Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

Dokumentácia

Service Hero

**Dávid Majerčák**

Študijný program: Informatika

Ročník: 2

Predmet: Databázové systémy a algoritmy

Vedúci projektu: Ing. Tomáš Kramár

Ak. rok: 2016/2017

# Zadanie

Vo vami zvolenom prostredí vytvorte databázovú aplikáciu, **ktorá komplexne rieši nižšie definované scenáre** vo vami zvolenej doméne. Presný rozsah a konkretizáciu scenárov si dohodnete s Vašim cvičiacim na cvičení.

V poslednom kontrolnom bode musí aplikácia realizovať tieto všeobecné scenáre (a podscenáre):

* Zobrazenie prehľadu všetkých záznamov (rozumne stránkovaných, ak je to potrebné), kde v každom riadku figuruje aj číslo, pre ktorého výpočet sa využíva GROUP BY,
  + extend podscenár: Filtrovanie záznamov spĺňajúcich určité kritériá zadané používateľom.
* Zobrazenie konkrétneho záznamu - detail,
* Vytvorenie nového záznamu,
* Aktualizácia existujúceho záznamu,
* Vymazanie záznamu.

Aplikácia musí mať grafické rozhranie (aplikácia **nemôže** byť realizovaná ako konzolová). Je dôležité aby scenáre boli realizované realisticky - teda aby aplikácia (a teda aj jej používateľské rozhranie) naozaj poskytovala časť funkcionality tak, ako by ju očakával zákazník v danej doméne (napr. namiesto zadavania ID záznamu do modálneho okna si záznam vyberiem kliknutím v tabuľke).

Scenáre, ktoré menia dáta musia byť realizované **s použitím transakcií** (explicitné transakcie, autocommit) a aspoň jeden z nich musí zahŕňať **prácu s viacerými tabuľkami** (typicky vytvorenie záznamu a naviazanie cudzieho kľúča).

# Špecifikácia scenárov

## Vytvorenie používateľov

Najdôležitejšou časťou aplikácie bude vytvorenie používateľov, teda možnosť registrácie a prihlásenie, s ich rolami a ďalšími objektami im prislúchajúcimi.

Medzi subjekty interagujúce s aplikáciou patrí administrátor, podnikatelia, zamestnanci a v neposlednom rade, tí ktorí sú najdôležitejší, zákazníci. Každý z nich musí mať prístup iba k určitej množine dát, poprípade dátam ktoré na používateľa referencujú poprípade používateľ má referenciu na nich.

Z dôvodu zvýšenia bezpečnosti bude musieť byť heslo, každého používateľa, v databáze uložené ako enkryptovaný reťazec znakov vytvorených z hesla.

Používateľa s rolou **administrátor (administrator)** bude možné vytvoriť iba pomocou využívaného framework-u, keďže chceme znížiť počet možných miest v ktorých môže v systéme nastať chyba. Vytvorenie priamo v databáze možné nebude, respektíve nebude účinné, keďže je potrebné použiť enkrypciu použitú v aplikácii.

Ďalším používateľom je **vlastník podnikov (business owner)**, ktorý vlastní podniky s verejnými službami, ktoré bude ponúkať na portáli. Pri tomto type používateľa je potrebné nielen meno a heslo ale aj fakturačná adresa a adresa trvalého bydliska, respektíve adresa na ktorú si chce nechať posielať dodatočné dokumenty v prípade potreby.

Najdôležitejšou persónou v rámci systému však ostáva **zákazník (customer)**. On je ten ktorého potreby sa snaží portál uspokojiť širokou škálou poskytovaných služieb v jeho okolí, prípadne miesta, ktoré sa chystá navštíviť a ponúknuť mu všetky nevyhnutné služby k maximálnemu pohodliu. Pri scenári registrácie tohto používateľa bude potrebné, ako v prípade vlastníka podniku, vyplniť fakturačnú adresu a pre prípad, keď je služba, napríklad kosenie trávnika, vykonávaná priamo u klienta aj adresu doručenia služieb.

Podnik či firma môže fungovať aj čisto vďaka podnikateľovi-živnostníkovi, avšak iba vďaka nemu nemôže firma expandovať a potrebuje zamestnancov. Realizovaný sú pomocou entity **zamestnanec (employee)**. Týchto má možnosť registrovať zamestnávateľ priamo cez svoj účet poslaním pozvánky na firemný email a iba vďaka tejto pozvánke sa bude dať zaregistrovať ako zamestnanec firmy.

## Editácia profilu

V rámci portálu ma každý používateľ dostupný svoj profil, avšak časom prídu zmeny a údaje, ktoré boli vložené pri registrácii sa stanú neplatnými a zastaralými. Z tohto hľadiska je dôležité aby mal každý zákazník možnosť údaje vo svojom profile editovať.

Zároveň však môže nastať situácia že používateľ už naďalej nebude chcieť využívať portál. V tomto prípade bude môcť vymazať svoje používateľské konto aj zo všetkými údajmi. V prípade tohto druhého scenára bude potrebné zmazať minimálne nalinkované adresy, prípadne aj ďalšie entity ak to okolnosti vyžadujú.

Všetka táto funkcionalita musí byť realizovaná čo najjednoduchšie a najintuitívnejšie. To znamená, že používateľ bude môcť editovať svoj profil po kliknutí tlačidla „Settings“ v menu obsahujúcom akcie prislúchajúce účtu. Odtiaľ sa mu zobrazí formulár v ktorom sa budú dať upravovať základné položky, prislúchajúce profilu ako aj adresy prislúchajúce používateľovi.

## Najviac profitujúce firmy

Obsahom scenára je poskytnúť prehľad administrátorovi o najviac profitujúcich biznisoch na portáli. Tieto sa mu vypíšu ihneď po prihlásení na úvodnej stránke.

Cieľom tejto funkcionality bude, možnosť sa zamerať na kľúčových hráčov portálu a pridať ich scenáre použitia medzi testovacie aby z dôsledku implementácie novej funkcionality, nebol ohrozený takto dôležitý zákazník.

Vypisovanie bude riešené jednoduchou tabuľkou s piatimi až desiatimi položkami a možnosťou stránkovania. Všetky tieto úkony musia prebiehať asynchrónne aby bol používateľ čo najmenej rušený opätovným načítavaním aplikácie.

Zároveň musí byť umožnené vyhľadávať medzi firmami podľa názvu a tak zistiť zisk každej jednej firmy.

## Pohľad nad právnickým subjektom

Administrátor a vlastník spoločnosti, prípadne zamestnanec, budú mať prístup na pohľad nad firmou. Tento bude zahŕňať základné informácie ako meno podnikateľského subjektu a dátum vytvorenia. Spolu aj s adresami a to konkrétne fakturačnou a doručovacou. Dynamicky načítavané budú štatistiky o danom podniku, ktoré zahŕňajú celkový profit, profit za posledný fakturačný rok, profit za posledný fakturačný mesiac, počet ponúkaných služieb a v neposlednom rade rast za posledný rok a mesiac.

Takáto štatistika musí byť načítavaná asynchrónne, z dôvodu možnej časovej náročnosti na výpočet a teda oneskorenie načítania stránky. Implementácia musí byť pripravená na budúce zmeny, konkrétne pridávanie ďalších dôležitých polí.

## Editácia právnického subjektu

Informácie v profile firmy sa môžu, tak isto ako v prípade používateľa, časom meniť. Preto je nutné zabezpečiť, aby ľudia na to poverený, mohli daný podnikateľský subjekt či už editovať alebo priamo vymazávať ak sa zmenili údaje prípadne daný podnikateľský subjekt zanikol.

Navigácia k takejto editácii musí byť jednoduchá a intuitívna. Z tohto dôvodu majú možnosť či už administrátor alebo vlastníci podnikateľského subjektu priamo prístup k pohľadu na tento podnikateľský subjekt, ktorý zároveň zahŕňa tlačidlá na editáciu alebo vymazanie konkrétnej entity.

V prípade druhého scenára, vymazanie entity, bude potrebné prihliadať aj na vymazanie úzko previazaných entít medzi ktoré patria napríklad adresy prislúchajúce podnikateľskému subjektu.

## Posledné realizované objednávky

Pre vlastníka je dôležité mať prehľad nie len nad jednou ale najlepšie nad všetkými firmami. Z tohto dôvodu bolo dôležité implementovanie pohľadu na posledné realizované objednávky nad všetkými podnikateľskými subjektami prislúchajúcimi aktuálnemu používateľovi.

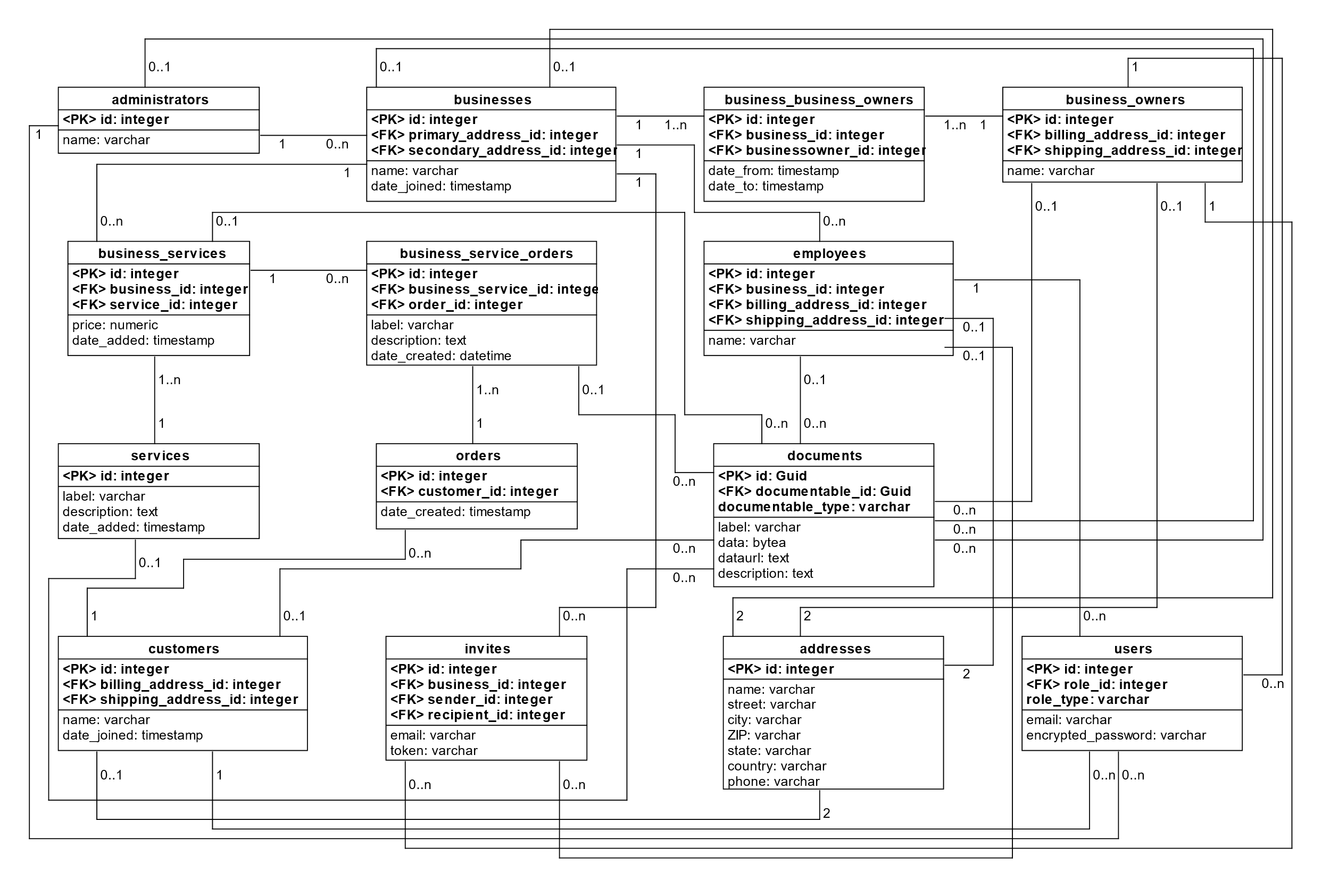
Tento pohľad bude potrebné implementovať priamo na „dashboard“ vlastníka podnikov aby mal prehľad o posledných objednávkach priamo po prihlásení. Z dôvodu možnosti dlho trvajúcej operácie nad databázou by mala byť táto operácia realizovaná dynamicky a asynchrónne.

## Moje firmy

Pre vlastníka firiem je veľmi dôležité dostať základný prehľad o všetkých firmách v ktorých je vlastníkom. Z tohto dôvodu by mal byť zoznam takýchto firiem dostupných ihneď po prihlásení. Daný pohľad bude realizovaný pomocou tabuľky, ktorá bude načítava asynchrónne. V nej sa budú nachádzať údaje ako meno firmy, aktuálny celkový profit a tlačidlo. Toto tlačidlo presmeruje používateľa na pohľad priamo nad konkrétnou firmou.

Pomocou tohto pohľadu bude používateľovi zobrazená štatistika firmy a základné údaje. Takisto bude mať možnosť ďalšej postupnosti na editáciu firmy alebo jej vymazanie z portálu.

# Dátový model



## Opis dátového modelu

Na obrázku vyššie môžeme vidieť fyzický dátový model. Tento model predstavuje spôsob uloženia dát v databáze a zároveň aj vzťahy s kardinalitami medzi nimi. V prípade aplikácií slúži ako základný kameň a má byť odvodený na základe scenárov použitia aplikácie.

Keďže daný model je zložitejší, nižšie môžete nájsť jeho opis. Pre lepšiu prehľadnosť bude model opisovaný zhora-dolu a sprava doľava.

### Administrators

Táto tabuľka a aj entita slúži výhradne pre účely autorizácie, reprezentáciu role administrátora. Myslí sa však aj na budúcnosť a to v prípade ak by bolo potrebné do tabuľky vložiť ďalší atribút. Vtedy nemusíme prepisovať tabuľku reprezentujúcu používateľa a teda všetky role ale iba tabuľku reprezentujúcu danú rolu.

Aj keď tabuľka neobsahuje cudzie kľúče, avšak odkaz na entity z tejto tabuľky môžeme nájsť v tabuľke „users“, keďže predstavuje rolu tohto používateľa.

### Businesses

Záznamy všetkých firiem, ktoré sú na portáli zaregistrované, môžeme nájsť v tejto tabuľke. Jedná sa však iba o základné údaje a to „name“, čiže meno danej firmy, a „date\_joined“, predstavujúci dátum, kedy sa firma registrovala na portáli. Pri atribúte „name“ je dôležité ešte podotknúť, že systém nerobí kontrolu či mená dvoch firiem nie sú náhodou totožné. To isté platí aj pre firmy každého používateľa, čiže jeden „business\_owner“ môže mať zaregistrované dve firmy s totožným menom.

Obsahom každého riadku sú zároveň dva cudzie kľúče na tabuľku „addresses“, kde jeden určuje fakturačnú adresu firmy a druhý adresu na doručovanie.

### Business\_business\_owners

V prvom rade sa jedná o väzobnú tabuľku na prepojenie entít, ako už názov napovedá, „business“ a „business\_owner“. Dôvod prepojenia je jednoduchý, pretože jedna fyzická osoba môže byť majiteľom, respektíve spolumajiteľom viacerých právnických subjektov, respektíve naopak a teda jeden právnický subjekt môže mať ako vlastníkov viaceré fyzické osoby.

Z predchádzajúceho odseku vyplýva, že každý riadok obsahuje dva cudzie kľúče. Prvý „business\_id“ ukazujúci na prislúchajúci riadok v tabuľke „businesses“ a druhý „business\_owner\_id“ ukazujúci na konkrétny záznam v tabuľke „business\_owners

“

### Business\_owners

Jedná sa o rolu používateľa, konkrétne fyzickú osobu, ktorá je vlastníkom alebo spoluvlastníkom právnického subjektu. Pre systém však predstavuje v prvom rade rolu pre použitie v autorizácii. Jediným atribútom, ktorý neodkazuje na inú tabuľku je meno.

Avšak odkazuje až na dve ďalšie entity a to konkrétne dve adresy. Prvú fakturačnú adresu právnickej osoby a druhú v prípade ak je iná tak adresa na doručovanie.

### Business\_services

Daná tabuľka uchováva údaje o jednotlivých službách ponúkaných entitou „business“, ktorá ukazuje na položky v tejto tabuľke. Dôležité atribúty obsiahnuté v tabuľke sú cena danej služby a dátum kedy bola daná služba, daným právnickým subjektom pridaná do systému.

Cudzie kľúče na ktoré odkazuje každý riadok tabuľky sú dva. Prvý „business\_id“ predstavujúci právnický subjekt ktorý danú službu zaradil do svojej ponuky. Druhý „service\_id“ predstavuje službu, ktorú právnicky subjekt pridal do svojej ponuky.

### Business\_service\_orders

Entita spájajúca dokopy „business\_service“, na ktorý zároveň obsahuj aj referenciu v podobe „business\_service\_id“, a „order“, na ktorý odkazuje referenciou v podobe „order\_id“. Jednoducho povedané spája službu definovanú podnikateľským subjektom a objednávku. Obsahuje aj nereferencujúce atribúty a to „label“, čiže krátky opis objednávky, a „description“, kde sa môže nachádzať správa od zákazníka alebo iný dlhší opis objednávky. Zároveň obsahuje aj najdôležitejší údaj a to kedy bola daná objednávka vykonaná.

### Employees

Ako už názov napovedá jedná sa o tabuľku ktorá slúži hlavne pri autorizácii a definuje ďalší typ role, zamestnanca. Obsahuje referencie na „business“, teda právnický subjekt pre ktorý daný zamestnanec pracuje, a dve adresy. Jedna je fakturačná a druhá je adresa určená pre doručovanie.

### Services

Definuje určitú službu nachádzajúcu sa na portáli. Jednotlivá služba je definovaná atribútmi ako „label“ čo je názov služby, „description“, predstavujúci opis danej služby a „date\_added“, čiže dátum kedy bola daná služba pridaná na portál.

### Orders

Podľa názvu môžeme usúdiť, že ide o tabuľka ktorá bude obsahuje dáta ohľadom objednávok. Obsahuje aj jeden cudzí kľúč a ten predstavuje referenciu na entitu zákazníka. Takisto obsahuje atribút „date\_created“ a teda dátum vytvorenia danej objednávky.

### Documents

Každý riadok v tejto tabuľke obsahuje buď súbor samotný alebo referenciu na takýto súbor. Jedná sa o dodatočné súbory, ako napríklad faktúra k objednávke a podobne. Obsahuje cudzí kľúč entity ktorej patrí daný súbor a aj typ entity ako string.

Takýto súbor je reprezentovaný kombináciou atribútov „label“, čiže označenie súboru, „description“, opis súboru alebo dokumentu, „data“, ak je súbor zapísaný priamo v databáze tento atribút drží jeho obsah, a „dataurl“, ktorý predstavuje odkaz na externý súbor alebo dokument.

### Customers

Táto tabuľka obsahuje dáta k ďalšej roli v systéme a tou je zákazník. Obsahuje iba dva nereferenčné atribúty a to „name“, meno zákazníka, a „date\_joined“, dátum založenia účtu prípadne dátum od kedy je zákazníkom danej spoločnosti.

V každom riadku reprezentujúcom entitu môžeme nájsť aj dva referenčné kľúče. Jeden reprezentujúci fakturačnú adresu a druhý adresu na doručenie.

### Invites

Táto tabuľka slúži na ukladanie pozvánok poslaných zamestnancom alebo celkovo len na email adresu, patriacu fyzickej osobe ktorá pracuje pre právnickú osobu od ktorej pozvánku obdržala. Základné atribúty obsiahnuté v tejto tabuľke a entite sú „email“, čiže email adresáta a „token“, čo je hash unikátny pre danú pozvánku.

Každý záznam obsahuje aj cudzie kľúče a to konkrétne „business\_id“, referenciu na právnický subjekt pod ktorý bude pozvaný zamestnanec spadať, „sender\_id“, referenciu na právnickú osobu reprezentujúcu právnický subjekt, a „recipient\_id“, referenciu na zamestnanca ak už sa nachádza v databáze a chceme mu len umožniť prístup k ďalšiemu právnickému subjektu.

### Addresses

Daná tabuľka ukladá dáta reprezentujúce adresy každej inej entity. Medzi atribúty obsiahnuté v každom riadku patria „name“, meno kontaktnej osoby, „street“, ulica, „city“, mesto, „ZIP“, PSČ, „state“, štát v rámci USA, „country“, štát v rámci sveta, „phone“, telefónny kontakt, „longitude“ a „latitude“, koordináty adresy pre budúce vykresľovanie pozície v rámci mapy.

### Users

Jedná sa o jednu z najdôležitejších entít reprezentujúcu každého používateľa systému. Každý záznam musí obsahovať referenciu aj na konkrétnu rolu vybraného používateľa.

Medzi ostatné atribúty obsiahnuté v tejto tabuľke patria „email“, email používateľa, ktorý používa pre prihlásenie sa do systému a „encrypted\_password“, čiže hash hesla používateľa.

# Návrh a implementácia

## Programovacie prostredie

Na vývoj aplikácie bolo využitých niekoľko nástrojov. Keďže aplikácia mala pracovať s databázou bolo potrebné odlíšiť nástroje pre operácie nad databázou a nástroje na vývoj aplikácie.

V prípade nástroja používaného s databázou bol využívaný hlavne pgAdmin vo verzii 3. Tento ponúkal intuitívne a prehľadné rozhranie. Takisto poslúžil pre bezproblémovú interakciu s databázov a vyobrazenia štatistík danej databázy.

Ako programovací jazyk bol zvolený Ruby, čo implicitne podnieťuje využitie frameworku Rails. Na vývoj aplikácií pod týmto framework-om je navhodnejšie použiť IDE od spoločnosti Jetbrains a to konkrétne RubyMine. Toto IDE ponúka aj priame napojenie na databázu a vyobrazovanie uložených dát a celkovú schému. Zároveň v hlavnej funkcionalite je aj autocomplete, čo dokáže zvýšiť produktivitu a vývoj aplikácií je omnoho jednoduchší.

Vývoj aplikácie však nebol vždy bezproblémový. Osobne preferujem narábať s nástrojmi pri ktorých viem čo sa deje na pozadí, avšak toto sa nedá povedať o framework-u Rails, kde sa veľa veci vykonáva jedným príkazom a vývojár vidí iba black box, do ktorého nemá prístup. Použitím ďalších „gem-ov“, čiže externých balíčkov, sa level abstrakcie iba zvýši čo na jednej strane zrýchli vývoj avšak ak sa niečo pokazí jediné čo ostane vývojárovi je mať hlavu v smútku pretože s tým niekedy nespraví nič. Nájdenie riešenia v tomto prípade znamená stráviť zopár nocí pozeraním sa na kód a nenapísaním ani čiarky.

## Návrhové rozhodnutia

Ako bolo spomenuté už v sekcii vyššie na vývoj bol využitý framework Rails. Tento ponúka ORM mapovač nad dátami, ktorý umožnuje vykonávať pokročilé operácie nad dátami. Avšak pri zložitejších operáciach bolo jednoduchšie napísať „raw“ SQL select ako písať tieto príkazy pomocou ponúkaného ORM.

Tieto zložitejšie SQL príkazy zahŕňali zložitejšie select-y, ktoré sa obsahovali „join“ nad viacerými tabuľkymi a vyžadovali si vyťahovanie iba určitých údajov. Takým príkladom je štatistika spoločnosti v aplikácií pri jej vyobrazení. Konkrétny príklad môžere vidieť nižšie.

**def** *get\_business\_statistics* **@profit** = ***Business***.select(**"sum(business\_services.price)"**)  
 .*joins*(**"JOIN business\_services ON business\_services.business\_id = businesses.id"**)  
 .*joins*(**"JOIN business\_service\_orders ON business\_service\_orders.business\_service\_id = business\_services.id"**)  
 .*where*(**"businesses.id=?"**, **"**#{**@business**[**:id**]}**"**)  
 **@annual\_profit** = ***Business***.select(**"sum(business\_services.price)"**)  
 .*joins*(**"JOIN business\_services ON business\_services.business\_id = businesses.id"**)  
 .*joins*(**"JOIN business\_service\_orders ON business\_service\_orders.business\_service\_id = business\_services.id"**)  
 .*where*(**"businesses.id=? and business\_service\_orders.date\_created>=? and business\_service\_orders.date\_created<?"**, **"**#{**@business**[**:id**]}**"**, 1.years.ago.beginning\_of\_year, ***Time***.now.beginning\_of\_year)  
 **@monthly\_profit** = ***Business***.select(**"sum(business\_services.price)"**)  
 .*joins*(**"JOIN business\_services ON business\_services.business\_id = businesses.id"**)  
 .*joins*(**"JOIN business\_service\_orders ON business\_service\_orders.business\_service\_id = business\_services.id"**)  
 .*where*(**"businesses.id=? and business\_service\_orders.date\_created>=? and business\_service\_orders.date\_created<?"**, **"**#{**@business**[**:id**]}**"**, 1.months.ago.beginning\_of\_month, ***Time***.now.beginning\_of\_month)  
 **@services\_count** = ***Business***.select(**"count(\*)"**)  
 .*joins*(**"JOIN business\_services ON business\_services.business\_id = businesses.id"**)  
 **@employees\_count** = ***Business***.select(**"count(\*)"**)  
 .*joins*(**"JOIN employees ON employees.business\_id = businesses.id"**)  
 **@annual\_growth** = **nil  
 if @business**.date\_joined <= 2.years.ago  
 **@last\_year\_profit** = ***Business***.select(**"sum(business\_services.price)"**)  
 .*joins*(**"JOIN business\_services ON business\_services.business\_id = businesses.id"**)  
 .*joins*(**"JOIN business\_service\_orders ON business\_service\_orders.business\_service\_id = business\_services.id"**)  
 .*where*(**"businesses.id=? and business\_service\_orders.date\_created>=? and business\_service\_orders.date\_created<?"**, **"**#{**@business**[**:id**]}**"**, 1.years.ago.beginning\_of\_year, 1.years.ago)  
 .as\_json[0][**'sum'**]  
 **@this\_year\_profit** = ***Business***.select(**"sum(business\_services.price)"**)  
 .*joins*(**"JOIN business\_services ON business\_services.business\_id = businesses.id"**)  
 .*joins*(**"JOIN business\_service\_orders ON business\_service\_orders.business\_service\_id = business\_services.id"**)  
 .*where*(**"businesses.id=? and business\_service\_orders.date\_created>=? and business\_service\_orders.date\_created<?"**, **"**#{**@business**[**:id**]}**"**, ***Time***.now.beginning\_of\_year, ***Time***.now)  
 .as\_json[0][**'sum'**]  
 **@annual\_growth** = (**@this\_year\_profit** - **@last\_year\_profit**)/**@last\_year\_profit  
 @annual\_growth** = **@annual\_growth**.round(4).to\_s + **"%"  
 end  
  
 @monthly\_growth** = **nil  
 if @business**.date\_joined <= 2.months.ago  
 **@last\_month\_profit** = ***Business***.select(**"sum(business\_services.price)"**)  
 .*joins*(**"JOIN business\_services ON business\_services.business\_id = businesses.id"**)  
 .*joins*(**"JOIN business\_service\_orders ON business\_service\_orders.business\_service\_id = business\_services.id"**)  
 .*where*(**"businesses.id=? and business\_service\_orders.date\_created>=? and business\_service\_orders.date\_created<?"**, **"**#{**@business**[**:id**]}**"**, 1.months.ago.beginning\_of\_month, 1.months.ago)  
 .as\_json[0][**'sum'**]  
 **@this\_month\_profit** = ***Business***.select(**"sum(business\_services.price)"**)  
 .*joins*(**"JOIN business\_services ON business\_services.business\_id = businesses.id"**)  
 .*joins*(**"JOIN business\_service\_orders ON business\_service\_orders.business\_service\_id = business\_services.id"**)  
 .*where*(**"businesses.id=? and business\_service\_orders.date\_created>=? and business\_service\_orders.date\_created<?"**, **"**#{**@business**[**:id**]}**"**, ***Time***.now.beginning\_of\_month, ***Time***.now)  
 .as\_json[0][**'sum'**]  
 **@monthly\_growth** = (**@this\_month\_profit** - **@last\_month\_profit**)/**@last\_month\_profit  
 @monthly\_growth** = **@monthly\_growth**.round(4).to\_s + **"%"  
 end** *render* **json**: {**"Profit"**: **@profit**.as\_json[0][**'sum'**], **"Annual Profit"**: **@annual\_profit**.as\_json[0][**'sum'**], **"Monthly Profit"**: **@monthly\_profit**.as\_json[0][**'sum'**], **"Services Count"**: **@services\_count**.as\_json[0][**'count'**], **"Annual Growth"**: **@annual\_growth**, **"Monthly Growth"**: **@monthly\_growth**}.as\_json(**except**: [**:id**])  
**end**

**Manuálne operácie SELECT**

Používaný framework robí automaticky každý select pomocou transakcií, čiže tieto nebolo potrebné definovať pre jednoduché operácia vykonávané nad jednou tabuľkou. Akonáhle však prišlo k vytváraniu používateľa bolo potrbné vložiť dáta do viacerých tabuliek. Príkladom môže byť vytvorenie právnickej osoby, „business owner“. Pri tejto sú potrebné dva zápisy do tabuľky „addresses“, jeden zápis do tabuľky „users“ a ešte jeden zápis do tabuľky „business\_owners“. Príklad implementácie takéhoto riešenia môžete vidieť nižšie.

**def** *create* **@business\_owner** = ***User***.new(user\_params)  
 **@business\_owner**.role = ***BusinessOwner***.new(business\_owner\_params)  
 **@business\_owner**.role.shipping\_address = ***Address***.new(shipping\_address\_params)  
 **@business\_owner**.role.billing\_address = ***Address***.new(billing\_address\_params)  
  
 ***BusinessOwner***.transaction **do  
 *User***.transaction **do  
 *Address***.transaction **do  
 @business\_owner**.role.shipping\_address.save!  
 **@business\_owner**.role.billing\_address.save!  
 **@business\_owner**.role.save!  
 **@business\_owner**.save!  
 **end  
 end  
 end** respond\_to **do** |*format*|  
 **if @business\_owner**.persisted?  
 *format*.html { *redirect\_to* dashboard\_path, **notice**: **'Your account was successfully created.'** }  
 *format*.json { *render* **:show**, **status**: **:created**, **location**: dashboard\_path }  
 **else** *format*.html { *render* **:new** }  
 *format*.json { *render* **json**: **@business\_owner**.errors, **status**: **:unprocessable\_entity** }  
 **end  
 end  
end**

**Použitie transakcií pri operáciach nad viacerými tabuľkami**

## Opis implementácie jednotlivých scenárov

### Vytvorenie používateľov

V rámci implementácie tohto scenára bolo potrebné pridať dva tlačidlá na úvodnú obrazovku a to „Customer sign up“ a „Business Owner sign up“. Už názov napovedá, že v prípade prvého pôjde o vytvorenie zákazníckeho účtu a v prípade druhého o vytvorenie účtu pre právnickú osobu.

Pre každý z týchto scenárov bolo potrebné napísať šablónu formuláru v html do ktorej bude môcť používateľ vyplniť údaje. Po potvrdení sa vytvorí objekt pre každú entitu a teda User, dvakrát Address a BusinessOwner, respektíve Customer. Následne je potrebné uložiť tieto dáta do databázy v rámci jednej transakcie. Po vykonaní sa používateľovi vypíše hláška o úspešnej registrácií a može začať s používaním účtu. V prípade neúspechu je presmerovaný späť na formulár, kde sa vyobrazí chybová hláška.

V prípade vytvorenia zamestnanca je proces o niečo zložitejší. Na realizáciu tohto procesu bolo potrebné pridanie tlačidla do drop-down menu aktuálneho používateľa ak sa jedná o právnickú osobu. Ďalej bolo potrebné vytvorenie šablóny s formulárom obsahujúcim textové pole pre email adresu na pozvanie používateľa a drop-down menu pre vybratie spoločnosti pod ktorú ho pridať. Po potvrdení, emailový klient odošle na vyplnenú emailovú adresu url pomocou ktorej sa môže zamestnanec zaregistrovať. Pre toto použitie bolo potrebné vytvorenie ďalšej šablóny pre registráciu zamestnanca. Ďalej scénar pokračuje tak isto ako predchádzajúci.

### Editácia profilu

Pri tomto scenári bolo potrebné spraviť šablónu na úpravu každej zvlášť s poliami typickými pre konkétnu rolu. Samozrejme, keďže pri editácii napríklad profilu právnickej osoby chceme mať možnosť editovať aj jej adresy, bolo potrebné zadefinovať v modeli „accept\_nested\_attributes\_for“ pre vhniezdené entity. Toto isté platí aj pre vytváranie pouťívateľa a role.

Po potvrdení úprav systém pokračuje v podobnej rutine ako pri vytvorení avšak s operáciou UPDATE nad databázou.

### Najviac profitujúce firmy

Jedná sa o prvý scénar kde bolo potrebné zakomponovať aj JavaScript na strane klienta. Na začiatku však bolo potrebné definovať cestu na ktorú sa klient bude dopytovať v súbore „routes.rb“. Následne bolo potrebné implementovať akciu vyvolanú cestou aj v controller-i.

V danej metóde sa vykonal SELECT nad databázou aby vrátil najviac profitujúce spoločnosti v rámci systému. Tieto dáta sa zabalili do JSON objektu s ktorým už JavaScript nemá problém pracovať.

Na strane používateľa boli tieto dáta dynamicky pretransformované do tabuľky. Používateľ mal možnosť stránkovania a vyhľadávania medzi spoločnosťami pomocou mena. Dopyty na server boli realizované pomocou AJAX-u.

### Pohľad nad podnikateľským subjektom

Môžeme ho nazvať takisto pohľad nad spoločnosťou. Tento má na starosti vypísanie základných informácii o spoločnosti ako aj štatistiku. Z pohľadu implementácie ide o zložitejší pohľad keďže kombinuje zobrazenie z databázových operácii synchrónne priamo do šablóny ale aj asynchrónne pomocou JavaScript-u.

Synchrónne operácie ktoré svoje výsledky zapisujú priamo do šablóny sú načítanie adries právnického subjektu ale aj základné informácie o subjekte ako jeho meno a dátum pridania do systému. Do základnej šablóny boli vložené aj tri extra tlačidlá. Prvé na editáciu spoločnosti, druhé na jej vymazanie a tretie na návrat späť na pohľad „dashboard“.

Asynchrónnu operáciu polo potrebné aplikovať na metódu ktorá vytvára komplexnú štatistiku o spoločnosti a vykonáva veľa SELECT operácii nad databázou. Výsledné dáta sa dynamicky vykreslia do HTML pomocou JavaScript-u. Dopyty na server sú realizované pomocou AJAX dopytov. Výsledok dopytu je klientovi doručený v JSON formáte.

### Editácia právnického subjektu

Pre realizáciu tohto scenára bolo potrebné vytvorenie šablóny obsahujúcej všetky polia pre všetky atribúty, ktoré obsahuje entita „business“. V prípade modelu bolo takisto potrebné zadefinovať „accept\_nested\_attributes\_for“ pre všetky vnorené atribúty entity, aby ich bolo možné či už vytvárať alebo editovať.

Po potvrdení zmien sa v controller-i otvoria transakcie nad všetkými tabuľkami, respektíve modelmi, ktoré bolo možné upraviť vo formulári. Následne sa vykoná uloženie a ak všetko prebehne úspešne, používateľ je informovaný hláškou o úspešnom vykonaní operácie. Inak je upozornený na chybu.

### Posledné realizované objednávky

Scenár je vykonávaný asynchrónne pomocou klientského JavaScript kódu. Ten sa dopytuje na zadefinovanú cestu v súbore „routes.rb“ pomocou AJAX dopytu. Na serveri je vykonaná operácia SELECT nad databázou a výsledok je vrátený vo formáte JSON. Na klientovi je následne dynamicky vygenerovaná tabuľka v JavaScript-e. Keďže tabuľka zobrazuje naraz iba 5 údajov funkcionalita zahŕňa aj pagination. Táto funkcionalita je implementovaná pomocou AJAX volaní s parametrami „offset“ a „limit“.

### Moje firmy

Posledný scenár avšak nie najmenej dôležitý je takisto realizovaný pomocou AJAX dopytov z klientského JavaScript kódu. Dopyty obsahujú atribúty „offset“ a „limit“, čo umožňuje implementáciu pagination funkcionality v prípade ak je používateľ vlastníkom alebo spoluvlastníkom viacerých právnických subjektov.

Tabuľka s výsledkami je dynamicky generovaná a dosadená pomocou JavaScript-u. Táto tabuľka obsahuje v každom jednom riadku aj tlačidlo ktoré slúži ako odkaz na pohľad nad daným právnickým subjektom.

# Záver

Vďaka projektu som mal možnosť sa stretnú s veľa, pre mňa novými, technológiami. Pred projektom som mal skúsenosť s prácou iba so základnými operáciami nad databázou. No vďaka práci na projekte a s PostgreSQL som mal možnosť pochopiť zložitejšie princípy vývoja a hlavne databázové modely. Prvýkrát sa mi teda podarilo mať „hands-on“ na „full-featured“ SQL databáze a vyskúšať si aspoň z časti funkcionalitu takýchto systémov.

Keďže som sa počas práce na projekte učil aj framework Rails, tak aj z tejto oblasti som nadobudol veľa nových skúsenosti. Mal som možnosť si vyskúšať aj nie priami písanie operácii SELECT ale realizovať operácie nad databázou pomocou ORM ktoré ponúka Rails framework.