VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRE ELEKTRONICKÚ EVIDENCIU TRŽIEB IMPLEMENTOVANÝ V LARAVEL

A SALES REGISTRATION SYSTEM IN THE LARAVEL FRAMEWORK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MICHAL GABONAY**
AUTHOR

VEDOUCÍ PRÁCE — **RNDr. MAREK RYCHLÝ, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/20062/2016/xgabon00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Gabonay Michal**
Obor: Informační technologie
Téma: **Systém pro elektronickou evidenci tržeb implementovaný v Laravel**
A Sales Registration System in the Laravel Framework
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s rámcem Laravel pro jazyk PHP. Zjistěte legislativní a technické požadavky na elektronickou evidenci tržeb (EET) a srovnejte existující open-source řešení EET.
2. Demonstrujte co nejširší použití Laravel na příkladu systému pro EET. Navrhněte systém umožňující registraci více podnikatelů a jejich oddělenou EET, dle legislativních a technických požadavků. Umožněte sledování historie tržeb; filtraci dle různých parametrů; výběr období, jejich porovnání a poznámkování; import a export dat; a rozhraní pro napojení na externí systémy (např. obecnou SMS bránu pro evidenci zprávami SMS). Zvažte integraci existujících open-source projektů.
3. Po konzultaci s vedoucím systém implementujte jako Laravel/PHP aplikaci s webovým rozhraním. Výsledný systém důkladně otestujte (využijte jednotkové testy v Laravel).
4. Zdokumentujte výsledky a zveřejněte projekt pod svobodnou licencí jako open-source.

Literatura:

- *etržby - elektronická evidence tržeb*. [http://www.etrzby.cz/]
- Ondřej Novák. *GitHub - ondrejnov/eet: EET Client for PHP*. [https://github.com/ondrejnov/eet]
- Arda Kilicdagi, H. Ibrahim Yilmaz. *Laravel design patterns and best practices: enhance the quality of your web applications by efficiently implementing design patterns in Laravel*. Packt, 2014. ISBN 978-1783287987. [https://www.packtpub.com/web-development/laravel-design-patterns-and-best-practices]

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Rychlý Marek, RNDr., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

V Českej republike sa v roku 2016 schválil nový zákon o evidencii elektronických tržieb. Podľa tohoto zákona musí každý podnik zaevidovať tržby na Finančnú správu Českej republiky a na doklad o zaplatení pridať FIK kód vrátený Finančnou správou. Táto práca popisuje vytvorenie systému pre správu spoločnosti a jej zamestnancov, elektronickú evidenciu tržieb, prezeranie si históriu tržieb, filtráciu, porovnávanie medzi sebou, poznámkovanie a ďalšie operácie s nimi. Ide o webovú aplikáciu určenú pre malo-podnikateľov, ktorí nepotrebujú zložité spoplatnené systémy.

Abstract

In Czech Republic, the new law about electronic revenue registration came into the force from the 1st of January 2017 . According to this law, each company has to register every revenue in the Czech Republic's Financial Office and add on the FIK code, the proof of payment which is returned every time from the Financial Office. This paper describes how to create a system for managing companies and employees, electronic revenue records, viewing revenue history, filtering, comparing, remarking, and other operations. This is a web app designed for small business owners who do not need complex charging systems.

Kľúčové slová

Web, webová aplikácia, informačný systém, PHP, Laravel, Google Charts, EET

Keywords

Web, web application, information system, PHP, Laravel, Google Charts, EET

Citácia

GABONAY, Michal. *Systém pre elektronickú evidenciu tržieb implementovaný v Laravel*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Rychlý Marek.

Systém pre elektronickú evidenciu tržieb implementovaný v Laravel

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána RNDr. Mareka Rychlého s použitím odbornej literatúry. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Michal Gabonay
14. mája 2017

Poďakovanie

Týmto by som sa chcel poďakovať vedúcemu bakalárskej práce, pánovi RNDr. Marekovi Rychlému Ph.D, za jeho pomoc a usmerňovanie. Taktiež sa chcem poďakovať za podporu svojej rodine a blízkym počas celého štúdia.

Obsah

Kapitola 1

Úvod

Povinnosť platiť dane je bežnou vecou už od nepamäti. Avšak krátenie daňovej povinnosti sa pre niektorých podnikateľov stala bežnou záležitostou aj tu, v Českej republike. Finančná správa nemá dostatok prostriedkov aby dokázala pri takom množstve podnikateľov efektívne dohliadať na správne platenie daní a uskutočňovať u každého daňové kontroly. Preto sa Česká republika v roku 2016 rozhodla zaviesť zákon, ktorý zaväzuje podnikatelov k elektronickej evidencii tržieb (EET).

Cieľom tejto bakalárskej práce bolo navrhnúť a vytvoriť informačný systém pre elektronickú evidenciu tržieb. Tento systém by mal byť schopný evidovať nové tržby na Finančnej správe, spravovať spoločnosti a ich zamestnancov z tohoto hľadiska, zobrazovať prehľadnú históriu tržieb pre analyzovanie, možnosť poznámkovať tržby a obdobia a taktiež import a export tržieb.

Tento dokument je rozdelený do niekoľkých kapitol, v ktorých sa popisuje tvorba tejto webovaj aplikácie od návrhu, cez implementáciu a popis jej súčastí až po otestovanie výslednej aplikácie. Prvú kapitolu predstavuje tento úvod, na ňu naväzuje druhá kapitola v ktorej si podrobnejšie popíšeme problém úlohy, vysvetlíme si, čo vlastne elektronická evidencia tržieb znamená, pozrieme sa na existujúce riešenia a uvedieme cieľ našej vytvorenej aplikácie. Nasledujúca, tretia kapitola, sa zaoberá výberom a popisom použitej technológie pri vytváraní webovaj aplikácie. Štvrtá kapitola sa venuje PHP frameworku Laravel, ktorý bol použitý na vytvorenie aplikácie pre túto bakalársku prácu, ukazuje kľúčové prvky a popisuje adresárovú štruktúru tohoto frameworku. V piatej kapitole sa rozoberá analýza a návrh informačného systému pomocou diagramov. Šiesta kapitola sa sústredí na popis implementácie kľučových častí aplikácie a testovaním aplikácie. Posledná kapitola je zhrnutie v závere doplnené o nápady na zdokonalenie a vylepšenie aplikácie a na konci je uvedený zoznam použitých referencií.

Kapitola 2

Úvod do problematiky

Úlohou tejto práce je vytvorenie webovej aplikácie pomocou ktorej sa budú môcť elektronicky evidovať tržby podľa legislatívnych a technických požiadaviek. Treba vopred upozorniť, že projekt nerieši tlač účtenky. Táto aplikácia má byť implementovaná v PHP frameworku Laravel za použitia existujúcich open-source projektov a tento projekt sám, má byť zverejnený pod slobodnou licenciou ako open-source. Nasledujúci popis a princíp EET je prevzatý a zhrnutý z oficiálnych stránok etrzby.cz [3].

2.1 Elektronická evidencia tržieb (EET)

Elektronická evidencia tržieb je spôsob, ktorým chce Česká republika bojovať proti daňovým únikom podnikateľov. V roku 2016 vošiel do platnosti zákon, ktorý zaväzuje elektronicky evidovať všetky tržby pochádzajúce z podnikateľskej činnosti (s výnimkou priameho prevodu z účtu na účet). Štát si od tohoto zákonu sľubuje veľké zmeny k lepšiemu, avšak tento zákon sa stretol aj s množstvom negatívnej odozvy. Za účelom plynulejšieho nábehu systému, Ministerstvo financií rozdelilo tento systém na štyri fázy, podľa toho, pre koho a kedy zákon začína platiť. Toto fázovanie slúži na zníženie záťaže v začiatkoch a taktiež poskytne podnikateľom dostatok času pripraviť sa na túto zmenu:

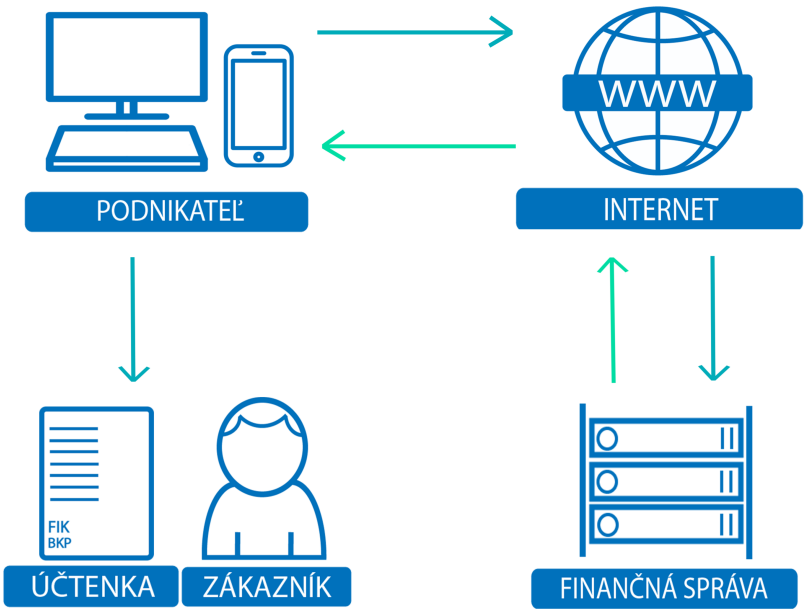
- 1. fáza – od 1. decembra 2016 - ubytovacie a stravovacie služby,
- 2. fáza – od 1. marca 2017 - maloobchod a veľkoobchod,
- 3. fáza – od 1. marca 2018 - ostatné činnosti s výnimkou tých vo 4. fáze, napríklad slobodné povolania, doprava, poľnohospodárstvo,
- 4. fáza – od 1. júna 2018 - vybrané remeslá a výrobné činnosti.

Systém evidencie tržieb je navrhnutý tak, aby bol otvorený, zo softvérového tak aj z hardvérového riešenia. To znamená, že nie je potreba používať konkrétne špeciálne zariadenia certifikované Ministerstvom financií. Jednotliví podnikatelia sa rozhodnú, aký systém použijú pre evidenciu tržieb. Je úplne jedno, aké zariadenia použije, napríklad pokladný systém, počítač alebo jednoduchý mobil. Stačí, aby to zariadenie bolo technicky vybavené tak, aby bolo schopné odoslať cez internet požadované informácie o evidovanej tržbe na server Finančnej správy a vytlačiť účtenku s fiskálnym identifikačným kódom (FIK). K účtenke sa priraďujú celkovo tri kódy: spomínaný fiskálny identifikačný kód (FIK), bezpečnostný kód poplatníka (BKP) a podpisový kód poplatníka (PKP).

- FIK je unikátny kód generovaný a zasielaný systémom Finančnej správy, ktorým potvrzuje prijatie dátovej správy o evidovanej tržbe.
- BKP je kód generovaný automaticky pokladným zariadením podnikateľa, ktorý útčuje jednoznačnú väzbu mezi účtenkou a podnikateľom, ktorý ju vydal.
- PKP je pomocným ochranným prvkom, ktorý umožňuje kontrolu integrity a dokazuje zodpovednosť podnikateľa za vystavenie účtenky. Je generovaný automaticky pokladným zariadením podnikateľa, na účtenku je však pridaný len v prípadoch, kedy na účtenku nie e možné z objektivních dôvodov pridať FIK.

Komunikácia medzi zariadením na evidovanie tržieb a serverom Finančnej správy prebieha v režime požiadavka/odpoveď (request/response). Princíp fungovania elektronického evidovania tržieb je znázornený na obrázku 2.1.

- 1. Podnikateľ zo svojho zariadenia odošle dátovú správu o transakcii vo formáte XML na server Finančnej správy,
- 2. zo systému Finančnej správy je zaslané potvrdenie o prijatí s unikátnym FIK kódom (v prípade neúspechu je zaslaná správa o chybe),
- 3. podnikateľ vystaví účtenku s platným FIK, ktorú odovzdá zákaznikovi,
- 4. zákazník dostane účtenku,
- 5. zákazník si môže overiť svoju účtenku na daňovom portále, podnikateľ si môže overiť svoje zaevidované tržby vo webovej aplikácii Elektronická evidence tržby.



Obr. 2.1: Princíp fungovania EET.

Aby bolo možné tržby evidovať, podnikateľ si musí vybaviť Certifikačnú autoritu EET. Túto je možné získať zo stránok daňového portálu. Tento certifikát sa použije na podpísanie dátových správ o evidencii tržieb.

Pre vývoj aplikácií umožnujúcich elektronickú evidenciu tržieb, Finančná správa vytvorila testovacie prostredie (PLAYGROUND). Na toto testovacie prostredie je možné normálne odosielať dátové správy o tržbe, avšak vrátený FIK je neplatný (čiže tržba nie je zaevidovaná, len sa tak tvári). Neplatný FIK sa liši tým, že posledné dva znaky má "ff".

- Príklad platného FIK kódu: b3a09b52-7c87-4014-a496-4c7a53cf9125-03
- Príklad neplatného FIK kódu: b3a09b52-7c87-4014-a496-4c7a53cf9125-ff

2.2 Existujúce riešenia

Ako sa dá predpokladať, väčšina systémov na elektronickú evidenciu tržieb je spoplatnená. Od veľmi prepracovaných, ktoré sa poskytujú s celým pokladničným systémom, po jednoduché, pekne spracované aplikácie pre Android a iOS. Niektoré takéto systémy sa navonok prezentujú ako bezplatné, ale v skutočnosti sa jedná o trial verzie. To znamená, že napríklad prvých 30 dní je zadarmo a potom si už pýtajú mesačný poplatok, alebo napríklad 50 vystavených účteniek na mesiac je zdarma a ak chcete viac, musíte si platiť.

Služba iDoklad [1] je rozsiahly účtovnický systém, kde je možné vystavovať faktúry, spravovať zákazníkov a transakcie medzi nimi a obsahuje mnoho ďalších užitočných funkcií. S príchodom EET si poradili veľmi dobre. Vyzdvihli by sme možnosť napojenia sa na ich API a tým správne evidovať tržby a vystavovať faktúry z vlastného sytému cez ich systém iDoklad.

Ďalšou pekne spracovanou spužbou na elektronickú evidenciu tržieb, ktorá je zadarmo, je FIKET [2]. Táto webová aplikácia je jednoduché a elegantné riešenie pre EET. Po registrácii a vyplnení informácií o firme (s pridaním certifikátu) umožnuje evidovať nové tržby. Pre každú vytvorenú účtenku vygeneruje online odkaz pre odoslanie zákazníkovi alebo ju vytlačí ako PDF.

Tretie existujúce riešenie pre EET, ktoré tu spomenieme je Smseet [3]. Smseet síce nie je zadarmo, ale konceptuálne zapadá medzi riešenia mojej práce. Táto služba umožnuje evidovať tržby pomocou jednoduchého zasielania SMS. Spätnou SMS vráti odkaz na elektronickú účtenku.

Naše riešenie tejto problematiky sa odlišuje od týchto už existujúcich bezplatných riešení tým, že umožnuje naraz spravovať viacero firiem, ku ktorým sú pridelení zamestnanci s rôznymi právomocami. Ďalej umožnuje pridávať poznámky ako k zadaným tržbám, tak aj k obdobiam.

2.3 Cieľ

Cieľom našej webovej aplikácie je vytvorenie systému pre elektronickú evidenciu tržieb pre maloobchody, stánkový predaj a jednoducho pre všetkých, ktorých sa táto povinnosť týka, ale ich potreby nemajú taký veľký rozsah, aby potrebovali nejaký zložitý spoplatnený systém. Keby sme chceli ešte viac špecifikovať cielovú skupinu, tak sú to podnikatelia, ktorí

[1] https://www.idoklad.cz/
[2] http://www.fiket.cz/
[3] http://smseet.cz/

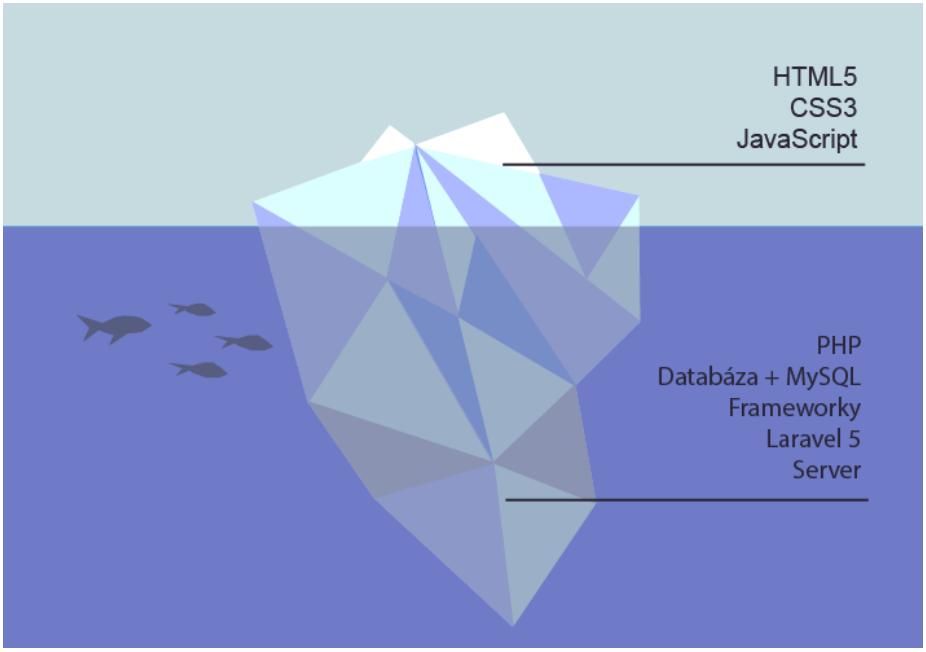
majú viacero malých spoločností a pre každú si potrebujú spravovať evidenciu samostatne. Preto by sme chceli takýmto podnikateľom poskytnúť možnosť elektronického evidovania tržieb pre všetky ich spoločnosti na jednom mieste, avšak odelene. A navyše im poskytnúť možnosť zobraziť prehľad o histórií tržieb.

Kapitola 3

Použité technológie

Súčasti webových stránok by sa dali zaradiť do dvoch skupín, označených ako frontend a backend. Ako je vidieť na obrázku 3.1, kde je stavba webovej aplikácie znázornená ako ľadovec, za **frontend** sa považuje to, čo je viditeľné pre bežného užívateľa. Avšak to je len špička ľadovca webovej stránky. Frontend webových stránok tvoria tri vzájomne prepojené prvky. HTML je značkovací jazyk ktorý nám prezentuje informácie na webe, CSS sa používa na vizuálne formátovanie internetových dokumentov, JavaScript dotvára interakciu na strane používateľa. Na druhej strane, **backend** je administratívna časť stránky, čiže tá, ktorá je neviditeľná pre bežného užívateľa. Túto časť tvoria hlavne tri prvky: databáza na uchovávanie dát webovej aplikácie, webový server a skript na spojenie všetkého dokopy a pridanie funkcionality webovej aplikácie. Pre každú z týchto súčastí existuje viacero možností/nástrojov čo použiť.

V tejto kapitole budú bližšie popísané vyšie spomínané programovacie jazyky a technológie, bude vysvetlená architektúra systému a taktiež tu bude spomenutý verzovací systém Git, ktorý bol používaný popri vývoji tejto práce.



Obr. 3.1: Webová aplikácie z pohľadu frontend vs. backend.

3.1 PHP

PHP je populárny open source skriptovací interpretovaný jazyk pre všeobecné použitie, ktorý je vhodný najmä pre vývoj webových aplikácií. Vďaka jeho obrovskej popularite a tým, že je najrozšírenejším serverovým jazykom, takmer každý webhosting ponúka podporu PHP. Jeho veľkou výhodou je možnosť vloženia php kódu do HTML súboru pomocou jednoduchých začiatočných a ukončovacích značiek <?php a ?> . Kód PHP je spúšťaný na strane servera a výsledok je potom odoslaný klientovi, napríklad vo forme HTML. PHP je dynamicky typovaný jazyk, to znamená, že premenné nie je treba deklarovať, typ premennej sa mení podľa hodnoty, ktorá je v nej uložená. V dnešnej dobre je PHP objektovo orientovaný, čiže je možné používať metódy a postupy obiektovo orientovaného programovania. [11] V praxi sa na zložitejšie webové aplikácie využíva nejaký PHP framework, ktorý prácu značne uľahčuje. Na túto webovú aplikáciu sme použili Laravel framework, o ktorom sa podrobnejšie píše v kapitole 4.

3.2 MVC architektúra

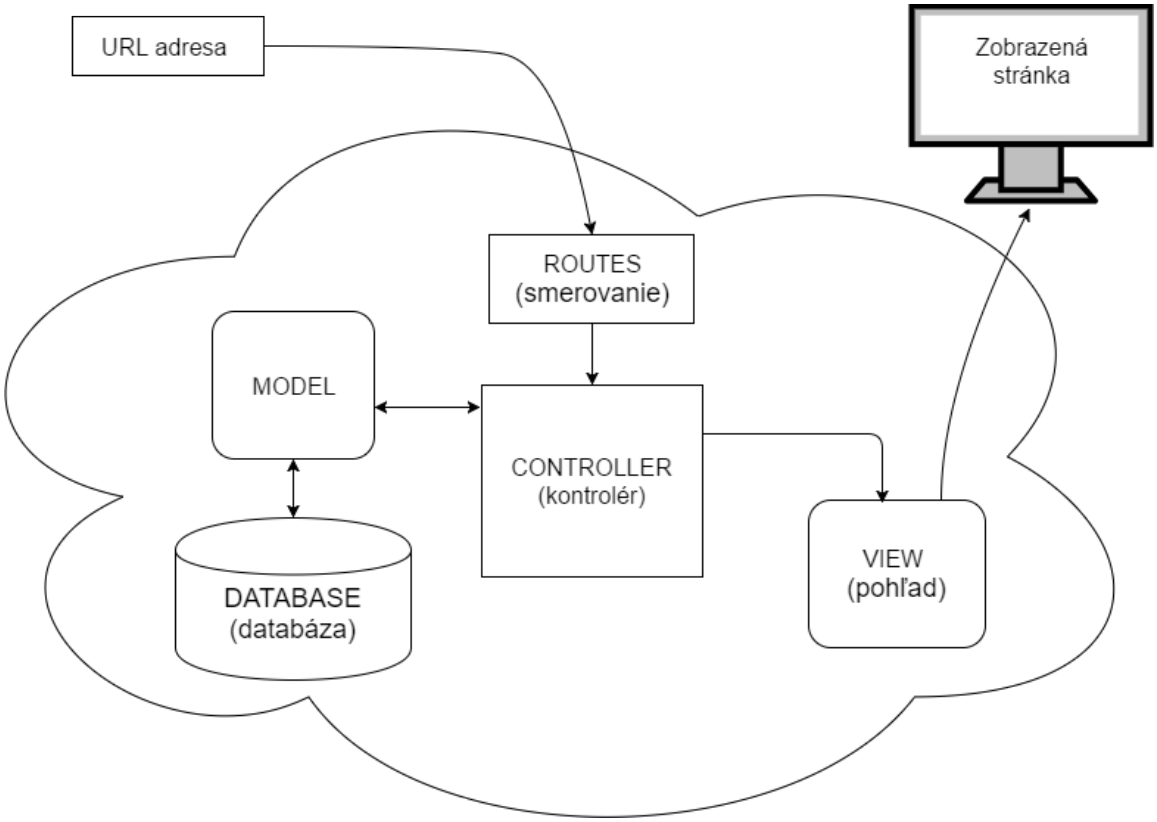
MVC (Model-View-Controller) je vzor softvérového dizajnu postavený na prepojení troch hlavných typov komponentov v programovacom jazyku. Rozdeľuje webovú aplikáciu do troch nezávislých častí: dátový model aplikácie (model), užívateľské rozhranie (view) a riadiacu logiku (controller).

- **Model** — slúži na interakciu s databázou aplikácie a získavanie informácií databázových objektov. Reprezentuje dáta. Funkcia modelu spočíva v prijatí parametrov a vydania dát. Nezaujíma sa o pôvod daných parametrov ani o formát vystupných dát.
- **View** — vykresľuje stránku užívateľovi, šablónu (najčastejšie HTML), do ktorej sa vkladajú premenné. To znamená, všetko čo užívateľ vidí alebo s čím interaguje na stránke.
- **Controller** — spája funkčnosť celého vzoru, spracuje požiadavky od užívateľov, načíta dáta s využitím modelov a zobrazí výsledok ako pohľad (view).

Laravel (a väčšina ďalších PHP frameworkov) ešte z časti rozširuje tento model o “smerovač“ (routes), ktorý mapuje URL adresy na určité akcie kontroleru. Prepojenie medzi komponentami MVC architektúr je znázornené na obrázku 3.2

3.3 MySQL

MySQL poskytuje veľmi rýchly, viacvláknový, viacúčelový a robustný databázový server SQL (Structured Query Language). Je ovládný jazykom SQL, čiže práca je vykonávaná pomocou dotazov. Funguje ako databázový relačný systém, čiže každá databáza je tvorená z tabuliek. Riadky tabuľky reprezentujú jednotlivé záznamy a stĺpce tabuľky reprezentujú dátové typy záznamov. MySQL je rýchly, dobre spolupracuje s PHP a je veľmi populárny, má teda veľkú podporu u webhostingov.



Obr. 3.2: MVC architektúra s dodatkom routovania.

3.4 HTML

HTML (Hypertext Markup Language) je štandardný značkovací jazyk používaný na vytváranie webových stránok. História tohoto jazyka siaha až do osemdesiatych rokov 20. storočia, kedy fungoval len ako jednoduchá podmonžina univerzálneho značkovacieho jazyka SGML, avšak postupom času sa vyvinul do samostatného štandardu. Dnes sa používa najnovší štandard HTML5, ktorý priniesol mnoho skvelých vecí, ako napríklad DOCTYPE, jazyk dokumentu a kódovanie. Pribudlo veľa nových tagov, ktoré pomohli zlepšiť prehľadnosť kódu. HTML sa už používa len na definovanie štruktúry stránky (v minulosti aj na vzhľad).

3.5 CSS

CSS (Cascading Style Sheets) alebo kaskádové štýly sú rozšírením HTML. Postupom času, ako sa zvyšovali nároky na webové stránky, sa v HTML prestal definovať vzhľad a túto funkciu nahradilo presne toto rozšírenie Cascading Style Sheets. CSS určuje ako sa budú jednotlivé elementy (HTML tagy) zobrazovať. Dá sa povedať, že je to súbor pravidiel, určujúcich výzor konkrétnych elementov. Takto sa dá definovať jednotný vzhľad elementu pre celý dokument. To uľahčuje aj urýchluje prácu. Napríklad, keď sa rozhodneme zmeniť farbu nadpisu h1, upravíme jedno pravidlo v CSS a zmenia sa všetky nadpisy h1 na celej stránke (ak to pravidlo nie je nižšie prepísané). Každé také pravidlo sa skladá z dvoch častí,

selektoru (názov elementu, tag, class alebo id) a deklarácie (čo má pre daný element platiť) [10]. Príklad stránky s a bez použitia CSS je vidieť na obrázku 3.3.

Obr. 3.3: Rozdiel s použitím CSS (naľavo) a bez CSS (napravo).

Bootstrap

Bootstrap je najpopulárnejší HTML, CSS a JavaScript framework pre vývoj responzívnych webových stránok. [2] Responzivita stránok znamená, že vzhľad sa prispôsobuje veľkosti okna (obrazovky) zariadenia, na ktorom si užívateľ prezerá webovú stránku. Obsahuje hlavne HTML a CSS návrhové šablóny, pomocou ktorých sa zostavuje stránka, čo dosť uľahčuje prácu, keďže si nemusíme vytvárať CSS pravidlá pre úplne každý komponent nášho webu.

3.6 JavaScript a jQuery

JavaScript je populárny skriptovací programovací jazyk využívaný pri tvorbe webov. Pridáva webovým stránka interakciu na strane užívatela. Taktiež sa pomocou neho vytvárajú animácie a efekty. Vďaka tomu môžu byť stránky krajšie a originálnejšie, ale aj lepšie ovládateľné a užívateľsky prívetivejšie. JavaScript je jeden zo základných stavebných kameňov webových stránok spolu s už spomínaným HTML a CSS. Pre lepšiu prácu s Javascriptom sa používa knižnica jQuery.

JQuery je rýchla a ľahká JavaScript knižnica "write less, do more". To znamená, že malým úsilím chce dosiahnuť veľa vecí. Účelom jQuery je uľahčiť používanie jazyka JavaScript pri vývoji webových stránok. JQuery vezme veľa bežných úloh, ktoré vyžadujú mnoho riadkov kódov JavaScriptu, aby sa vyriešili a tie úlohy obalí do metód. Tie sa potom jednoducho zavolajú a tým skrátia vyriešenie bežnej úlohy na jeden riadok kódu. JQuery tiež zjednodušuje veľa komplikovaných vecí od jazyka JavaScript, ako napríklad volania AJAX[1] a manipuláciu DOM. [5]

[1] Asynchronous JavaScript + XML

Datatables

Ďalšiu výhodou knižnice jQuery je rozmanitosť pluginov pre túto knižnicu. Napríklad DataTables je výkonný plugin pre knižnicu jQuery na vytváranie výpisov dát na stránke v podobe tabuliek. Vylepšuje HTML tabuľky a pridáva do nich interaktívnosť. Poskytuje vyhľadávanie, triedenie a stránkovanie bez akejkoľvek konfigurácie. [9]

Google charts

Google Charts poskytuje skvelý spôsob vizualizácie údajov na webových stránkach. Ponúka mnoho rôznych typov grafov a nástrojov ako zobrazovať určité typy dát. Od jednoduchých lineárny grafov až po zložité stromové mapy. Použitie Google Charts nie je vôbec zložité. Stačí načítať pár javascript knižníc, vybrať si aký typ grafu chceme použiť, vybrať údaje, ktoré sa majú zobrazovať a prispôsobiť si nastavenia grafu. Na stránku sa vloží pomocou s prideleným id, ktoré sa definuje pri grafe. [8]

3.7 Git

Git je bezplatný distribuovaný systém pre správu verzií, používaný pri vývoji softvéru. Správa verzií je systém, ktorý zaznamenáva zmeny súboru, alebo sady súborov v priebehu času a užívateľ tak môže kedykoľvek obnoviť jeho alebo inú konkrétnu verziu. [4] Pre použitie tejto práce sme použili vývojársku platformu GitHub. Je to najrozšírenejšia možnosť pre správu verzií. GitHub vytvára komunitu programátorov, ktorí tu môžu pridávať a zverejňovať kód svojich projektov alebo hodnotiť a komentovať kód druhým, ktorí sa o neho podelili. Skvelá možnosť pre publikovanie open-source projektov.

Kapitola 4

Laravel framework

PHP je momentálne najobľúbenejším skriptovacím jazykom na strane servera. Od začiatku, kedy sa začali objavovať prvé inline fragmenty PHP kódu v statických HTML súboroch, prešlo PHP veľkou expanziou a rozvinutím. Už od počiatkov museli vývojári budovať zložité weby a webové aplikácie a nad istou úrovňou zložitosti zaberalo príliš veľa času a úsilia vždy začínať úplne od začiatku. Odtiaľ vznikla potreba štruktúrovanejšieho prirodzeného spôsobu vývoja. Adekvátne riešenie tejto potreby poskytujú vývojárom PHP frameworky[1]. Pre túto webovú aplikáciu bol použitý Laravel 5.3 framework, ktorý bude bližie popísaný v tejto kapitole.

Laravel je free open-source PHP framework pre tvorbu webových aplikácií, vytvorený Taylorom Otwellom v roku 2011. Odvtedy sa stal jednym s najlepších PHP frameworkom, ak nie celkovo najlepším. Jeho popularita stále rapídne rastie. Vývojári preferujú Laravel kvôli elegancii jednoduchosti používania a prispôsobovania sa ich potrebám. Laravel je postavený na softwarovej architektúre MVC 3.2, ktorá bola popísaná v kapitole vyššie. Snaží sa o odstránenie zložitosti vývoja webov tým, že uľahčuje základné úlohy, ktoré sa používajú takmer na všetkých väčších a zložitejších weboch, ako je napríklad autentifikácia, smerovanie, relácie a ukladanie do vyrovnávacej pamäte. Skvelou výhodou, ktorá nás obzvlášť potešila, je skvelá dokumentácia a množstvo návodov k práci. Takéto video návody (screencasty) nazvali Laracast a je to de facto vzdelávací zdroj špeciálne pre vývojárov vytvárajúci web pomocou PHP a JavaScript hlavne zameraný na Laravel. Sami sa označujú ako "Netflix pre vašu karieru". Ja sám som začínal prácu s Laravelom práve pomocou Laracastu. Obsahuje vysvetlenia od základov až po zložité funkcie tohoto frameworku, jednoduchými názornými príkladmi. Návody sú stále rozširované a aktualizované. Na rozdiel od frameworku Nette, ktorý je od českých vývojárov a má skvelú komunitu v Českej republike, je ovládanie anglického jazyka nutnosťou. Nasledujú jednotlivé vysvetlenia niektorých dôležitých prvkov Laravel frameworku. Väčšina informácií je čerpaných z oficiálnej webovej dokumentácie [7] a z knihy Laravel Design Patterns and Best Practices [13].

4.1 Kľúčové prvky frameworku

Composer

Pri vývoji webových aplikácií často natrafíme na bežné problémy, ktoré sa opakujú vo viacerých projektoch a sú už vyriešené. To je jeden z dôvodov používania frameworkov. Čím sú zložitejšie, tým potrebujú viac knižníc, ktoré riešia určité problémy, čakajúc na ich použitie. Takéto knižnice sa označujú ako závislosti (dependencies). Keďže tieto knižnice môžu mať svoje vlastné závislosti, je jasné, že je v tom potrebný určitý systém. Riešením na to je práve Composer. Composer je nástroj pre správu závislostí v PHP. Povoľuje deklarovať knižnice, na ktorých je projekt závislý a tie potom jednoduchým spôsobom spravuje (inštaluje, aktualizuje atď.). To vytvára možnosť používať funkcie z rôznych frameworkov naraz. Napríklad samotný Laravel v základe obsahuje mnoho skvelo zaužívaných knižníc zo Symphony frameworku. Taktiež veľa vývojárov pridáva svoje open-source riešenia na určité problémy, ktoré sa dajú použiť. Najjednoduchším riešením inštalácie Laravelu je cez composer.

```
composer create-project laravel/laravel web-eet
```

Smerovanie (routing)

Smerovanie (routing) určuje, aká logická operácia sa spustí pri navštívení určitej URL adresy. Laravel prichádza s ľahko použiteľným prístupom k smerovaniu. Routa môže byť spustená v aplikácii s dobrou flexibilitou a kontrolou. Všetky routy sú definované v špecialnych súboroch umiestnených v adresári `routes`. Tieto súbory sú automaticky načítané frameworkom. Routy je možné usporiadavať do skupín s rovnakými vlastnosťami ako napríklad prefix alebo middware (operácia, ktorá sa vykoná pred spustením akcie definnovanej v route). Pri route sa určuje, o aký HTTP dotaz ide (GET, POST, PATCH ...), URL s ktorou sa má zhodovať, názov routy, ktorý je nepovinný, ale veľmi uľahčuje používanie routy jednoduchým odkazovaním sa naň a akcia, aká sa má vykonať, najčastešie konkrétna metóda z kontroléru.

Blade

Zobrazovanie výstupu pre užívateľa rieši, Laravel perfektne. Blade je jednoduchý ale účinný engine pre tvorbu šablón. Na rozdiel on iných PHP enginov na tvorbu šablón, blade v ňom neobmedzuje používanie PHP kódu. V skutočnosti sú všetky blady kompilované do čistého PHP. Tento engine na tvorbu návrhov poskytuje množstvo príkazov na jednoduchú integráciu PHP kódu do HTML. Najviac používané obyčajné zobrazenie premennej (príkaz `echo`), stačí obaliť premennú do štyroch zložených zátvoriek, napríklad `{{ $name }}`. Alebo pre kontrolné štruktúry, slučky a mnoho dalších určiť začiatok a koniec danej časti pomocou "@", napríklad:

```
@if (count($people) > 0)
    <td>{{ $people[$id]["name"] }}</td>
@else
    <td> - </td>
@endif
```

Pre neopakovanie sa častí kódu HTML v rôznych blade súboroch je možné vytvárať šablóny a sekcie. Napríklad, každá stránka potrebuje mať jednotnú hlavičku a pätu. Vytvoríme

master šablónu, do ktorej vložíme pomocou príkazu `@yield('content')` miesto, kde bude časť stránky, ktorá sa obmieňa. Na jednotlivých stránkach už len definujeme, čo má brať ako jeho šablónu pomocou `@extends('master')` a jej obsah obalíme do `@section('content')` a `@endsection`. Takýchto sekcií (yields) samozrejme šablona môže obsahovať viac, napríklad pre nadpis, CSS a JavaScript alebo slider.

Eloquent ORM

Eloquent ORM poskytuje jednoduchú implementáciu pre prácu s databázou a to vďaka tomu, že pre každú tabuľku z databázy má zodpovedajúci "model", ktorý sa používa na interakciu s touto tabuľkou. Modely umožňujú vyhľadávať údaje vo vašich tabuľkách, ako aj vkladať do tabuľky nové záznamy.

Migrations a Seeding

Migrations pomáhajú udržiavať aktuálnu schému databázy pri práci viacerých ľudí na jednom projekte. Dá sa povedať, že Migration je verzovanie (version control) pre databázu, ktoré umožňuje jednoducho upravovať a zdielať schému databázy aplikácie v tíme. Migrácie sú typicky spárované so stavebnou schémou Laravelu, vďaka čomu je veľmi ľahké vytvárať databázovú schému aplikácie.

Seeding sa používa na naplnenie údajov do databázy pre testovanie alebo na začiatočné nastavenie aplikácie. Ide o automatizovaný proces. Takto môžu byť databázové tabuľky vložené s predvolenými údajmi. Je to vhodné napríklad na vloženie dát, o ktorých sme si istí, že ich budeme potrebovať, napríklad dáta pre užívateľa SuperAdmin.

Artisan

Artisan je rozhranie príkazového riadku, ktoré je súčasťou Laravelu. Poskytuje celý rad užitočných príkazov, ktoré pomáhajú pri zostavení aplikácie. Najčastejšie používané príkazy sú na generovanie tried s možnosťou preddefinovania obvykle používaných funkcií pre danú triedu. Je možné si vytvárať aj vlastné artisan konzolové príkazy.

Autentifikácia

Laravel umožnuje veľmi jednoduchú implementáciu autentifikácie. V skutočnosti je takmer všetko už pripravené a nakonfigurované. Konfiguračný súbor na autentifikáciu je umiestnený v súbore `config/auth.php`, ktorý obsahuje niekoľko dobre zdokumentovaných možností na úpravu správania autentifikačných služieb. V mnohých prípadoch nebude nikdy potrebné upravovať predvolenú konfiguráciu. V jadre sa táto autentifikácia skladá zo stráži (guards) a poskytovateľov (providers). Stráže definujú, ako budú jednotliví užívatelia autentizovaní pre každú požiadavku na systém. Poskytovatelia definujú, ako sú užívatelia načítaní z databázy. Oba tieto prvky je možné upravovať a vytvárať.

Middleware

Ako naznačuje názov, Middleware pôsobí ako prostredník medzi požiadavkou a odpoveďou. Je to typ filtračného mechanizmu. Napríklad Laravel obsahuje middleware, ktorý overuje, či používateľ aplikácie je autentizovaný alebo nie. Ak je používateľ overený, bude presmerovaný na domovskú stránku, inak bude presmerovaný na prihlasovaciu stránku. Pre vytvorenie novej middleware stačí príkaz `php artisan make:middleware <middleware-meno>`. Takto novo vytvorený middleware bude vytvorenu v adresári `app/Http/Middleware` [6].

Bezpečnosť

Pri vývoji aplikácie musí každý vývojár myslieť na to, ako zabezpečiť aplikáciu. Laravel sa stará o niektoré bezpečnostné prvk vo svojom frameworku. Používa hashované heslo, čo znamená, že heslo nebude v databáze nikdy uložené ako obyčajný text. Využíva algoritmus Bcrypt hashing na generovanie šifrovaného zobrazenia hesla. Laravel používa pripravené príkazy SQL, ktoré robia injekčné útoky nepredstaviteľné. Spolu s tým poskytuje Laravel jednoduchý spôsob ako predísť tomu, aby užívateľ nejakým vstupom vložil značku `<script>` do kódu stránky.

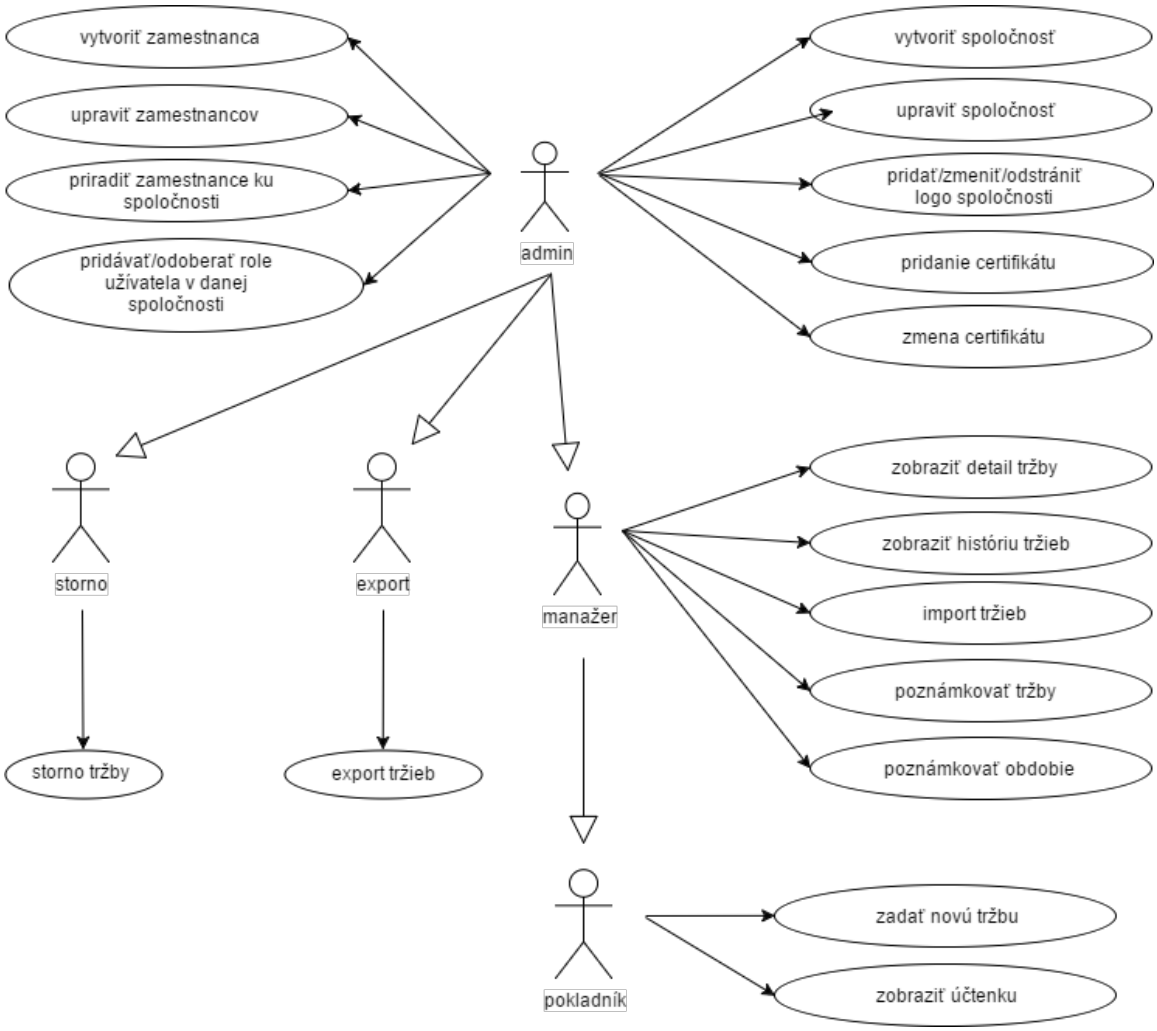
Kapitola 5

Návrh informačného systému

Teraz keď vieme, čo je našou úlohou a aké technologické prostriedky na riešenie tejto úlohy budeme využívať, môžeme sa pustiť do návrhu implementácie systému pre túto bakalársku prácu. Práve v tejto kapitole sa budeme venovať tejto analýze návrhu webovej aplikácie. Základom funkčnosti tejto webovej aplikácie je, aby v nej bolo možné vytvárať užívateľov priradených k firmám s rôznymi právami pre vykonávanie rôznych úloh ako zadávanie novej tržby, storno, prezeranie histórie, porovnávanie a poznámkovanie, import a export tržieb. To všetko je treba premyslieť a navrhnúť. Pomocou diagramu prípadu použitia (use case diagram) priblížime, ako sa užívateľ bude môcť správať v tomto systéme. Ďalej využijeme ER (entity-relationship) diagram na rozobtratie návrhu pre databázu a organizovanie dát. V neposlednom rade sa budem venovať návrhu užívatelského rozhrania. Pre metodiku návrhu tohoto informačného systému sme čiastočne čerpali informácie zo študijnej opory z predmetu Informační systémy. [12]

5.1 Diagram prípadu použitia (use case diagram)

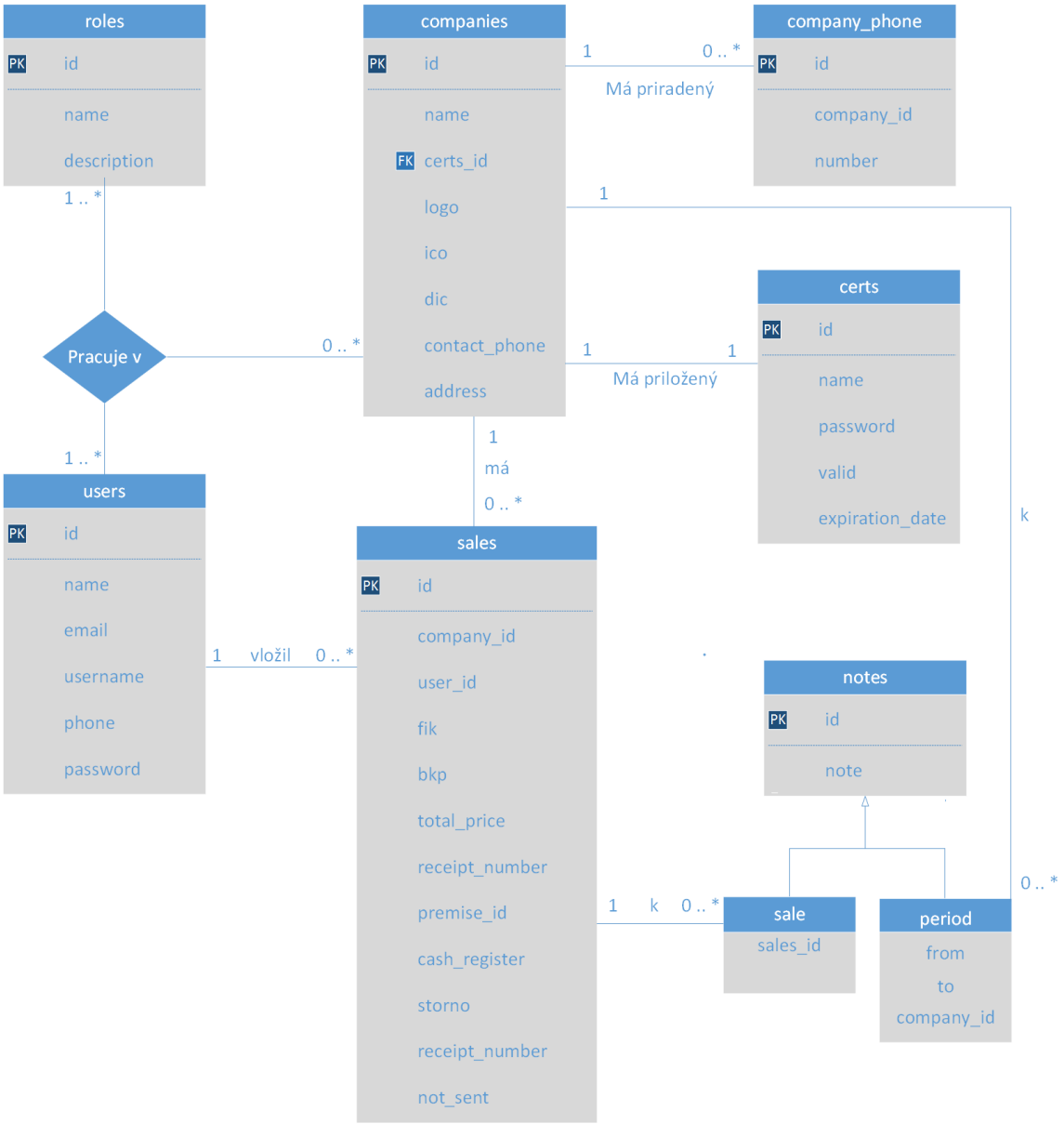
Väčšina informačných systémov pre ich správne fungovanie musí rozdeliť užívateľov do viacerých skupín podľa ich právomocí, ktoré v systéme majú. Po analýze úlohy sme sa rozhodli pre vytvorenie piatich rolí. Admin, ktorý môže využívať v podstate všetky možnosti sytému, manažér, ktorého funkcia má byť na prezeranie histórie tržieb a analyzovanie ich, pokladník, ktorého úlohou je evidovať nové tržby a posledné dve, storno tržieb a export tržieb. V našom prípade má však rola význam akéhosi povolenia na vykonávanie určitých príkazov, keďže užívateľ môže mať priradených viacero rolí. Tento diagram prípadu použitia 5.1 popisuje práve tieto roly. Diagram je jemne zjednodušený pre lepšiu prehľadnosť. Taktiež treba brať do úvahy, že užívateľ môže mať priradených viac firiem a v každej mať rozdielne roly. Rola administrátora webu preberá oprávnenia všetkých rolí. Takéto rozdelenie bolo zvolené hlavne preto, aby možnosť stornovať a exportovať tržby bola volitelná pre každého užívateľa.



Obr. 5.1: Use case diagram

5.2 ER diagram

Jednou zo základných častí informačných systémov je databáza na uchovávanie údajov. Navrhnutie dobrej databázy dokáže veľmi uľahčiť prácu pri implementácii. Jeden zo spôsobov je vytvorenie Entity-relationship diagramu (ERD). Vďaka nemu je vidieť, aké entity budeme využívať a aké sú vzťahy medzi nimi. Reprezentuje len abstraktný pohľad na databázu, v skutočnosti je databáza o niečo zložitejšia. Najdôležitejšou vecou tohto návrhu je uvedomenie si vzťahu užívateľ-rola-spoločnosť (user-role-company). Užívateľ môže byť vo viacerých spoločnostia majúc rozdielne roly. Ďalšou vecou, ktorú je vidieť v návrhu je, že poznámka (note) je naviazaná buď na samotnú tržbu (sale) alebo na spoločnosť, v prípade, že sa jedná o poznámku pre obdobie.

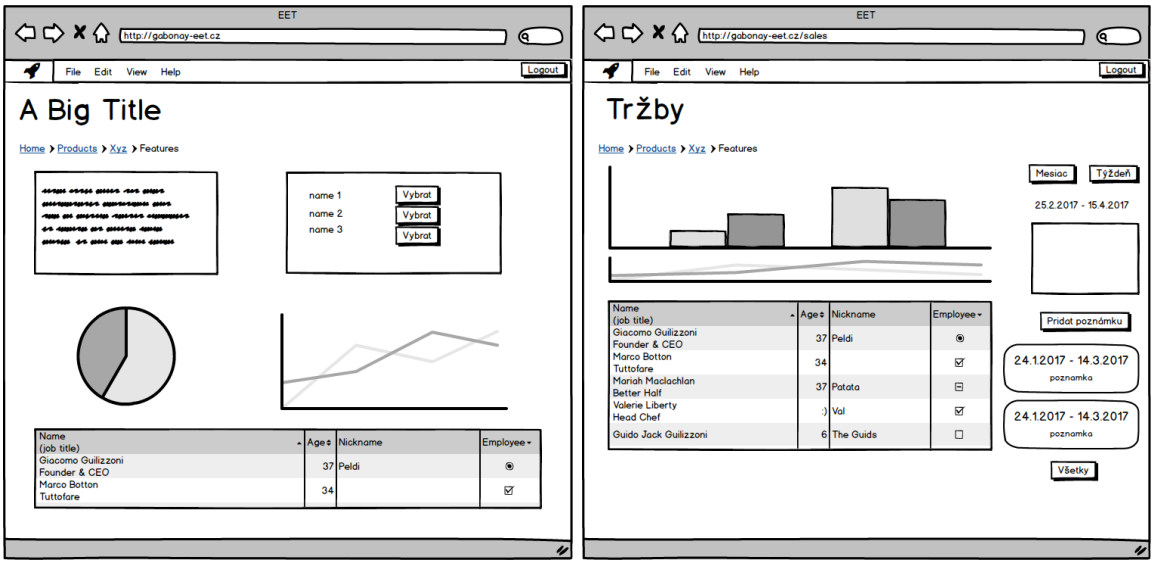


Obr. 5.2: ER diagram.

5.3 Návrh užívateľského rozhrania

Vždy je dobré mať vopred predstavu o tom, ako by mala vytváraná aplikácia vyzerať. Na to dobre poslúži vytvorený mockup. Na tvorbu návrhou GUI [1] sme využili službu Balsamiq Mockups [2], v ktorej sa dajú vytvárať jednoduché ale aj zložitešie mockupy veľmi rýchlo.

Pre návrh GUI tejto webovej aplikácie hrajú rolu 2 základné veci a to zobrazovanie dát a zadávanie novej tržby. Na obrázku 5.3 je znázornený dashboard a hlavné zobrazovanie tržieb pre manažéra. Dashboard je vo väčšine prípadov prvá stránka, ktorá je zobrazená po prihlásení. Podstatné pre ňu je výber zo spoločností, ktoré má užívateľ priradené a nejaké základné údaje o vývoji tržieb vo vybranej spoločnosti. Pre hlavný prehľad tržieb sme zvolili kombináciu zobrazenia dát pomocou grafu a tabuľky. Pridávanie a zobrazenie poznámok je v pravej časti obrazovky.

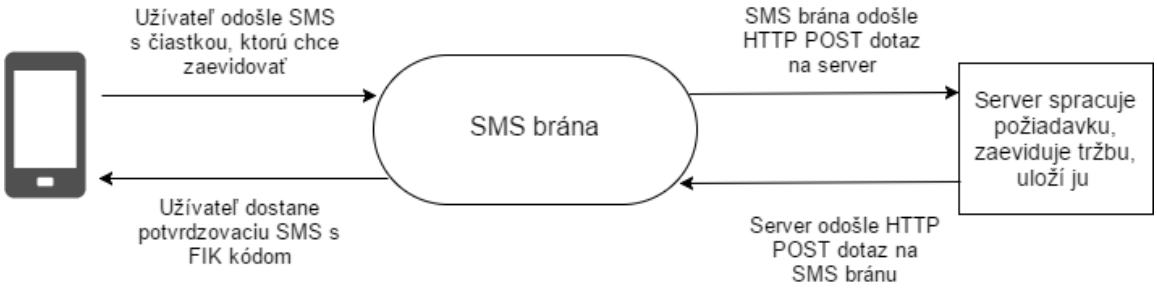


Obr. 5.3: Mockup pre dashboard (naľavo) a tržby (napravo) .

5.4 Návrh napojenia na externý systém

Pre napojenie na nejaký externý systém máme viacero možností, ako napríklad napojenie na cloud print pre tlač účteniek alebo napojenie na napojenie pre obecnú SMS bránu. Nakoniec sme zvolili tú druhú spomenutú možnosť, napojenie na SMS bránu. K tomu bude potrebné navrnúť jednoduché API (Application programming interface), ktoré bude na serverovej časti. Klient odošle SMS na jedno z čísel nastavených v systéme, API príjme dotaz z SMS brány s potrebnými dátami, vyvolá určité akcie ohľadom evidovania danej tržby a odošle naspäť dotaz na SMS bránu a tým doručí potvrdzovaciu SMS užívateľovi. Ide naozaj o jednoduché API s jednou špecifickou metódou pre prijatie tržby. Princíp je znázornený na obrázku 5.4.

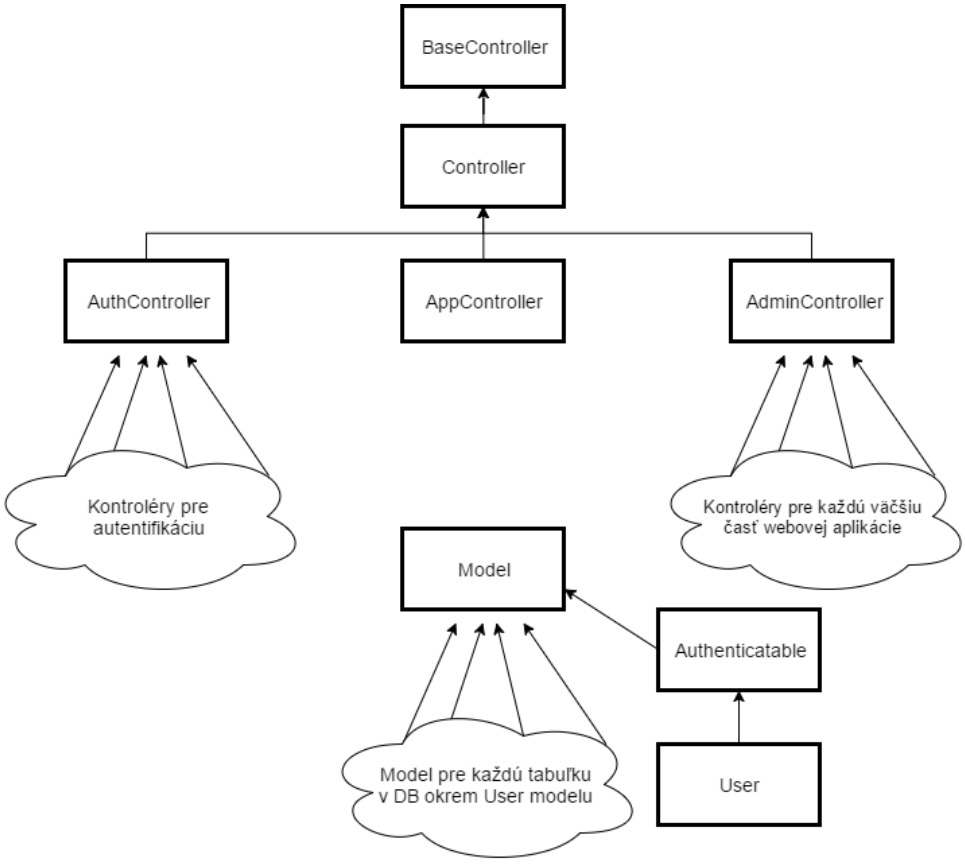
[1] Graphical User Interface
[2] https://www.mybalsamiq.com



Obr. 5.4: Napojenie na externý systém (SMS bránu).

5.5 Diagram tried

Dve najhlavnejšie skupiny PHP tried, ktoré budeme vytvárať, sú pre modely a pre kontroléry. Dedičnosť týchto tried je v zjednodušenej podobe znázornená na obrázku 5.5.



Obr. 5.5: Diagram tried pre kontroléry a modely.

5.6 Harmonogram implementácie

Ešte predtým, ako rozoberieme jednotlivé časti implementácie v nasledujúcej kapitole 6, popíšeme stručný harmonogram implementácie, ktorý sme si rozdelili do niekoľkých fáz.

- **1. Prípravná** - v tejto fáze vytvoríme všetko potrebné pre nadchádzajúce fázy, čo nám uľahčí ďalšiu prácu. Ako prvé si vytvoríme kostru databázy, ktorú sme si navrhli. Na to využijeme migrations z Laravelu, vďaka čomu nebudeme mať problém s neskoršími úpravami a zásahmi do databázy. Ďalším krokom bude vytvorenie modelov so základnými funkciami pre každú tabuľku z databázy. Následne vytvoríme vizuálnu kostru, na ktorú budeme ďalej nabaľovať. Tým sa rozumie vytvorenie dvoch šablónových súborov `empty.blade.php, master.blade.php` a jeden s vlastnými kaskádovými štýlmi `style.css`. Ďalej nahratie všetkých html, css a javascript frameworkov, potrebných pre túto webovú aplikáciu. V tomto kroku si ešte pripravíme potrebné kontroléry, do ktorých v neskorších fázach budem už len pridávať ďalšie funkcie. Posledné, čo v tejto fáze urobíme, je pripravenie si súboru `routes/web.php` pre mapovanie URL adries.

- **2. Užívateľská** - spracovanie užívateľov a spoločností. To bude obnášať, autentizáciu a registráciu. Ďalej vytváranie nových užívateľov a nových spoločností, k tomu patrí aj prideľovanie užívateľov k spoločnostiam. Ďalšou dôležitou časťou tejto fýzy je vytvorenie rolí a práv a ich viacnásobné prideľovanie užívateľom vo vzťahu k spoločnostiam.

- **3. Systémová** - teraz bude potreba pridať všetky funkcie webovej aplikácie a to zadávanie nových tržieb a následné vrátenie FIK kódu, možnosť zobraziť jednoduchý doklad o platbe, históriu tržieb, výber obdobia, poznámkovanie obdobia alebo samotnej tržby, import a export.

- **4. Finálna** - táto fáza poslúži ako dokončenie celkovej aplikácie. Zahŕňa to rozhranie pre napojenie na externý systém (SMS bránu), finálne úpravy webovej aplikácie, testovanie a nakoniec oprava chýb a debugovanie.

Kapitola 6

Implementácia

Najdôležitejšou a najrozsiahlejšou fázou vývoja webovej aplikácie je jej samotná implementácia. Tej sa bude venovať táto kapitola, ktorá je rozdelená do niekoľkých častí, ako prácu s databázou, správu užívateľov a spoločností, evidovanie tržieb, napojenie na externý systém, zobrazovanie dát, import a export a nakoniec testovanie. Popíšeme základný princíp implementácie v týchto jednotlivých častiach a v každej riešenie konkrétneho problému. Tieto časti čiastočne nadväzujú na predošlú kapitolu o návrhu 5.

Ako vývojové prostredie sme zvolili PhpStorm od firmy JetBrains. Webovú aplikáciu sme vytvárali na lokálnom serveri za pomoci apachu XAMPP.

6.1 Modely a práca s databázou

Práca s databázou je dôležitou súčasťou našej webovej aplikácie, či už na vyberanie údajov z databázy alebo na vkladanie do databázy. Prvé, čo je treba definovať, je konkrétne pripojenie k vybranej databáze. Pripojenie sa dá definovať priamo v súbore `config/database.php`, avšak praktickejšia možnosť je využiť takzvané enviroment configuration (konfigurácia prostredia), ktoré sa nastavuje v súbore `.env`. Pri zmene databázového servera stačí upraviť pár riadkov v tomto `.env` súbore. Výhodou je prehľadnosť a možnosť rýchlej modifikácie.

Ďalším krokom bolo vytvorenie štruktúry databázy, na čo sme využili migrations z Laravelu. Pre každú vytvorenú tabuľku sme následne vytvorili model. V každom tomto modeli sme pridali premennú typu `protected` pre názov tabuľky. Týmto sú modely o trochu prehľadnejšie a o trochu univerzálnejšie. V modeli pre spoločnosť používame rozšírenie SoftDelete. To znamená, že pri volaní funkcie na odstránenie záznamu, sa v skutočnosti neodstráni z databázy, len sa u neho nastaví, že je odstránený. To má dve výhody: vyvarujeme sa možnej fatálnej chybe v prípade volania záznamu naviazaného na spoločnosť, čo by sa však nemalo stať, a druhou výhodou je, že v prípade, keď si užívateľ rozmyslí zmazanie danej spoločnosti, je možné to navrátiť do pôvodného stavu ako pred odstránením.

Veľmi dôležitou informáciou pre systém je aktuálne zvolená spoločnosť. To sa odzrkadľuje na tom, že takmer každý dotaz pre výber z databázy obsahuje ID spoločnosti. Najzložitejšou časťou ohľadom databázy je vytvorenie vzťahu užívateľ, rola a spoločnosť. Pre toto spojenie boli vytvorené dve pomocné tabuľky v databáze, `user_company` a `user_company_role`. Na túto problematiku nadviažeme a bližšie to rozoberieme v nasledujúcej kapitole 6.2.

6.2 Správa užívateľov v spoločnosti

Jednou z najpodstatnejších vecí, aby systém pracoval správne, je možnosť zaregistrovanému užívateľovi vytvárať viacero spoločností, do nich prideľovanie zamestnancov (ďalších užívateľov) a aby každý tento zamestnanec mal v danej spoločnosti určité práva.

To máme spracované nasledovne. Po zaregistrovaní a vytvorení spoločnosti sa daný užívateľ automaticky stáva administrátorom (priradená rola admin) v tejto spoločnosti. Spravovať ju a jej zamestnancov môže výlučne len admin v danej spoločnosti. Máme ošetrené, že každá spoločnosť má priradeného vždy minimálne jednoho admina, to znamená, že ak je len jeden admin v spoločnosti, nie je možné mu odobrať toto právo.

Keďže systém je stavaný pre prípad, že užívateľ je priradený vo viacerých spoločnostiach, je treba mať určené, aká spoločnosť je aktuálne zvolená. Po prihlásení je zvolená prvá užívateľova spoločnosť a další výber je možný na úvodnej strane (dashboarde). Pri každom zvolení spoločnosti, či už pri prihlásení alebo pri výbere, sa volá funkcia `changeSelectedCompany`, v ktorej sa ID aktuálnej zvolenej spoločnosti uloží do `HTTP session` a zistia a nastavia sa užívateľove oprávnenia taktiež do `session`. To znamená, že v systéme je vždy primárne nastavených šesť session premenných, jedna s ID aktuálnej spoločnosti a päť pre každú užívateľovu rolu v aktuálnej spoločnosti s hodnotou `true` alebo `false`. Ukladanie si týchto stavov nie je jedinou možnosťou ako tento problém riešiť a sme si vedomí, že pravdepodobne toto riešenie nie je najlepším, ale túto metódu sme zvolili hlavne pre jej jednoduchosť a nekonfliktnosť.

Administrátor má možnosť meniť užívateľom oprávnenia dvoma spôsobmi. Buď v detaile spoločnosti, kde má prístup ku všetkým užívateľom v danej firme, alebo v detaile užívateľa, kde má prístup ku všetkým spoločnostiam, v ktorých je adminom. To je znázornené na obrázku 6.1. V oboch prípadoch sa po kliknutí na ikonu k danej role zavolá funkcia `switchRoleUC`. Podstata tejto funkcie je, že buď odstráni záznam o role užívateľa v spoločnosti z tabuľky `user_company_role` alebo ho tam vloží. Pri odobratí užívateľa zo spoločnosti a následne jeho opätovné pridanie naspäť zachová jeho oprávnenia. To je jeden z hlavných dôvodov, prečo tento vzťah bol rozdelený do dvoch tabuliek v databáze a nie do jednej, ako sa spomínalo vyššie 6.1.

Obr. 6.1: Správa oprávnení, v detaile spoločností (naľavo) a v detaile užívateľa (napravo)

Ak sa prihlási užívateľ, ktorý nemá priradenú žiadnu spoločnosť alebo v priradenej spoločnosti nemá žiadne oprávnenia, tak ho systém upozorní a dá mu možnosť vytvoriť novú spoločnosť. V prípade, ak sa prihlási užívateľ, ktorý má priradenú len jednu spoločnosť a v tej má len oprávnenie pre pokladníka, je automaticky presmerovaný na stránku zadávania novej tržby. Tieto špeciálne prípady boli takto riešené, aby bol systém o trochu viac užívateľsky prívetivejší.

Pre riešenie oprávnení sme postupovali podľa návrhu, avšak po implementácii systému sme si uvedomili, že sme nezvolili najvhodnejšiu metódu pre tento konkrétny systém. Náš systém aktuálne podporuje päť oprávnení, ktoré sa prideľujú užívateľovi v rôznych kombináciách, čo sa ukázalo byť zbytočné, keďže oprávnenie admin, manažér a pokladník sú hierarchické (admin rozširuje oprávnenie manažéra a manažér rozširuje oprávnenie pokladníka). Lepším riešením by bolo, keby existovali len tri roly (admin, manažér a pokladník) a užívateľ by mal priradenú len jednu z nich. Možnosť exportu a stornovania tržieb by bola len ako zaškrtávacia možnosť pri nastavení užívatela (flag v databáze). Výhodou nášho riešenia je, že by nebolo až tak problematické vytvorenie nejakých nových oprávnení a ich následné integrovanie do tohto systému.

6.3 Elektronické evidovanie tržieb

Dostávame sa k dôvodu, prečo vlastne systém elektronické evidovanie tržieb vznikol. Obecný princíp fungovania EET bol vysvetlený v sekcii 2.1. Túto časť sme sa snažili generalizovať a čo najviac zjednodušiť, tým pádom by nemal byť problém ju v budúcnosti rozširovať a modifikovať pre konkrétneho záujemcu o prevzatie tejto webovej aplikácie.

Certifikácia

Aby bolo možné evidovať tržby, spoločnosť musí mať priradený platný certifikát. V prípade, že tak nie je učinené, systém je blokovaný a užívateľ upozornený, nech sa obráti na administrátora v danej spoločnosti. Certifikát sa vkladá z detailu spoločnosti a len administrátor má právo ho vkladať. Užívateľ vkladá certifikát vo formáte pkcs12 [1], ako ho dostal z webovej aplikácie EET a heslo k nemu, ktoré taktiež obdržal. Následne sa zavolá funkcia `addCert`. V tejto funkcii sa načíta obsah nahraného certifikátu a následne sa tento obsah rozdelí pomocou funkcie `openssl_pkcs12_read` na jednotlivé kryptografické objekty. V tomto prípade na súkromný kľúč (pkey) a verejný certifikát (cert). Verejný certifikát sa uloží pomocou funkcie `openssl_x509_export`. Tieto dva vyexportované súbory (private.key a public.pub) sa uložia do priečinku s názvom podľa ID spoločnosti. Vďaka tomu by sa nemali premiešať certifikáty medzi spoločnosťami. Do databázy sa uloží prepojenie spoločnosti s certifikátom, heslo k nemu, dátum s expiráciou certifikátu a či je vložený certifikát platný. To, či je platný, otestuje funkcia `testOfCert`, v ktorej sa systém pokúša odoslať tržbu v overovacom móde s pridaným certifikátom ku spoločnosti.

Pridávanie a evidovanie tržieb

Nové tržby sa vkladajú pomocou formulára, ktorý je rozdelený do dvoch častí. V prvej časti sa pridávajú jednotlivé produkty, ktoré majú byť súčasťou tržby a ich cena, pomocu jQuery. Toto je však len pre užívateľa, aby mal prehľad o tom, čo vlastne patrí do tržby. Až pri potvrdení druhej časti sa eviduje tá tržba, pre ktorú stačí len údaj o celkovej cene tržby.

Funkcia `eetSend`, ktorá sa stará o zaevidovanie tržby na Finančnú správu vezme všetky potrebnú údaje a túto tržbu odošle s príslušnou certifikáciou. Pre samotné evidovanie tržby využívame open-source klientskú knižnicu `slevomat/eet-client` viz. [2]. Aby sa mohla odosielať šifrovaná certifikovaná správa na Finančnú správu, server, kde je aplikácia umiestnená, musí mať povolené okrem iného PHP SSL rozšírenie. Údaje o tržbe sa následne uložia do

[1] súborový formát pre archivovanie viacerých kryptografických objektov v jednom súbore
[2] https://github.com/slevomat/eet-client

databázy. Záznam sa ukladá aj v prípade, ak evidovanie tržby neprebehlo úspešne. Pôvodne sme mysleli, že ju budeme ukladať len v prípade, ak chyba bola na Finančnej správe, čiže napríklad dočastne vypnuté servery, ale nakoniec sme sa rozhodoli, ukladať všetky neúspešne zaevidované tržby, aby bolo jasné, že niečo nefunguje v systéme správne, napríklad chyba certifikátu. Tieto tržby sú viditeľne odlíšené a na viacerých miestach v systéme je varovná hláška o počte nezaevidovaných tržieb. Taktiež je možné sa pozrieť, koľko hodín ubehlo od jej pridania. Systém má možnosť sa jednim kliknutím pokúsiť zaevidovať všetky tieto tržby naraz. Vďaka tomú sa odkrýva možnosť evidovania tržieb offline bez prístupu internetu. Napríklad je možné celý deň na lokálnom serveri na počítači, kde nie je prístup k internetu neúspešne evidovať tržby a večer na mieste s prístupom k internetu všetky tržby naraz správne zaevidovať. Samozrejmosťou je možnosť tržby aj stornovať. Pre stornovanie tržby sa jednoducho odošle správa ako pri evidovaní, ale so zápornou hodnotou celkovej tržby, ktorú chceme stornovať. Pri stornovaní nesprávne zaevidovanej tržby sa jednoducho nastaví príznak v databáze. Účtenka pre tržbu je generovaná len ako HTML, kde sú zahrnuté informácie o spoločnosti aj s čiernobielym logom, ak má spoločnosť nejaké logo pridané, a informáciami o tržbe, z ktorých sú hlavné FIK a BKP.

Riešenie, ktoré sme zvolili, by sa dalo rozšíriť rôznymi smermi. Sme si taktiež vedomí, že má svoje nedostatky, avšak až príliš neskoro sme si uvedomili, akým smerom by sme chcel realizovať túto časť, preto je len v základnej podobe. Možnostiam, akým by sme sa snažili rozšíriť a modifikovať tento systém, sa budeme venovať v sekcii 7.1.

6.4 Napojenie na externý systém (evidovanie pomocou SMS)

Jednou z viacerých funkcií tejto práce je napojenie na externý systém. Nakoniec sme sa rozhodli pre napojenie systému na obecnú SMS bránu. Keďže sa nám nepodarilo nájsť vhodnú webovú službu pre SMS bránu, ktorá by bola zadarmo, vytvorili sme riešenie pre fiktívnu SMS bránu. Toto riešenie by sa dalo s malým úsilím pretvoriť pre nejakú konkrétnu SMS bránu.

Funkcia `receiveSms` popisuje správanie systému po obdržaní HTTP POST dotazu od SMS brány, napríklad, keď SMS brána vyvolá:

```
curl --data "tel-number=730998602&msg-text=150,50"
http://gabonay-eet.cz/receive-sms
```

Funkcia podľa telefónneho čísla zistí, pod ktorou spoločnosťou má tržbu zaevidovať. Spoločnosť môže mať priradených viacero telefónnych čísel, z ktorých je možné SMS posielať. Následne systém uskutoční nejaké kontroly a zaevidovanie tržby. Pomocou `curl` je odoslaný HTTP POST dotaz na server SMS brány so spätnou správou obsahujúcou FIK kód alebo informáciu, že sa platbu nepodarilo správne zaevidovať.

6.5 Zobrazovanie dát

Pre zobrazovanie dát v informačnom systéme je dôležité, aby boli zrozumiteľné a prehľadné. Je treba vyvážiť to, aby sa zobrazovalo čo najviac dôležitých informácií a to, aby stránka nebola zahltená infomáciami, čo by ju zneprehľadňovalo. Na zobrazovanie dát používame tabuľky implementované pomocou DataTables a grafy implementované pomocou Google Charts.

Tabuľky pomocou DataTables nie je obtiažne implementovať. S kombináciou `blade` šablon od Laravelu, do tabuliek naskladáme všetky informácie tak, ako je treba. Kontrolér

odovzdá informácie modelu, ktorý vyberie dáta z databázy pomocou jednoduchších, ale aj zložiteších dotazov, napríklad kombináciou až z piatich tabuliek naraz. Tieto dáta sú následne zformátované späť v kontroléry a odovzdané do `blade` šablóny. Aby bolo možné triediť tabuľky podľa dátumu, využívame rozšírenie `date-de`. Aby bol prechod medzi výpisom spoločností alebo výpisom užívateľov do ich detailov praktickejší, vytvorili sme klikateľné riadky tabuľky pomocou jQuery.

Pre zobrazovanie štatistík o tržbách spoločnosti sme zistili, že ich nestačí zobrazovať len pomocou tabuliek, ale je treba ich rozšíriť aj o grafy. To je však rozlíšené, či je prihlásený manažér alebo pokladník. Pre pokladníka sa zobrazujú len jeho posledné pridané tržby, aby mal prehľad, čo vlastne zadával do systému. Nemusí mať plné štatistiky o priebehu pomocou grafu.

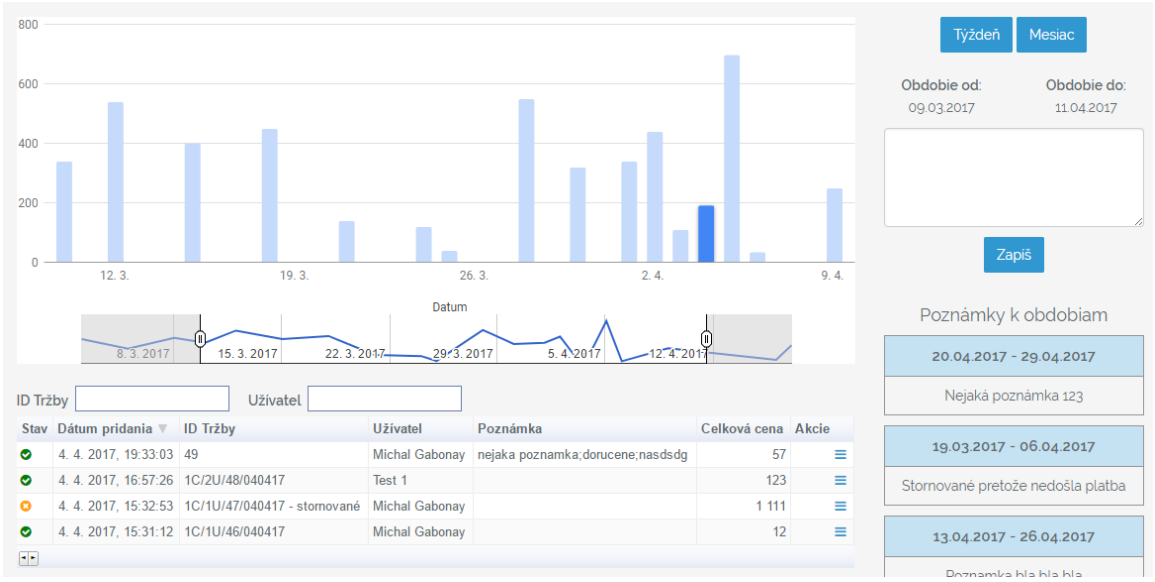
Pre manažérske zobrazovanie histórie tržieb využívame kombináciu stĺpcového grafu a tabuľky. Graf zobrazuje tržby združené podľa dní a tabuľka zobrazuje jednotlivé tržby. To znamená, že pracujeme s dvoma dátovými zdrojmi pre jednu sústavu grafov, označených v Google Charts ako `dashboard`. To zo začiatku tvorilo určité problémy s filtráciou. Pre filtráciu podľa dátumu, alebo lepšie povedané výber obdobia, využívame posúvny slider (`ChartRangeFilter`) pod grafom. Tento slider ovplyvňuje ako graf, tak aj tabuľku. Taktiež je možné pomocou tlačidiel vybrať týždeň dozadu, čiže posledných sedem dní a mesiac dozadu, čiže posledných 30 alebo 31 dní dozadu. Podstatou týchto výberov obdobia je definovanie rozsahu dátumu od-do. K takto vybraným obdobiam je možné vytvárať poznámky v pravej časti stránky, ktoré vezmú rozsah vybraných dátumov od-do a priradia k tomu textovú poznámku. Ako sme už spomínali, v grafe sú tržby združené podľa dní. Po kliknutí na určitý stĺpec pre deň v grafe, tabuľka zobrazí všetky jednotlivé tržby v tom danom dni. Tabuľku je možné ešte filtrovať podľa ID tržby a užívateľa, ktorý zaevidoval túto tržbu. To sme docielili pomocou `StringFilter`.

Pre vytvorenie toho všetkého sme museli správne naimplementovať niekoľko spolu súvisiacich častí. Ako prvé nadefinovať dva datové zdroje. Ďalej správne podľa požiadaviek nastaviť každý prvok, čiže stĺpcový graf, tabuľku a všetky filtrácie. Nasledovalo určiť, aký prvok pracuje s akým dátovým zdrojom (`setDataTable`), ako majú byť tieto prvky navzájom prepojené (`bind`) a vykreslenie každého prvku do určených HTML entít podľa atribútu `id`. Ostáva už len nastaviť správanie jednotlivých filtrácií. To, kedy sa majú spúšťať pomocou takzvaných poslucháčov (Listeners) a taktiež čo a ako sa ovplyvní ich zmenou. Na obrázku 6.2 sú znázornené jednotlivé prvky zobrazovania histórie tržieb.

Najzložitejšou časťou našej práce bola implementácia týchto grafov do webovej aplikácie. Pracovali sme s technológiou, ktorú sme nikdy predtým ani okrajovo nevyužívali. To je jeden z dôvodov, prečo sme si zvolili využitie Google Charts, keďže ako je známe, Google poskytuje dobre napísané dokumentácie. Ďalším dôvodom bolo to, že táto možnosť je zadarmo. Avšak musíme priznať, že pri opätovnom riešení podobného problému, by sme siahli po iných možnostiach tvorby grafu.

6.6 Import a export

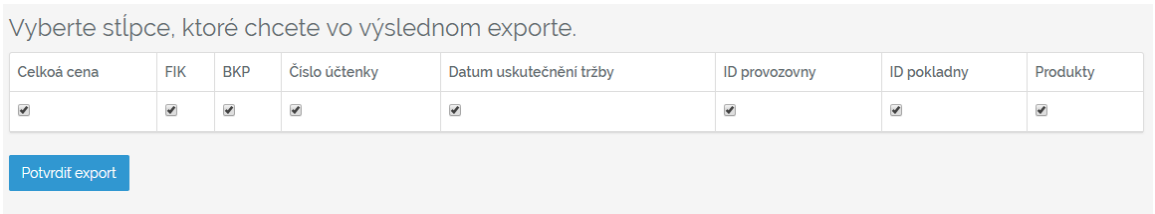
Hromadné operácie s dátami, ako vkladanie (import) a vyberanie (export) zo systému, sú veľmi užitočné, napríklad na prenášanie informácií z jedného informačného systému do druhého, alebo na zálohovanie dát, alebo pre prácu mimo webovej aplikácie, napríklad v exceli.



Obr. 6.2: Zobrazovanie histórie tržieb.

- **Import -** pre import tržieb je nutné nahranie CSV[3] súboru s presne predpísaným formátom. Tento formát je presne napísaný na stránke, alebo si užívateľ môže stiahnuť ukážkový súbor. Samotný import prebieha tak, že je nahraný súbor otvorený a zpracovávaný riadok po riadku. Riadok následne podlieha určitým kontrolám správnosti formátu a ak je všetko v poriadku, tržba z riadku je vložená do databázy. Výsledok importu je zobrazený v HTML elemente `iframe`, čo je v podstate prostriedok na zobrazovanie html dokumentu vo vnútri iného html dokumentu. V tomto iframe je vypísaný každý riadok zo súboru pre import s informáciou, či sa ho podarilo správne importovať alebo nie.

- **Export -** dáta sú exportované do CSV súboru, rovnako ako importované dáta. Užívateľ si má možnosť vybrať, ktoré informácie o tržbách potrebuje, ako je to znázornené na obrázku 6.3. Podľa tohoto vstupu sa vygeneruje CSV súbor pre všetky tržby z danej spoločnosti. Na začiatok súboru sme umiestnili znaky \xFF\xBB\xBF, vďaka čomu excel rozpozná, že súbor je vo formáte UTF-8. Názov vyexportovaného súboru je označený dátumom a časom, aby bolo jasne dané, kedy bol export uskutočnený.



Obr. 6.3: Export tržieb.

[3]Comma-separated values

6.7 Testovanie

Testovanie webovej aplikácie je v praxi často zanedbávané z časových a finančných dôvodov, avšak v konečnom dôsledku dokáže o dosť uľahčiť prácu, keď sa chyby odhalia vopred. V tejto časti práce popíšeme, akými všetkými spôsobmi sme sa snažili docieliť plnú funkčnosť webovej aplikácie a ako sme sa snažili odhaliť chyby. Priebežne popri vývoji, vždy po imlementácii nejakej časti sme na najnižšej úrovni testovania ručne skúšali funkčnosť toho, čo nové sme pridali, ale aj funkcie, čo mali na to nejakú náväznosť. To považujeme za samozrejmosť, ktorú by mal každý programátor pri vývoji aplikácie robiť.

Po dokončení finálneho prototypu aplikácie bolo treba nejako túto aplikáciu otestovať ako celok. To by sa dalo rozdeliť do troch častí. V prvej bol poskytnutý prototyp ôsmim subjektom, ktoré mali ručne otestovať aplikáciu z používateľského hľadiska. Na podnety od týchto subjektov sme jemne upravovali zobrazenie dát v dashboarde a detaile spoločností, graficky trochu preštylizovali celkový vzhľad aplikácie a taktiež boli odhalené 2 bugy. Druhou časťou bolo ručné testovanie mnou, ktoré som zameral na okrajové správanie aplikácie. Taktiež som aplikáciu testoval na viacerých webových prehliadačoch. Keďže by aplikácia mala podporovať responzívne zobrazovanie, skúšal som ju aj na viacerých mobilných zariadeniach. Ale to len čiastočne, keďže webhosting (webzdarma), ktorý poskytoval server, kde som aplikáciu nasadil, nepodporoval určité závislosti, ktoré sú pre plné fungovanie našej aplikácie potrebné. Zámer otestovania responzivity bol však splnený. Táto časť testovania taktiež odhalila niekoľko chýb, z ktorých bola väčšina opravená.

Poslednou fázou testovania boli automatizované jednotkové testy (unit testing). Laravel dosť spríjemňuje toto testovanie, napríklad vytvorením a použitím novej databázy, ktorá je využívaná len pre testovanie. Ďalšou vecou, ktorú využívame pri testovaní, je `Factory`, vďaka čomu vytvárame falšované dáta (mocky), s ktorými následne pracujeme. Týmto testujeme základne funkčné kamene aplikácie, ako napríklad autentizáciu, prihlásenie a základné operácie systému alebo HTTP POST dotaz na aplikáciu. Ďalším objektom testovania sú modely. U tých skúšame ich vytváranie a základné používanie.

Kapitola 7

Záver

Táto práca je o webovej aplikácii pre elektronickú evidenciu tržieb. Podarilo sa nám ju úspešne otestovať a až na drobné nedostatky som s ňou spokojný. Pri riešení tejto práce som si musel naštudovať postupy pre tvorbu webových aplikáciu, technológie pre ich tvorbu a hlavne zoznámiť sa s problematikou PHP frameworku Laravel, pomocou ktorého sme aplikáciu vytvárali. Zároveň som sa musel zoznámiť s problematikou elektronického evidovania tržieb z legislatívneho aj technického hľadiska. Podarilo sa nám vytvoriť systém umožňujúci registráciu viacerých podnikateľov, ktorí majú možnosť spravovať svoje spoločnosti z hľadiska elektronického evidovania tržieb a prezerať si históriu takýchto tržieb.

Vďaka tejto práci som sa toho veľa naučil. Napríklad nové technológie pre tvorbu webových aplikácií, základnú ale aj pokročilú prácu s Laravel frameworkom alebo systematické riešenie problémov. Táto práca ma taktiež obohatila o znalosť naštudovania si problému z hľadiska legislatívy, čo mi príde veľmi prospešné, pretože práve takéto novo vzniknuté zákony, ako napríklad elektronické evidovanie tržieb, otvára nové možnosti na trhu. Tiež som si vďaka tejto práci uvedomil, ako je užitočné mať správne navrhnutý systém ešte pred implementáciou. Pri opätovnom riešení podobného problému by som návrhu venoval oveľa viac času. Až pri implementácii som si začal uvedomovať určité zle navrhnuté časti systému a bohužiaľ, nie všetko som dokázal upraviť. Projekt sme zverejnili pod slobodnou licenciou ako open-source na GitHub platforme. Aj keď má systém určité nedostatky, je pripravený tak, že s neveľkým úsilím by sa mohol doladiť na používanie v praxi.

7.1 Čo by sa dalo zdokonaliť

Na záver tejto práce by som chcel ešte popísať niektoré časti systému, ktoré by sa dali vyladiť a modifikovať k dokonalešiemu fungovaniu vytvorenej webovej aplikácie. Ako už bolo spomínané na konci sekcie 6.2, bolo by dobré pretvoriť oprávnenia. Ďalším vhodným rozšírením do budúcnosti by bolo dotvoriť import a export pre väčší obsah dát, ako napríklad dáta spoločností a užívateľov. To by bolo užitočné pri kompletnom prechode z jedného systému do druhého. Podľa môjho názoru, najdôležitejšia vec, ktorú by bolo potrebné upraviť, je zadávanie nových tržieb. Zdá sa mi, že z jedného pohľadu použitia je neúplný a z druhého pohľadu zase zbytočný. Konkrétne myslím na vkladanie produktov pod tržbu. Jednou z možností by bolo pridať tabuľku produktov do databázy, umožniť ich vkladať a upravovať. Pri produktoch by bola vyplnená cena aj daň. Takto by bolo možné vkladať do tržby produkty jednoduchým selectom s rozšírením o vyhľadávanie. Druhou možnosťou by bolo úplne vylúčiť produkty z tržieb. Pre evidovanie by sa zadávala len celková cena tržby a daň pod akou by sa mala evidovať. Pre rôzny druh obchodu by sa hodil iný prístup, preto by možno nebolo zlé naimplementovať oba spôsoby zadávania a v nastaveniach spoločnosti, by bolo možné vybrať, ktorý spôsob sa k danej spoločnosti viac hodí.

Literatúra

[1] *10 PHP Frameworks For Developers - Best of*. [Online; navštíveno 22.04.2017].
URL http://www.hongkiat.com/blog/best-php-frameworks/

[2] *Bootstrap - The world's most popular mobile-first and responsive front-end framework*. [Online; navštíveno 20.04.2017].
URL http://getbootstrap.com/

[3] *Etržby - elektronická evidence tržeb*. [Online; navštíveno 16.04.2017].
URL http://www.etrzby.cz

[4] *Git - Správa verzí*. [Online; navštíveno 21.04.2017].
URL https://git-scm.com/book/cs/v1/%C3%9Avod-Spr%C3%A1va-verz%C3%AD

[5] *jQuery Introduction*. [Online; navštíveno 21.04.2017].
URL https://www.w3schools.com/jquery/jquery_intro.asp

[6] *Laravel Middleware*. [Online; navštíveno 23.04.2017].
URL https://www.tutorialspoint.com/laravel/laravel_middleware.htm

[7] *Laravel - The PHP framework - ducumentation*. [Online; navštíveno 22.04.2017].
URL https://laravel.com/docs/5.3

[8] *Using google charts*. [Online; navštíveno 21.04.2017].
URL https://developers.google.com/chart/interactive/docs/

[9] *Working with jQuery DataTables*. [Online; navštíveno 21.04.2017].
URL https://www.sitepoint.com/working-jquery-datatables/

[10] *Úvod do CSS*. [Online; navštíveno 20.04.2017].
URL http://www.webtvorba.cz/css/uvod-do-css.html#zaciname

[11] Gutmans, A.; Rethans, D.; Bakken, S. S.: *Mistrovství v PHP 5*. Brno : Computer Press, 2005, ISBN 80-251-0799-X.

[12] Hruška, T.; Máčel, M.: *Pokročilé informační systémy - Studijní opora*. FIT VUT v Brně, 2012, [Online; navštíveno 23.04.2017].
URL https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaPISAnalyzaNavrhImplementace.pdf

[13] Kilicdagi, A.; Yilmaz, H. I.: *Laravel design patterns and best practices: enhance the quality of your web applications by efficiently implementing design patterns in Laravel*. Packt, Jul 2014, ISBN 978-1783287987.

Prílohy

Príloha A

Obsah CD

- `doc/` - Technická správa
  - `source/` - Zdrojové súbory technickej správy pre LaTeX
  - `xgabon00_report.pdf` - Technická správa v PDF
- `src/` - Zdrojové súbory aplikácie
- `manual.txt` - Návod na inštaláciu

Príloha B

Návod na inštaláciu

- 1. Naklonujete Git repozitár (https://github.com/MichalGabonay/bp-eet) alebo skopírujte súbory z CD umiestnené v priečinku `src`.
- 2. Vytvorte `.env` súbor v koreňovom adresári, skopírovaním a premenovaním `.env.example`.
- 3. Nainštalujte všetky závislosti vložením príkazu `composer install` do príkazového riadku. Musíte sa nachádzať v koreňovom adresári projektu. Ak nemáte nainštalovaný Composer, postupujte podľa inštrukcií na stránke https://getcomposer.org/doc/00-intro.md.
- 4. Vygenerujte unikátny kľúč pre vašu aplikácie príkazom `php artisan key:generate`
- 5. Vytvorte prázdnu databázu a informácie o tejto databáze vložte do súboru .env (server, meno db, užívateľ a heslo)
- 6. Vytvorte tabuľky v databáze jednym z nasledujúcich spôsobov. Otvorte URL `example.cz/migrate` (namiesto example.cz vašu base URL) alebo zavolajte príkaz `php artisan migrate` a `php artisan db:seed`.

Systém je pripravený k použitiu. Zaregistrujte užívateľa, vytvorte spoločnosť a vložte platný certifikát k spoločnosti. Teraz je možné evidovať tržby.