už bylo zmíněno v analýze, je potřeba ukládat otázky, které budou zákazníci pokládat v průběhu fungování webového portálu. Takovéto otázky se budou zaznamenávat do tabulky Otázky (Questions), která bude obsahovat následující údaje:

- Otázka (Question)
- Řešení (Solution)

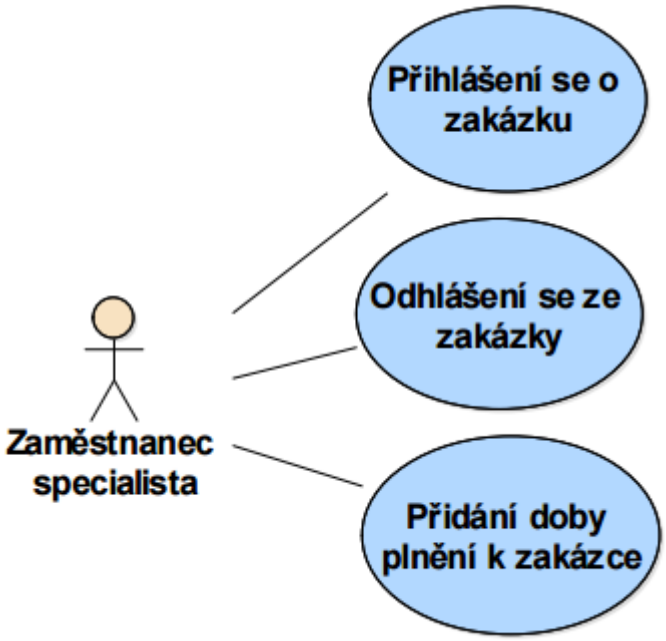
Z takto uložených otázek budou pak moci zaměstnanci zvolit otázky, které jsou pokládány často, a vložit je do tabulky FAQ, která bude napojena na tabulku Otázky (Questions) vazbou N:1. Z této tabulky pak bude čerpat frontend pro zobrazování FAQ. Vkládání do tabulky FAQ z tabulky Otázky (Questions) bude muset být prováděno manuálně, a to z důvodu že jedna otázka může být formulována mnoha způsoby. To má za následek, že je nemožné toto vkládání provádět automaticky.

3.2 Funkcionality

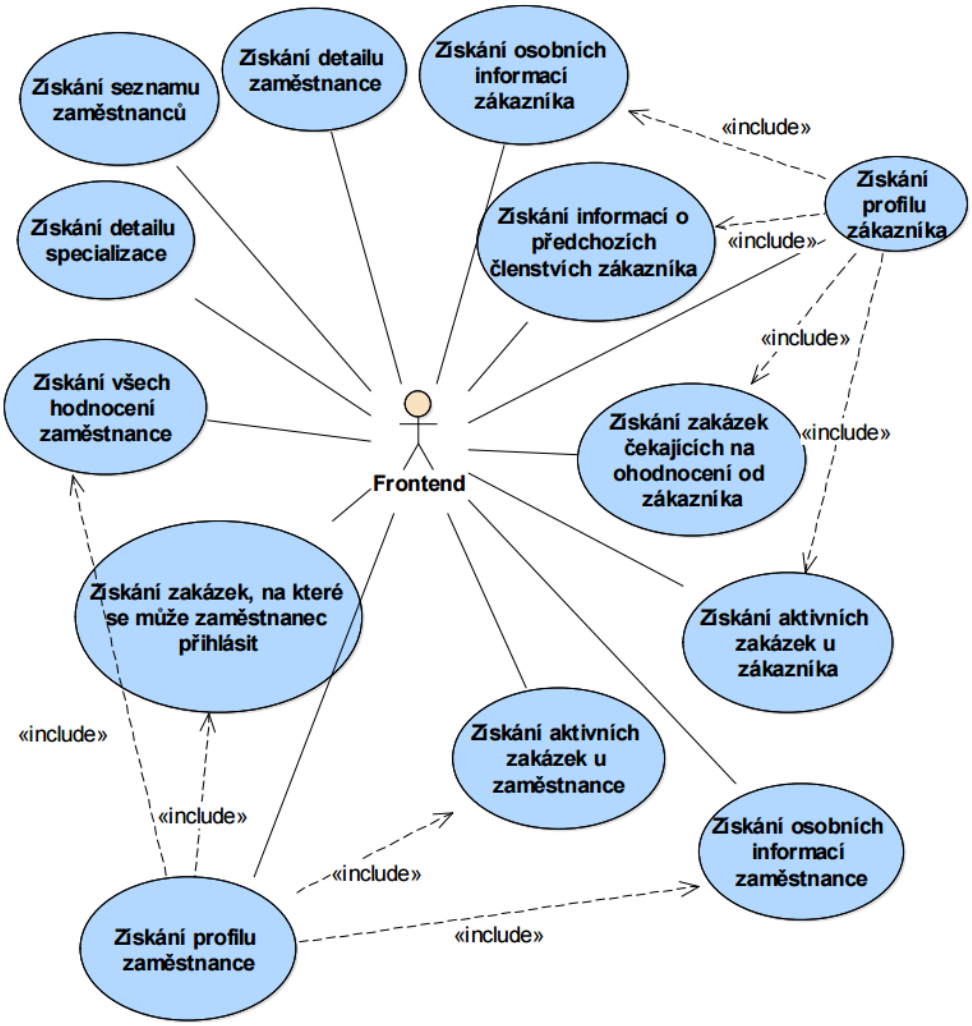
Funkcionality se dělí na tři části. První částí jsou funkcionality pro uživatele. Druhou částí jsou funkcionality pro zaměstnance specialistu a poslední částí jsou funkcionality pro posílání dat na frontend.



Obrázek 3.1: UseCase uživatel

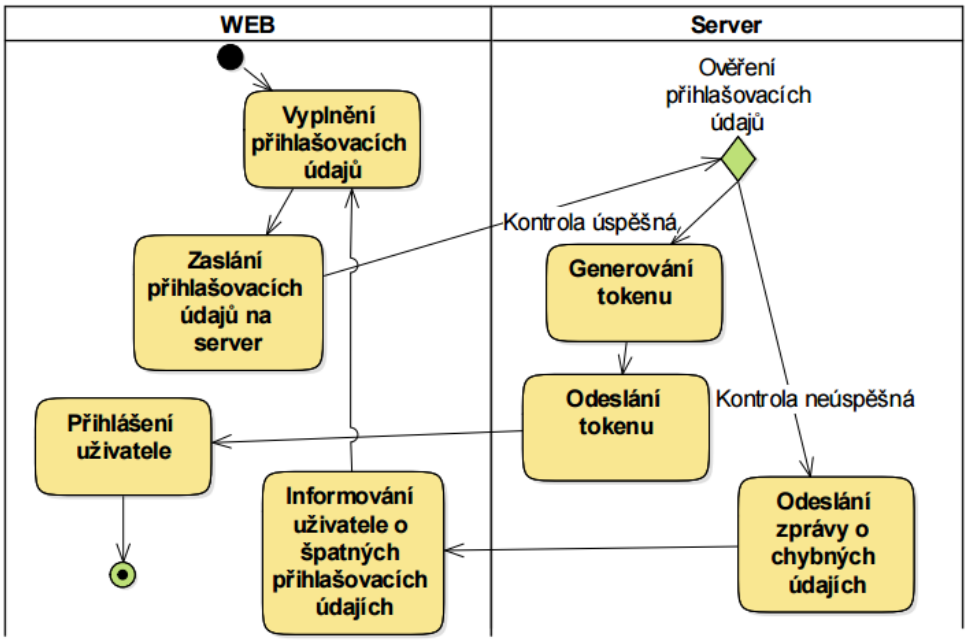


Obrázek 3.2: UseCase zaměstnanec specialista



Obrázek 3.3: UseCase frontend

3.2.1 Přihlášení



Obrázek 3.4: Activity diagram Přihlášení

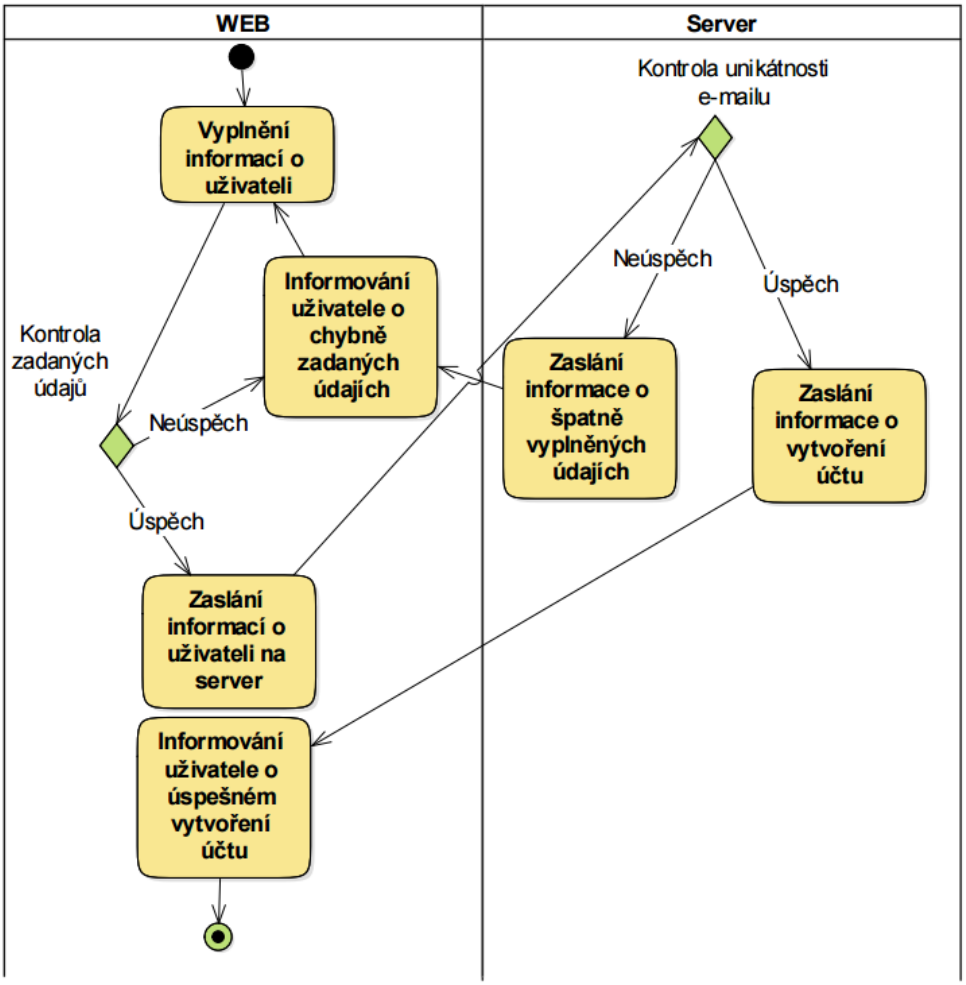
Jak už bylo zmiňováno dříve, uživatelé se pro práci s webovým portálem budou muset přihlásit. Pro přihlášení uživatelé vyplní na webovém portálu svoji e-mailovou adresu a heslo. Tato data se zašlou na server. Na serveru se ověří, že existuje uživatel s danou e-mailovou adresu a heslem. Pokud takový uživatel neexistuje, pošle se upozornění o špatně zadaném e-mailu nebo heslu.

Pokud uživatel existuje, vygeneruje se pro něj token. Ten poslouží k ověření, zda je uživatel přihlášen vždy, když bude provádět akci, která bude mít za následek změnu dat v databázi např. vytvoření zakázky. Token se bude generovat pomocí hashe a to md5. Pro vstup do md5 se použije uživatelův e-mail a heslo. Tento token se následně uloží do databáze k uživateli do položky token a zašle se na frontend zároveň se zprávou o úspěšném přihlášení.

3.2.2 Odhlášení

Pokud uživatel dokončí práci na webovém portálu, provede odhlášení. Odhlášení bude spočívat v tom, že nejdříve se ověří, zda má validní token. Pokud disponuje tokenem, vymaže se tento token ze záznamů o uživateli. To způsobí, že uživatel nebude moci nadále provádět úkony na webovém portálu, které by měnily data v databázi.

3.2.3 Vytvoření uživatelského účtu



Obrázek 3.5: Activity diagram Vytvoření uživatele

Zákazník i zaměstnanec si budou muset před využíváním webového portálu vytvořit uživatelský účet. Pro vytvoření účtu bude na webovém portálu připraven formulář, kde vyplní uživatelské údaje, jež jsou zmíněny v tabulce uživatel. V případě, že se bude jednat o zaměstnance, přibydou ještě informace obsažené v tabulce zaměstnanec.

Takto vyplněné údaje se z webového portálu zašlou na server. Na serveru bude nutné zkontrolovat, zda e-mailová adresa není již využívána jiným uživatelem, z důvodu využívaní e-mailové adresy jako přihlašovacího jména. Pokud kontrola proběhne v pořádku, uživatel bude uložen do databáze a na webový portál se zašle informace o úspěšném vytvoření uživatele. V případě duplicitní

e-mailové adresy bude poslána na webový portál informace o neúspěšném vytvoření účtu.

3.2.4 Úprava údajů uživatele

Může nastat situace, kdy uživatel, ať už zákazník nebo zaměstnanec, změní své bydliště. Popřípadě se může změnit kontakt na uživatele, a to telefonní číslo. V některých případech se může u uživatele změnit i jeho jméno. E-mailová adresa bude neměnná. Možnost změny těchto údajů bude uživateli poskytnuta na webovém serveru v jeho profilu, kde bude moci změnit veškeré údaje mimo zmíněné e-mailové adresy. Změněné údaje se pak zašlou na server a to tak, že nezměněné údaje budou označeny značkou false, ostatní údaje se zašlou podle vyplnění uživatele.

Pokud se bude jednat o zaměstnance, může nastat možnost, že si bude chtít změnit své specializace. V tom případě budou rozděleny na dvě části. V první části budou uvedeny specializace, které se mají zaměstnanci přidat. V druhé části budou uvedeny specializace, které si zaměstnanec přeje odebrat. Jednotlivé údaje se pak změní v databázi.

3.2.5 Prodloužení členství

Pokud zákazník uzná za vhodné, že by se mu vyplatilo být členem, aby platil za zakázky sníženou sazbu, nebo zákazník již členem je a chtěl by si členství prodloužit, bude mít dvě možnosti. První možností je předplatit si členství na půl roku. Druhou možností je předplacení si členství na celý rok. Pokud tak zákazník udělá a zaplatí, bude mu buď prodlouženo jeho stávající členství o dobu, kterou si zvolil, nebo se u něj zavede členství nové. Zavedení nového členství obnáší vyplnění zákazníkovy kolonky "členem od", to bude den, kdy zákazník za členství zaplatil, a kolonky "členem do", která závisí na době, na kterou si zákazník členství předplatil.

3.2.6 Získání podkladů pro peněžní ohodnocení

Pro funkčnost webového portálu bude nutné provést každý měsíc revalidaci vykonané práce zaměstnanců. Každý zaměstnanec bude po ukončení jednotlivých zakázek uvádět dobu, kterou mu daná zakázka zabrala. Na konci se sečtou odpracované hodiny každého zaměstnance za zakázky, které odpracoval. Uloží se do databáze s datem, kdy byla revalidace provedena. Na základě měsíčních revalidací bude vypočteno peněžní ohodnocení jednotlivých zaměstnanců.

Tato funkčnost bude realizována pomocí modulu crone[11]. Crone dokáže naplánovat funkce, které se mají provádět v určitou dobu. Crone na konci

každého měsíce zavolá funkci, která sečte odpracované hodiny u jednotlivých zaměstnanců, přidá k nim údaje zaměstnanců a uloží se do souboru na server s názvem mm-yyyy s aktuálním měsícem a rokem, ke kterému se revalidace váže.

3.2.7 Změna přihlašovacího hesla

Bude nutné umožnit uživateli změnit si heslo, a to ze dvou důvodů. Prvním důvodem je bezpečnost, zákazník by si měl průběžně měnit heslo, aby nedošlo k jeho prolomení. Druhým důvodem je zapomenuté heslo. Pokud zákazník zapomene své heslo, je nutné, aby zákazník mohl obdržet nové heslo.

Pro změnu hesla bude muset uživatel vyplnit staré heslo, aby se ověřilo, že se opravdu jedná o daného uživatele. V případě, že ověření bude úspěšné, uloží se nové heslo k uživateli do databáze. V opačném případě bude uživatel notifikován o chybném zadání stávajícího hesla.

Pokud uživatel zapomene své heslo, tak jediné možné ověření je skrze jeho e-mailovou adresu. Uživatel tedy zadá svou e-mailovou adresu, kterou používá pro přihlášení, a tato adresa bude ověřena, jestli je uvedena v databázi. Pokud e-mailová adresa v databázi uvedena je, vygeneruje se uživateli nové heslo. Nové heslo se bude generovat následovně: vygeneruje se 5 náhodných znaků a následovně jedno malé písmeno, jedno velké písmeno a nakonec jedna číslice. Takto vygenerované heslo se následně zašle uživateli na e-mail. V případě, že emailová adresa v databázi uvedena není, pošle se uživateli pouze upozornění o neexistující adrese.

Zasílání e-mailů se realizuje pomocí modulu nodemailer, v tomto modulu se dá využít jak vlastního mailového serveru, tak externího mailového klienta. Pro testování byla použita služba Gmail, kde se vytvořil testovací účet, ze kterého se zasílaly testovací e-maily.

```
var nodemailer = require('nodemailer')
var sendmail = function (mail, subject, text, callback) {

    var transporter = nodemailer.createTransport({
        service: 'Gmail',
        auth: {
            user: 'email address',
            pass: 'password'
        }
    })
    var mailOptions = {
        from: 'email address',
        to: mail,
        subject: subject,
        text: text
    }
    transporter.sendMail(mailOptions, function (err) {
        if(err)
            return console.error(err)
        return callback(null);
    })
}
exports.sendmail = sendmail;
```

Obrázek 3.6: Ukázka kódu zasílání e-mailů

Zde je ukázka mnou vytvořeného modulu, který zasílá e-maily. Nejdříve si načtu požadovaný externí modul nodemailer, který obsahuje funkcionality pro zasílání e-mailů. Následně vytvářím funkci, jež bude realizovat posílání e-mailů. Funkci si musím uložit do proměnné (zde sendmail), aby mohla být následně exportována, díky čemuž ji budu moci volat z ostatních funkcionalit.

Funkce sendmail má čtyři argumenty:

- mail- e-mailová adresa, na kterou bude e-mail zaslán
- subject - předmět e-mailu
- text - samotný obsah e-mailu
- callback

Callback se využívá v Node.js u všech funkcí. Callback tvoří z blokujících funkcí funkce neblokující, a to tím způsobem že funkce po dokončení zavolá callback, ve kterém předá parametry popř. error. Tím dá najevo, že proces,

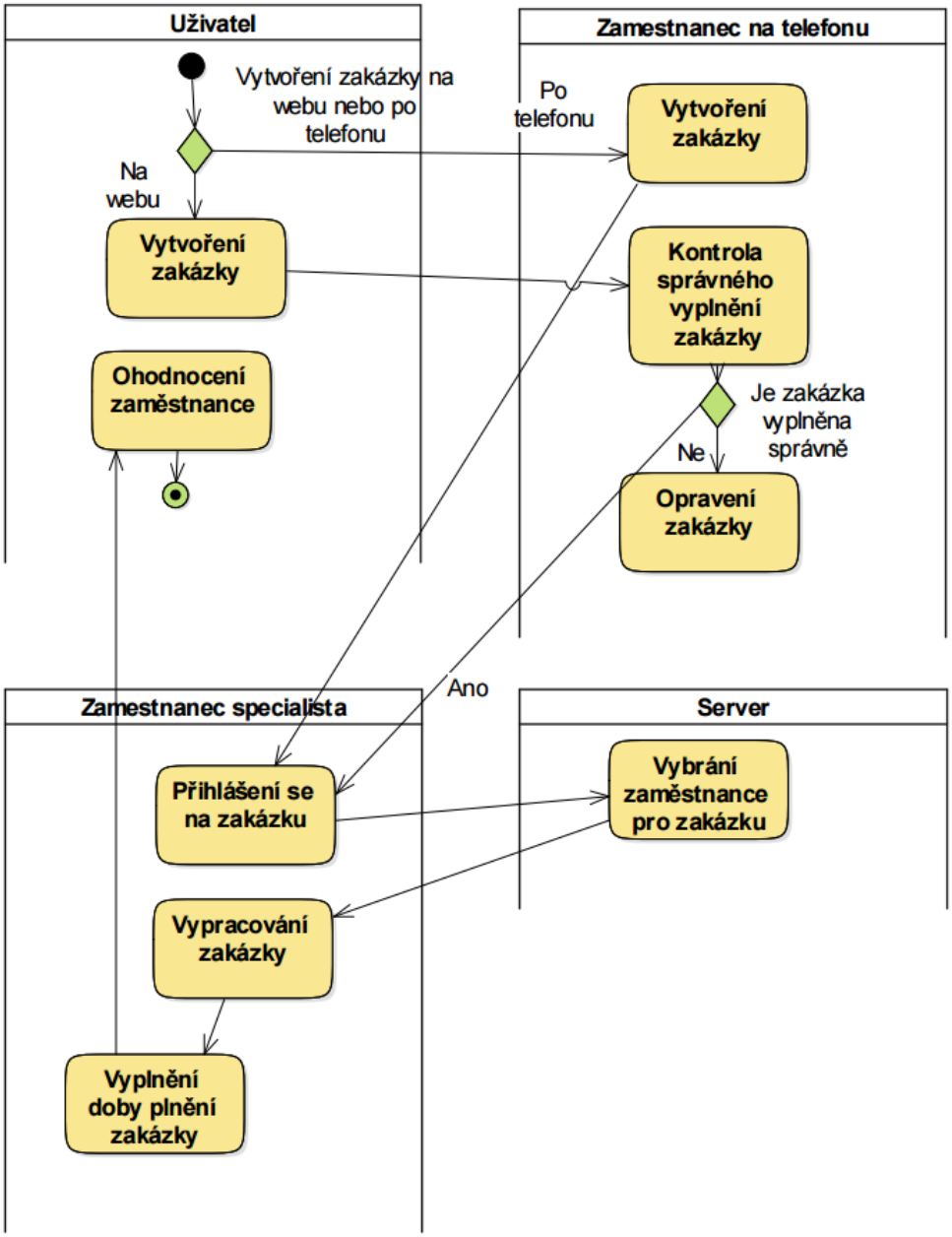
který funkci volal, může pokračovat. Toto způsobí, že během provádění funkce může proces nadále asynchronně pracovat.

Dále definuji transportér. Transportér je objekt, který bude následně schopný poslat e-mail. V transportéru je možno definovat mnoho parametrů, pro testování mi postačily parametry dva, a to:

- service : služba pro posílání emailů (Gmail)
- auth : e-mail a heslo do dané služby

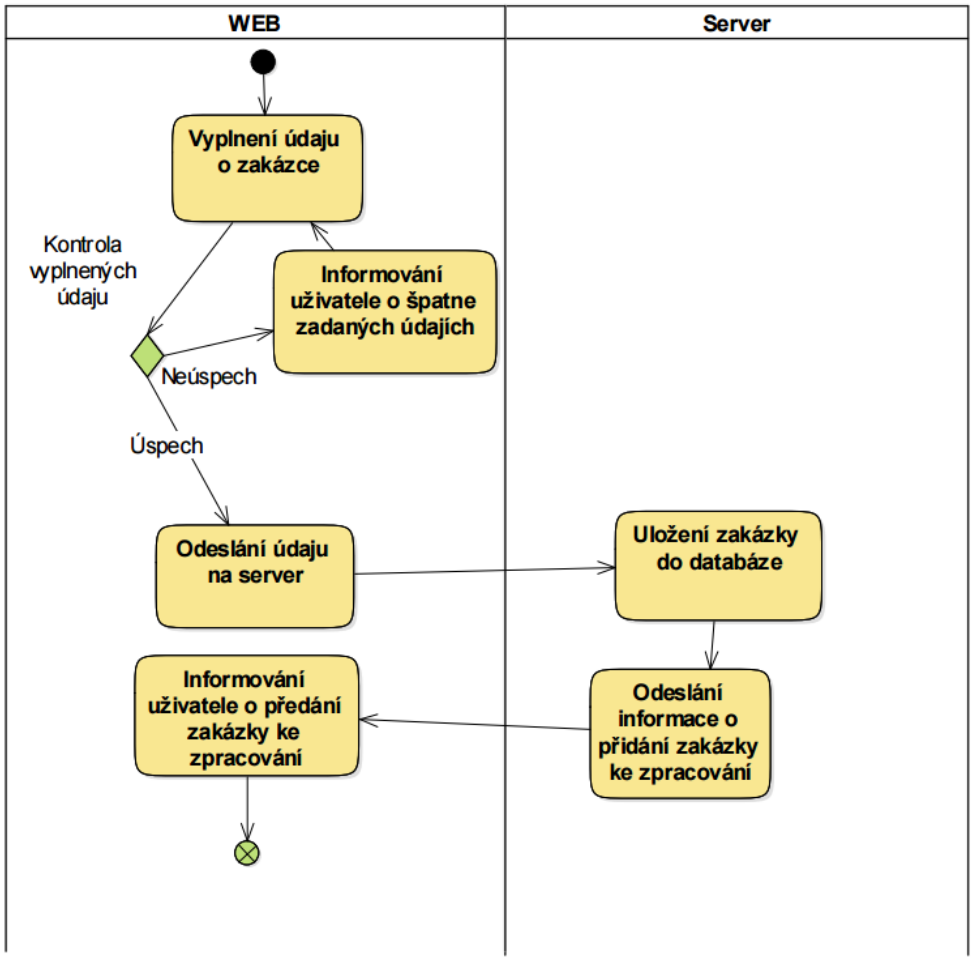
Následně definuji mailOptions, což už je samotný e-mail, který se odešle. Do mailOptions předám parametry, které byly do funkce předány. Navíc je potřeba přidat odesílatele, který musí být stejný jako e-mailová adresa v transportéru.

Na závěr proběhne samotné odeslání e-mailu pomocí zavolání funkce sendmail na transportér. Do funkce předávám parametry mailOptions, které jsem dříve definoval a callback tvořený funkcí s parametrem err. Pokud by při posílání e-mailu nastala chyba, např. by neexistoval adresát z callbacku, přijde error v parametru err, který se pak vypíše do konzole. V opačném případě funkce zavolá callback s parametrem null signalizujícím, že nenastala chyba.



Obrázek 3.7: Activity diagram Zakázka

3.2.8 Vytvoření zakázky



Obrázek 3.8: Activity diagram Vytvoření zakázky

Jak už bylo dříve zmíněno, vytvářet zakázku bude moci jak zákazník, tak zaměstnanec na telefonu ve spolupráci se zákazníkem.

V obou případech je nutné vyplnit údaje zmíněné v tabulce zakázka, k těmto informacím následně specializaci resp. podspecializaci. Tato data následně obdrží server, který je vloží do příslušné tabulky.

3.2.9 Úprava údajů o zakázce

Tato funkcionalita bude využita v případě, že bude nutné změnit jakékoli údaje o zakázce.

Možnost upravit údaje o zakázce bude využívána zákazníky, kteří se rozhodnou vytvořit si zakázku sami na webovém portálu, ale to pouze do doby, než zakázka projde kontrolou u zaměstnance pracujícího na telefonu.

Tato funkcionalita bude zejména využita zaměstnanci na telefonu v případě, že některé údaje vyplněné zákazníkem nebudou zcela úplné. V takovémto případě zaměstnanec na telefonu zkontaktuje zákazníka a za jeho spolupráce zakázku upraví. Tato úprava by byla možná provádět i druhým způsobem, a to vymazáním staré zakázky a vytvořením nové. Zmiňovaným postupem by ale bylo nutné přepisovat i údaje, které budou korektní. Například u údaje "adresa plnění zakázky"můžeme počítat s tím, že zákazník ví, na které adrese potřebuje pomoci a je tedy zadaná korektně.

Opět jako u změny údajů o uživateli, se při zasílání změněných údajů o zakázce na server pošlou nezměněná data jako false, ostatní data se pošlou tak, jak je uživatel zadá. Jednotlivé údaje se pak změní v databázi.

3.2.10 Přiřazování zaměstnanců k zakázkám

Pro to aby mohla být zakázka vypracována, je nutné k ní přiřadit zaměstnance specialistu. Tento zaměstnanec poté navštíví zákazníka a pomůže mu vzniklý problém vyřešit. Přiřazování zaměstnance bude probíhat ve dvou krocích.

Prvním krokem v přiřazování bude, že od zkontrolování zakázky zaměstnancem na telefonu popř. vytvoření zakázky zaměstnancem na telefonu budou mít zaměstnanci čas do půlnoci dalšího dne projevit zájem o zakázku, a to přihlášením se k zakázce.

V druhém kroku se vybere ze zaměstnanců, kteří se o zakázku přihlásili, ten, který má nejméně odpracovaných hodin. Tento zaměstnanec pak bude přiřazen k zakázce. Tímto se docílí rovnoměrného rozdělení zakázek mezi zaměstnance.

Druhý krok bude realizován pomocí výše zmiňovaného modulu cron. Cron bude každý den o půlnoci spouštět funkci, která vyhledá zakázky, které byly zkontrolovány nebo vytvořeny zaměstnancem na telefonu předešlý den, a u těchto zakázek pak vybere zaměstnance podle kritérií zmiňovaného v minulém odstavci.

3.2.11 Dokončení zakázky

Po dokončení zakázky zaměstnancem bude mít zaměstnanec u sebe dokument, do kterého doplní čas plnění zakázky a následně ho zákazník podepíše, čímž dá souhlas s délkou práce. Tímto se docílí, že si zaměstnanec nebude moci

zadávat více odpracovaných hodin, než ve skutečnosti odpracoval. Na základě tohoto dokumentu pak zaměstnanec vyplní údaj o době plněni zakázky do systému, který ho uloží k dané zakázce.

3.2.12 Přidání hodnocení

Zákazníci budou poskytovat zpětnou vazbu v podobě hodnocení zaměstnance, který jim pomáhal řešit jejich problém. Toto bude realizováno tak, že po dokončení zakázky bude zákazník vyzván, aby ohodnotil daného zaměstnance.

Toto hodnocení bude mít dva parametry. První parametrem bude samotné hodnocení, které se bude pohybovat v rozpětí 1-5 bodů (hvězdiček), druhým volitelným parametrem bude komentář. Komentář bude sloužit k tomu, aby uživatelé mohli vysvětlit jejich hodnocení.

Zákazník vyplní hodnocení popř. komentář. Tato data se zašlou na server a uloží se do databáze. Toto hodnocení se musí uložit i s vazbou na danou zakázku, zaměstnance a samozřejmě na samotného zákazníka.

3.2.13 Selekce dat z databáze pro frontend

Frontend bude využívat data z databáze, a to tak že se na ně bude dotazovat na předem určených odkazech. Pokud tedy frontend bude potřebovat např. detail určitého zaměstnance, pošle požadavek na server, přesněji na https://server:port/api/lectors/:id, kde id je uživatelské id zaměstnance. Server pak vybere data provázané s tímto zaměstnancem, vloží je do JSON struktury, jejíž formát je předem domluvený, a odešle je na frontend. Struktura pro tento požadavek vypadá následovně:

- id: uživatelské id zaměstnance
- name: křestní jméno
- surname: příjmení
- email: e-mail
- photo: fotografie (popř. defaultní avatar)
- averageRating: průměrné hodnocení
- school: jméno univerzity
- faculty: jméno fakulty
- department: obor studia
- level: bakalářské/magisterské/doktorandské studium
- grade: ročník
- skills: pole specializací
  - type: zkrácený název specializace
  - title: název specializace
  - skilled: označení, jestli zaměstnanec specializaci ovládá (ovládá alespoň jednu podspecializaci)
  - subcategories: pole podspecializací pro danou specializaci
    - type: zkrácený název podspecializace
    - title: název podspecializace
    - skilled: označení, jestli zaměstnanec podspecializaci ovládá
- reviews: pole hodnocení, jež zaměstnanec obdržel
  - id: id hodnocení
  - content: komentáře k hodnocení
  - rating: počet obdržených hvězdiček
  - autor: jméno autora

Všechny funkcionality, které bude frontend využívat jsou uvedeny zde:

- Získání informací o zaměstnanci
- Získání seznamu zaměstnanců
- Získání detailu specializace
- Získání profilu uživatele

Funkcionalita získání informací o zaměstnanci je uvede výše v ukázce komunikace.

V seznamu zaměstnanců budou informace o jednotlivých zaměstnancích. Tyto informace se budou skládat z části osobních dat a dále z jednotlivých specializací, které zaměstnanec ovládá či nikoliv. Tato data se budou zobrazovat v přehledu zaměstnanců pro zákazníky.

U získání detailu specializace mohou nastat dva případy: buď se jedná o specializaci nebo podspecializaci. V obou případech se data budou skládat z informací, ať už o specializaci nebo podspecializaci, a seznamu zaměstnanců, kteří ji ovládají. V případě, že se bude jednat o specializaci, připojí se k těmto datům i seznam všech podspecializací, které daná specializace obsahuje. Tato data se budou využívat, pokud si některý návštěvník webového portálu bude chtít prohlédnout jednotlivé specializace.

Funkcionalita získávání profilu uživatele se bude lišit u zákazníků a zaměstnanců. Pouze osobní informace budou poskytnuty v obou funkcionalitách, ale jelikož zaměstnanec má některé informace navíc, bude nutné je odlišit.

U zaměstnance se profil bude skládat z následující částí:

- Osobní informace
- Specializace s podspecializacemi
- Hodnocení
- Zakázky k výběru
- Přiřazené zakázky

V části osobní informace se budou vyskytovat všechny informace o zaměstnanci. Specializace s podspecializacemi bude obsahovat všechny existující specializace i podspecializace s indikátorem, zda ji zaměstnanec ovládá, či nikoliv. V sekci hodnocení budou všechna hodnocení, které obdržel. Zakázky k výběru budou obsahovat volné zakázky, které mají specializaci, již zaměstnanec ovládá, a ještě nebyly nikomu přiřazeny. Přiřazené zakázky jsou zakázky, o které zaměstnanec požádal, nebo mu již byly přiřazeny.

U zákazníka se profil bude skládat z těchto sekcí:

- Osobní informace
- Členství
- Dřívější zadané zakázky
- Aktivní zakázky
- Zakázky k hodnocení

Osobní informace budou obsahovat všechny informace o zákazníkovi. Členství se bude skládat ze dvou složek, jedna bude obsahovat informaci o stavu aktuálního členství a druhá bude zobrazovat, kdy v historii byl zákazník členem. Dříve zadané zakázky jsou zakázky, které již byly splněny a nejsou nadále aktivní. Aktivní zakázky obsahují zakázky, jež jsou přiřazeny zaměstnanci a čekají na vyhodnocení nebo čekají na přiřazení zaměstnance. Sekce "zakázky k hodnocení"se bude skládat ze zakázek, které jsou již vypracované a čekají na hodnocení zákazníka.

Jednotlivé funkcionality bude možné využívat i samostatně. To z důvodu, že pokud se změní informace pouze v nějaké části, je zbytečné získávat opět všechny části profilu, ale stačí pouze ta, ve které změna nastala.

3.2.14 Testování

Testování jsem prováděl postupně během vytváření jednotlivých funkcionalit. Na závěr jsem pak testoval funkcionality jako celek a to tak, že jsem nejdříve pomocí funkcionalit jako je vytvoření zakázky a vytvoření uživatelských účtů vytvořil data, díky kterým jsem pak mohl testovat funkcionality, které jsou na těchto datech závislé.

Závěr

Výsledkem této bakalářské práce je funkční backend skládající se z databáze a jednotlivých funkcionalit. Jednotlivé části byly otestovány a jsou připraveny na propojení s ostatními částmi vyvíjených v rámci projektu ITStudentsHelp portál. Na kterých pracovali Igor Kulka a Štefán Töltési.

Spolupráce probíhala tak, že Igor Kulka použil Apiary pro získávání statických dat pro svůj frontend. Z těchto statický dat jsem čerpal informace o struktuře dat, které buď backend bude přijímat nebo naopak odesílat. Podle struktury těchto dat jsem následovně tvořil jednotlivé funkcionality. Štefán Töltési využívá některých funkcionalit vytvořených mnou pro jeho administrátorskou část. Díky Apiary věděl, jaká data má do těchto funkcionalit posílat, a jak je strukturovat. Zároveň jsme pro sdílení kódu využili GIT. Tato spolupráce byla výhodná, jelikož každý věděl, jaká data má očekávat a jaká data by měl odesílat. Nejasnosti případné problémy se pak řešily na osobních schůzkách. S oběma členy týmu se vcelku dobře spolupracovalo.

Pro tvorbu databáze byla použita technologie PostgreSQL, pro funkcionality bylo využito Node.js a modulů, které Node.js obsahuje. Komunikace mezi backendem a frontendem je zprostředkovávána pomocí protokolu HTTPS. Přenášení dat mezi frontendem a backendem je realizováno za pomoci struktury JSON.

V budoucnu by se měly propojit jednotlivé části vyvíjené v rámci projektu ITStudentsHelp. Před samotným nasazením je ještě potřeba zajistit mnoho věcí, např. zajistit firmu, která se bude starat o finanční transakce.

Literatura

[1] Studentaanhuis [online], [seen 2016-03-28] Dostupné z: https://www.studentaanhuis.nl/

[2] Studentathome [online], [seen 2016-03-28] Dostupné z: http://www.studentathome.co.uk/

[3] Web Databases: Introduction to Relational & Object-Oriented Databases [online]. Janette B. Bradley, 1996 [seen 2016-04-23]. Dostupné z: http://www.axswave.com/weblibry/relobjdb.htm

[4] Relační vs. objektově-relační vs. objektové databáze [online]. Michal Batko [seen. 2016-04-24]. Dostupné z: http://www.fi.muni.cz/~xbatko/oracle/compare.html

[5] Database Limits and Possibilities [online]. gpence, 2015-02-27 [seen 2016-04-24]. Dostupné z: http://www.bleepingcomputer.com/forums/t/563884/database-limits-and-possibilities/

[6] Which programming language you should use for a web backend [online]. Robert Zaremba, 8.5.2013 [seen 2016-04-27]. Dostupné z: http://rz.scale-it.pl/2013/03/08/which_programming_language_should_you_use_for_a_web_backend.html

[7] SQL Injection a zabezpečení [online]. Petr Vojáček, 23.04.2007 [seen 2016-05-10]. Dostupné z: http://programujte.com/clanek/2007041802-sql-injection-a-zabezpeceni/

[8] Proč používat PostgreSQL [online]. Marek Olšavský, 01.09.2004 [seen 2016-03-08]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=354

[9] Why The Hell Would I Use Node.js? A Case-by-Case Tutorial [online]. Tomisla Capan, [seen 2016-03-20]. Dostupné z: https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

Příloha **A**

Obsah přiloženého CD

- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS