VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

OPENSOURCE ŘEŠENÍ EET S EVIDENCÍ SKLADU A PORTÁLEM PRO ZÁKAZNÍKY

OPENSOURCE SOLUTION OF ELECTRONIC SALES RECORDS WITH WAREHOUSING AND PORTAL FOR CUSTOMERS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **BC. JAKUB ŠVESTKA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/20286/2017/xsvest05

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Švestka Jakub, Bc.**
Obor: Informační systémy
Téma: **Opensource řešení EET s evidencí skladu a portálem pro zákazníky**
Opensource Solution of Electronic Sales Records with Warehousing and Portal for Customers
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s technologiemi pro vývoj informačních systémů na webu a dostupnými frameworky.
2. Analyzuje pořadavky na informační systém pro prodej libovolného zboží zahrnující EET, s možností evidence skladových zásob, zákazníků a uživatelů EET a různými statistikami. K nabízení zboží uživateli budou využita asociační pravidla získaná vhodným algoritmem. Dále bude systém obsahovat portál pro zákazníky, kteří zde budou prohlížet a stahovat účtenky. Účtenky bude možné také zaslat na e-mail nebo zobrazit online pomocí vygenerovaného URL.
3. Navrhněte informační systém dle požadavků.
4. Navržený systém implementuje a ověřte jeho funkčnost na vzhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a další možné pokračování tohoto projektu.

Literatura:

- Welling, L., Thomsonová, L.: PHP a MySQL: rozvoj webových aplikací. Vyd. 1. Praha: SoftPress, 2003, 910 s. ISBN 80-86497-60-7.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9
- Han, J., Kamber, M.: Data Mining - Concepts and Techniques, 2nd Edition. Morgan Kaufmann Publishers, 2006.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% 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: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

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

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá studií webových technologií pro vývoj informačních systémů, analýzou požadavků, návrhem a implementací systému pro prodej zboží, který podporuje evidenci tržeb do `EET`, evidenci skladu a dále nabízí portál pro zákazníky sloužícího pro předání účtenek a zajištění podpory pomocí tiketovacího systému. Návrh systému vychází z již realizované pokladny a bude oproti ní pokročilejším a daleko univerzálnějším řešením. Aplikace je postavena na běžně dostupných a bezplatných technologiích, a to `PHP7.1` s použitím frameworku *Nette*, `CSS` frameworku *Bootstrap* a knihovny *jQuery*. Velmi užitečnou funkcí pokladny je našeptávač produktů, který je založen na algoritmu *Apriori*. Našeptávač našeptává produkty na základě položek z akutálně otevřeného účtu a analýzy zakoupených položek z již realizovaných dokladů. Kompletní řešení bude následně uvolněno veřejnosti jako opensource, které dosud neexistuje.

Abstract

The aim of this term project is to study web technologies for developing information systems, requirements analysis, a system design for selling goods which supports sales recording to electronic records of sales and stock recording. It also offers the customer portal for bills handover and customer support by a ticket system. The system design is based on already developed cash register and in comparison with it will be a more advanced and far more universal solution. The application is built on commonly available and free technologies, such as `PHP 7.1` with *Nette* framework, `CSS` framework *Bootstrap* and *jQuery*. Very useful function of cash is the product suggester which is based on the *Apriori* algorithm. The suggester suggests products based on items from the actual opened bill and analysed receipts with previously purchased items. The complete solution will then be released to the public as an opensource, which does not exist yet.

Klíčová slova

EET pokladna, prodej zboží, registrační pokladna, elektronická evidence tržeb, portál pro zákazníky, evidence skladu, PHP, Nette, Bootstrap, AdminLTE, našeptávač produktů, Apriori

Keywords

e-sales cash, goods sale, cash register, electronic records of cash sales, customer portal, stock records, PHP, Nette, Bootstrap, AdminLTE, product suggester, Apriori

Citace

ŠVESTKA, Jakub. *Opensource řešení EET s evidencí skladu a portálem pro zákazníky.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Opensource řešení EET s evidencí skladu a portálem pro zákazníky

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Bc. Jakub Švestka
24. dubna 2018

Poděkování

Rád bych poděkoval Ing. Vladimíru Bartíkovi, Ph.D. za cenné rady, osobní přístup a čas věnovaný vedení mé diplomové práce. Bc. Antonínu Markovi za cenné rady, připomínky, a dále Bc. Katrin Kalay za pomoc s formální podobou práce. Největší díky však patří mé rodině za podporu a povzbuzení.

Obsah

Kapitola 1

Úvod

Počátkem června 2016 byl spuštěn ostrý provoz EET[1], a tím postupně v několika fázích vznikla povinnost podnikatelům evidovat tržby elektronickou formou. Jejím smyslem je zaznamenání každé jednotlivé tržby podnikatele do systému finanční správy a opatření účtenky unikátním daňovým kódem.

O EET jsem se začal zajímat před zahájením II. fáze jejího nasazení, do kterého spadali podnikatelé pro maloobchod/velkoobchod a týkala se i mého rodinného příslušníka provozujícího maloobchodní prodej elektroniky, jenž mě pověřil nalezením vhodného řešení včetně jeho nasazení. Po analýze nabídek EET pokladen a studiu metodických pokynů k aplikaci zákona o evidenci tržeb jsem zjistil, že komunikace není implementačně náročná a díky zkušenosti s tvorbou webových aplikací a již existující knihovně pro komunikaci se systémem finanční správy jsem se rozhodl pro vlastní implementaci. Implementovaná pokladna obsahuje základní sadu funkcionality, a to katalog zboží s jednoduchou evidencí skladu, historii vydaných daňových dokladů a samotný prodej. Pokladna úspěšně funguje již necelý rok, ale díky omezené funkcionalitě není vhodná pro nasazení u jiných obchodníku, kde mohou její nedostatky bránit ve výkonu prodejní činnosti.

V této práci se zabývám vývojem EET pokladny, jež bude pokročilejším a univerzálnějším řešením než výše uvedená první verze, a bude tak vhodná k použití u vícero podnikatelů. Pokladna by měla co nejvíce snížit náklady na pořízení potřebného hardwaru a náklady na provoz, jako je třeba nutný tisk účtenek z důvodu absence možnosti vydání elektronické účtenky. Kompletní řešení bude následně uvolněno jako otevřený software (open-source) z důvodu jeho absence mezi komunitou, a dále pomůže podnikatelům, pro které je zavedení EET finančně náročné.

Pokladna bude rozdělena do tří hlavních částí, mezi první z nich patří samotný *prodej*. Ten bude umožňovat veškerou funkcionalitu potřebnou pro realizaci prodeje, tedy od markování zboží až po zaevidování účtenky do systému finanční správy. Kromě běžného tisku dokladů pomocí tiskárny systém umožní vydání elektronického dokladu. Jeho předání bude realizováno jedním z následujících způsobů:

- elektronickou poštou
- vyzvednutím dokladu na portálu pro zákazníky pomocí obdrženého unikátního kódu v případě neregistrovaného zákazníka
- zobrazením dokladu v historii dokladů v portálu pro zákazníky v případě registrovaného zákazníka a přiřazení dokladu pod jeho zákaznický účet.

[1] Elektronická evidence tržeb.

Další částí bude administrace neboli *správa pokladny*, jež bude sloužit ke správě katalogu zboží, zákazníků, zaměstnanců, tiketovacího systému podpory, skladové evidenci a dalších nastavení potřebných pro provoz. Poslední částí je *portál pro zákazníky*, který zákazníkům nabídne historii vydaných dokladů, tiketovací systém pro komunikaci s prodejcem a sekce sloužící pro podporu, jako jsou často kladené otázky, soubory ke stažení a aktuality.

Cílem této práce je nejprve čtenáře seznámit s webovými technologiemi, kde bude lehce nastíněn samotný vznik webu a popsán `HTTP` protokol, jenž je jejich základem. V poslední řadě bude definována webová aplikace s následným uvedením výhod/nevýhod při jejím použití a popsán princip a účel webové služby. Třetí kapitola pojednává o elektronické evidenci tržeb v ČR a jejím principu, jednotlivým krokům procesu evidence a technickému řešení. Čtvrtá kapitola se zabývá analýzou požadavků a zhodnocení již existujících řešeních. V páté kapitole jsou specifikovány požadavky na jednotlivé částí aplikace, definování aktéři systému a popsány vybrané případy užití. Šestá kapitola se zaobírá výběrem vhodných technologií pro vývoj a bližším popisem těch, které budou pro implementaci aplikace použity. Sedmá kapitola je již věnována samotnému návrhu, kde bude popsána dekompozice aplikace na moduly, navržena modelová vrstva, a dále popsán našeptávač produktů, jenž využívá asociačních pravidel. V neposlední řadě bude lehce nastíněno rozložení uživatelského rozhraní aplikace. Osmá kapitola nastíní strukturu aplikace a bude zde popsána implementace důležitých nebo implementačně zajímavých částí. Předposlední kapitola popisuje výsledky testování, díky nimž byla získána zpětná vazba od účastníků testování a provozovatele, jenž pokladní aplikaci využíval v reálném provozu. Na základě uživatelských podnětů byly provedeny změny, díky kterým došlo ke zvýšení uživatelské přívětivosti. Závěr práce popisuje zhodnocení dosažených výsledků a další možné pokračování vývoje.

Kapitola 2

Webové technologie

Tato kapitola pojednává o vzniku webových technologiích a protokolu `HTTP`, který je s ním spjat. Dále je vysvětleno, co je to webová aplikace včetně jejích výhod a nevýhod. Poslední část je věnovaná webovým službám, jež jsou v současnosti hojně používané.

2.1 World Wide Web

World Wide Web (neboli WWW, Web) je rozsáhlá služba celosvětové sítě internet zpřístupňující informace a zdroje, které jsou identifikovány pomocí `URI`. Podstatou `WWW` jsou hypertextové dokumenty obsahující hypertextové odkazy, jenž dále odkazují na další hypermediální obsah a tvoří tak mezi dokumenty vazby. Odkazovaný obsah může být umístěn kdekoliv v internetové síti a nemusí být nutně hypertextový. Hypertextový protokol dovoluje přenášet soubory rozličných typů jako textové dokumenty, multimediální soubory, binární soubory apod. Uživatel pomocí těchto odkazů „přeskakuje“ mezi hypertextovými dokumenty a dostává se tak k hledaným informacím bez nutnosti znalosti jakýchkoliv příkazů. Díky tomu je ovládání jednoduché a intuitivní.

Historie webu sahá do roku 1991, kdy došlo k jeho uvolnění pro veřejnost. Toto období bývá označováno termínem `Web 1.0`, jenž charakterizuje web jako stránky s minimální interakcí mezi uživatelem a serverem. Dalším typickým rysem je, že stránky jsou určené pouze ke čtení – uživatel netvoří obsah stránek.

Web se stal nejpoužívanější službou internetu a jeho popularita stále roste. Většina aplikací se stěhuje do webového prostředí, a tím se stávájí uživatelům dostupnější díky minimálním nároků na potřebný hardware/software. `Web 2.0` je od roku 2004 další etapou vývoje webu, jež je charakteristická následujícími vlastnostmi [8]:

- uživatel je vtažen do tvorby obsahu (např. přidání komentáře u článku, vytvoření uživatelského účtu, nahrání fotografie)
- obsah není tvořen pouze vlastníkem webu – dochází k decentralizaci autorit, spolupráce na tvorbě
- interaktivní komunikace
- propracovaná hyperlinková struktura
- webové stránky se stávají platformou poskytující webové aplikace
- sdílení a znovuvyužití informací.

V současné době se již mluví o etapě `Web 3.0`, jelikož požadavky na web stále rostou. Kancelářské aplikace se stěhují na web, stejně jako další aplikace všeho druhu. Například aplikace Google Docs[1] umožňuje tvorbu dokumentů pomocí `WYSIWYG`[2] editoru přímo v prohlížeči. Jako další příklad bych uvedl službu Overleaf[3] určenou pro sázení dokumentů v LaTeXu. Hlavní výhodou zmíněných služeb je jednodušší proces tvorby, rychlé publikování, možnost spolupráce při tvorbě s více lidmi, okamžité sdílení a funkce uchovávání starších verzí souborů.

2.2 Webové aplikace

Webová aplikace je kolekcí webových stránek poskytovaných webovým serverem přes počítačovou síť internet, nebo intranet v případě umístění v lokální podnikové síti. Uživateli pro používání aplikace stačí mít nainstalován pouze webový prohlížeč, který plní funkci tenkého klienta, neboť sám o sobě logiku aplikace nezná. Není tedy nutná instalace speciálního softwarového balíku. Komunikace je založena na síťové architektuře klient/server probíhající skrze `HTTP` protokol, jenž je popsán v 2.3.

2.2.1 Výhody webových aplikací

- **Nezávislost na platformě**
  Webová aplikace funguje shodně na všech platformách. Uživateli stačí mít webový prohlížeč, který je v dnešní době součástí každé platformy, včetně platforem určených pro mobilní zařízení. Místo psaní variant pro různé platformy stačí teoreticky aplikaci napsat jednou a tu nabídnout téměř kdekoliv.

- **Distribuce nových verzí**
  Dodavatel webového řešení již nemusí ke klientovi jezdit či mu zasílat nové verze, ale pouze vzdáleně provede opravu na jediném centrálním místě (serveru). Uživatel tak má vždy přístup k aktuální zveřejněné verzi aplikace.

- **Přizpůsobení stránky**
  Uživatelé si mohou pomocí funkcí (doinstalovaných pluginů) dostupných ve webovém prohlížeči stránku přizpůsobit a omezit tak její funkce/obsah. Dále pro hendikepované návštěvníky jsou webové technologie nespornou výhodou oproti desktopovým aplikacím, které nejsou pro takové osoby uzpůsobeny, jestliže nejsou vyvíjeny přímo pro tento okruh lidí.

2.2.2 Nevýhody webových aplikací

- **Vykreslovací jádra nedodržují standardy**
  Většina vývojářů se potýkala s problémy vykreslovacích jader prohlížečů. Vývojář tedy musel kód webové aplikace přizpůsobovat danému vykreslovacímu jádru, aby zachoval korektní zobrazení. Nicméně v dnešní době s nástupem nových vykreslovacích jader je tento problém minimální.

[1] Dostupné na http://docs.google.com/.
[2] Akronym anglické věty „What you see is what you get“ → „co vidíš, to dostaneš“.
[3] Dostupné na https://www.overleaf.com/.

- **Aplikace třetích stran**
  Aplikace třetích stran činí doposud některé webové aplikace nepoužitelné na některých zařízeních. Patří mezi ně např. *Adobe Flash Player*, *Microsoft Silverlight*, *ActiveX* apod. Aktuálně jsou postupně vytlačovány samotnými výrobci webových prohlížečů z důvodů bezpečnostních rizik a nástupu `HTML5`, který již poskytuje potřebné funkcionality [26].

- **Závislost na poskytovateli aplikace**
  Očividnou nevýhodou využití tenkého klienta je vysoká závislost na poskytovateli aplikace. Pokud se poskytovatel rozhodne provoz poskytované služby ukončit či dočasně pozastavit z důvodu úprav, pak nelze službu používat, na rozdíl od lokálně provozovaného softwaru (tlustého klienta).

- **Nutnost připojení k Internetu**
  Činnost webových aplikací je závislá na připojení k Internetu. Nicméně současné době téměř každá domácnost disponuje internetovým připojením. A pokud je člověk mimo domácnost, lze využít mobilního připojení či veřejných bezdrátových sítí, které jsou dnes standardem.

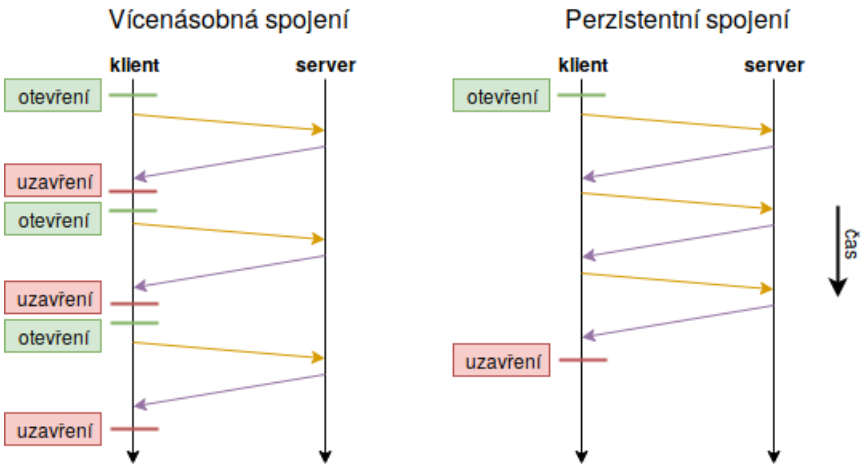
2.3 HTTP

Hypertext Transfer Protocol (`HTTP`) je aplikační protokol pro distribuované, spolupracující a hypermediální[4] informační systémy [24]. Protokol je určen pro výměnu hypertextových dokumentů ve formátu `HTML` mezi klientem a serverem, které jsou přístupné pod určitou `URL` adresou specifikující jednoznačné umístění zdroje v Internetu [14]. V současné době se používá i pro přenos dalších informací. Díky standardizovanému `MIME`[5] rozšíření dovoluje přenášet soubory rozličných typů. Dále se používá ve spojení s `XML` pro tzv. webové služby [23].

Protokol `HTTP` je postaven na principu dotaz-odpověď, kdy klient iniciuje spojení s webovým serverem. V požadavku od klienta je uvedena `URL` adresa požadovaného zdroje, informace o schopnostech prohlížeče apod. Server následně odpoví několika řádky textu o tom, zda se podařilo dokument najít, jakého je typu (dle standardu `MIME`), za kterými v případě úspěchu následuje samotný obsah požadovaného zdroje. Tímto je spojení ukončeno a v případě dalšího požadavku musí klient zaslat další nezávislý dotaz v novém spojení. Kvůli tomu zbytečně dochází k zatěžování serveru dalšími požadavky na spojení a delšímu načítaní obsahu stránky. Toto nežádoucí chování může webový prohlížeč obejít tak, že v iniciačním požadavku v hlavičce `Connection` uvede hodnotu `keep-alive` a nastaví požadovaný časový limit, po který má být spojení aktivní. Tímto další požadavky na stejný server proběhnou v jednom spojení – viz obrázek 2.1. Od verze `HTTP/1.1` je perzistentní spojení implicitní.

V případě požadavků v několika spojeních nelze z hlediska serveru poznat, zda spolu dotazy souvisejí. Kvůli této vlastnosti se `HTTP` protokolu říká *bezstavový protokol* – protokol neumí uchovávat stav komunikace. Tato vlastnost je z hlediska některých funkcionalit webových aplikací nežádoucí (např. potřeba uchovávat informaci o identitě zákazníka a obsahu košíku). K tomuto účelu byl protokol rozšířen o tzv. `HTTP cookies`, s jejichž pomocí dostane klient (webový prohlížeč) od webového serveru jednoznačný identifikátor, který

[4] Rozšíření konceptu hypertextu směrem ke komplexnějšímu audiovizuálnímu obsahu.
[5] Multipurpose Internet Mail Extensions.



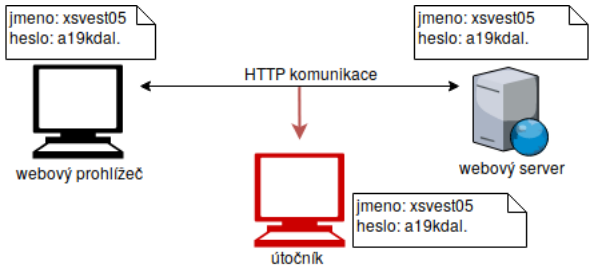
Obrázek 2.1: Schéma vícenásobného spojení versus perzistentní spojení

následně posílá v každém dalším požadavku. Takto je možné uchovávat informace o stavu spojení na počítači uživatele.

`HTTP` využívá ve většině případů `TCP/IP` jako transportní vrstvu. Komunikace mezi klientem a serverem probíhá skrze `TCP` port `80`, jenž je implicitní. Není doporučeno používat jiný port, jelikož klient nedokáže sám zjistit, na jakém portu webový server naslouchá a spoléhá tak na implicitní port. Samotný `HTTP` protokol není vhodný pro přenos citlivých dat, jelikož komunikace není nikterak šifrována. Tento nedostatek byl odstraněn zavedením zabezpečené varianty protokolu, kterou je `HTTPs` 2.3.1.

2.3.1 HTTPs

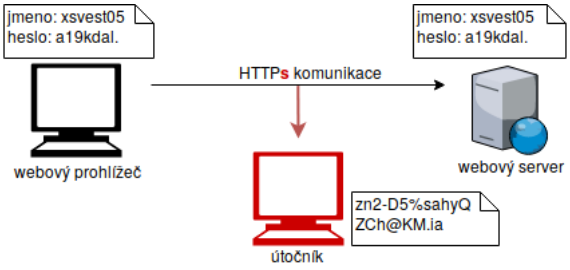
Jak již bylo zmíněno, komunikace skrz `HTTP` protokol není nikterak šifrována. Data, která si vyměňuje klient se serverem, může na síti kdokoliv odposlouchávat – viz obrázek 2.2. To je u současných webových aplikací (např. internetové bankovnictví) velmi nežádoucí.



Obrázek 2.2: Úspěšný odposlech `HTTP` komunikace

Proto došlo k vyvinutí protokolu `HTTPs`, jenž umožňuje zabezpečenou komunikaci v počítačové síti. Převážně se používá pro komunikaci webového prohlížeče s webovým serverem – viz obrázek 2.3. `HTTPs` využívá `HTTP` protokolu, kdy je mezi transportní (vetšinou `TCP/IP`) a aplikační vrstvu (`HTTP`) vložena vrstva založená na protokolu `SSL`[6] nebo `TLS`[7]. Díky této mezivrstvě je zajištěna autenticita, důvěrnost přenášených dat a jejich integrita.

[6] Secure Sockets Layer.
[7] Transport Layer Security.



Obrázek 2.3: Neúspěšný odposlech `HTTPs` komunikace

`HTTPs` komunikace oproti `HTTP` probíhá standardně přes `TCP` port `443`. Díky odlišným portům lze na serveru provozovat obě verze současně. Nicméně ve většině případů dochází k nucenému přesměrování z nezabezpečené verze (`HTTP`) na zabezpečenou verzi (`HTTPs`) z důvodu zajištěné bezpečnosti pro uživatele [14].

2.3.2 HTTP/2

Jedná se o novou verzi `HTTP` protokolu, jež je standardizována v `RFC 7540`. Zatím není příliš využívaná webovými servery – dle statistik serveru *W³ Techs Web Technology Surveys* vyplývá, že ji k prosinci 2017 využívá pouze 22.5 % ze všech webových aplikací [1].

Předchozí hojně využívaná verze `HTTP/1.1` přišla sice s perzistentním spojením, kdy server po odpovědi nezavírá `TCP` spojení a umožní tak klientovi položit další dotazy v jednom spojení (viz 2.1). Ovšem nevýhoda je v tom, že klient musí vždy čekat na odpověď od webového serveru, než může položit další dotaz (v stejném `TCP` spojení). Aby tuto nevýhodu webové prohlížeče obešly a zrychlily tak načítání stránky, navazují se serverem několik paralelních `TCP` spojení, mezi něž své dotazy rozkládají. Cenou tohoto řešení je daleko větší režie, jež nejvíce trápí provozovatelé webových serverů a síťových infrastruktur [19].

Cílem `HTTP/2` je zrychlit vzájemnou komunikaci mezi klientem a serverem a minimalizovat tak zátěž na obou stranách včetně samotné síťové infrastruktury. Koncept protokolu je založen na takzvaných proudech (streamech), jenž paralelně přenáší všechny klientovy dotazy v jednom `TCP` spojení. Každý dotaz má svůj vlastní proud, jenž vznikne při položení dotazu ze strany klienta. Jakmile je serverem odeslána celá odpověď klientovi, je proud uzavřen. Na rozdíl od vytvoření `TCP` spojení však založení a ukončení proudu nezahrnuje prakticky žádnou režii ani zpoždění. Pro založení nového proudu klient jednoduše položí dotaz s novým identifikátorem proudu, a tím je proud založen. Naproti tomu ukončení proudu je serverem realizováno příznakem `END_STREAM`, kterým označí poslední paket v odpovědi [25].

2.4 Webová služba

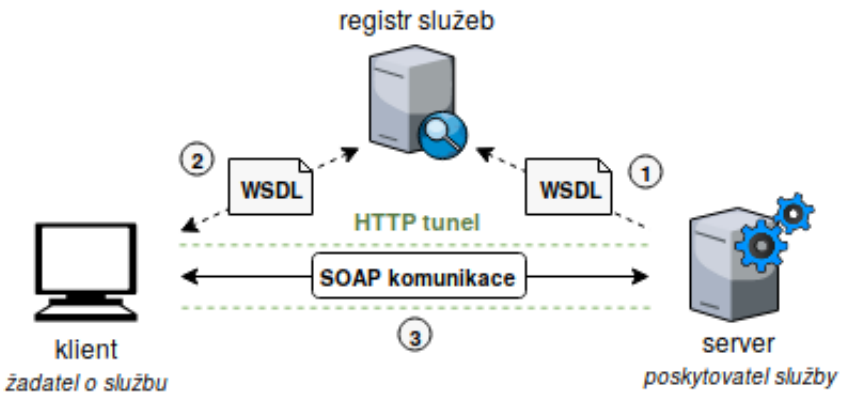
Jedná se o softwarový systém umožňující interakci dvou strojů na síti [9]. Koncept webových služeb umožňuje skrz `HTTP` protokol volat funkce na vzdáleném serveru (poskytovatel služby), předávat jim parametry a pracovat s vrácenými výsledky. `HTTP` protokol je využíván z důvodu nedostatku důvěry ze strany firewallů již zavedených systémů, které často propouští pouze webovou komunikaci a při zavedení nového protokolu (využívajícího jiný port) by na nich mohlo docházet k blokování provozu. Výhodou webových služeb je nezávislost na programovacím jazyce, v jakém je služba napsána i z jakého jazyka ji voláme. Roz-

díl nastává ve výměně dat, kterým je obvykle XML. Díky tomu jakýkoliv program schopný pracovat s XML daty může plnit roli klienta, využívající webovou službu, či serveru, jenž ji poskytuje [29].

Jako protokol pro práci s webovými službami se nejčastěji používá SOAP 2.4.2, který vychází z jednoduššího protokolu XML-RPC, jehož vývoj je v současné době ukončen [10]. K webové službě založené na SOAP protokolu lze dále připojit tzv. WSDL dokument, sloužící k popisu veškeré funkčnosti webové služby, jež poskytuje (viz 2.4.3). Protokoly SOAP a WSDL využívají pro svůj popis syntaxi jazyka XML z důvodu snadné strojové zpracovatelnosti, možnosti rozšiřitelnosti a byly navrženy tak, aby byly co nejméně závislé na určité verzi standardu XML.

2.4.1 Komunikace po síti

Komunikace je znázorněna na obrázku 2.4. Při provádění webové služby specifikace počítá se třemi účastníky. Jsou jimi již výše zmíněný žadatel o službu (klient) a poskytovatel služby (server), třetím účastníkem je dosud nezmíněny registr služeb, který bude popsán níže. Nicméně použití registru není povinné – klient může obdržet specifikaci služby jiným způsobem.



Obrázek 2.4: Schéma komunikace klienta s webovou službou

V části ① poskytovatel zveřejňuje definici služby v registru pomocí UDDI[8] protokolu. Tímto se služba stává veřejnou a dohledatelnou pro klienty. Část ② znázorňuje vyhledání služby žadatelem v registru služeb, který poskytne bližší informace o službě a samotný WSDL dokument. Tímto má žadatel veškeré údaje o poskytovateli služby a může se s ním spojit ③.

Registr služeb

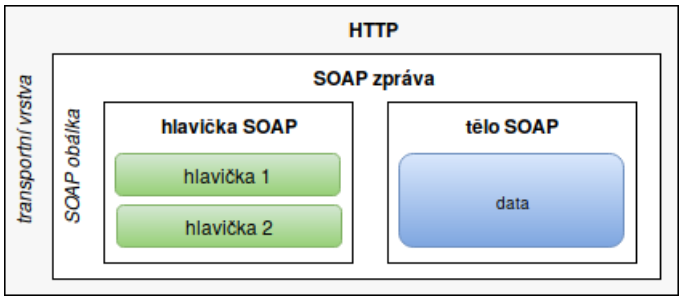
Registr služeb slouží pro registrování a vyhledávání webových služeb. Jedná se o databázi obsahující URL odkazy na dokumenty WSDL a popisy webových služeb. Nejčastěji se využívá mechanismu UDDI, jehož standardizací se zabývá konsorcium OASIS[9]. Tento způsob vyhledání služby ovšem neposkytuje záruku toho, že webová služba bude opravdu dělat to, co je v jejím popisu. Rovněž není nijak ověřena identita webové služby – nemůžeme si být jisti, že autor uvedený v registru je opravdu ten, jenž ji skutečně vytvořil. Alternativou je použití

[8] Universal Description, Discovery and Integration.
[9] Organization for the Advancement of Structured Information.

WSIL[10], která zaručuje autorství a funkce webové služby. WSIL funguje na opačném principu než UDDI, kdy v části ② (obrázek 2.4) dojde nejprve k vyhledání důvěryhodného poskytovatele webové služby. Klient si následně vybere službu, kterou poskytuje a následně požádá přímo poskytovatele webové služby o WSDL dokument. V současnosti je však rozšířenější protokol UDDI než WSIL [23].

2.4.2 SOAP

Protokol SAOP[11] je základem webových služeb. Umožňuje volat vzdálené funkce pomocí XML. Jako aplikační vrstva se využívá HTTP, SMTP[12] nebo MQseries[13]. Nicméně použití s HTTP protokolem dosáhlo daleko většího uplatnění [28, s. 39].



Obrázek 2.5: Struktura SOAP zprávy

Každá SOAP zpráva je jednoduchý XML dokument, obsahující tyto základní elementy [23]:

- **Obálka** – kořenový element zprávy *(povinný)*
- **Hlavička** – obsahuje pomocné informace pro zpracování zprávy *(nepovinný)*
- **Tělo** – obsahuje vlastní zprávu *(povinný)*.

2.4.3 WSDL

Abychom mohli s webovou službou komunikovat, musíme o ní znát základní informace. Především potřebujeme vědět cestu k webové službě (URL adresa, IP adresa, ...), jakými protokoly umožňuje komunikaci, jaké funkce nabízí, a dále popis vstupů a výstupů těchto funkcí. K tomuto účelu slouží protokol WSDL, která popisuje veřejné rozhraní webové služby. Výsledkem tohoto popisu je WSDL dokument, jenž je tvořen několika elementy s následujícím významem [23]:

- **Types** – množina datových typů nějakého typového systému (nejčastěji XML schéma)
- **Message** – udává informace pro konkrétní operaci, které jsou potřeba pro její vykonání (např. vstupy a výstupy)
- **Operation** – definuje funkce podporované službou

[10] Web Services Inspection Language.
[11] Simple Object Access Protocol.
[12] Simple Mail Transfer Protocol.
[13] knihovna pro frontové dočasné držení dat.

- **PortType** – tvoří abstraktní rozhraní webové služby; definuje operace, které mohou být vykonány na jednom či několika koncových bodech
- **Binding** – určuje konkrétní přenosový protokol a formát přenosu zpráv pro konkrétní *portType*
- **Port** – koncový uzel, brána, která je typicky reprezentovaná `URL` adresou služby
- **Service** – množina koncových uzlů, portů.

`WSDL` je založen na `XML`, díky němuž je platformně nezávislý. Zpravidla popisuje `SOAP` komunikace, jelikož většina webových služeb komunikuje tímto protokolem.

2.4.4 Jiné způsoby pro komunikaci

Krom výše uvedených protokolů existují i jiné způsoby pro komunikaci s webovou službou. Je jím například komunikace podle filosofie `CRUD`[14] realizovaná typicky jako `REST`[15] (architektura rozhraní), která je založena na `HTTP` protokolu a jeho metod `POST`, `GET`, `PUT` a `DELETE` [9]. Tyto metody umožňují pracovat s daty na serveru a jsou dostačující pro používání služby.

[14] Create, Read, Update, Delete – vytvoř-zapiš, přečti-vrať, změň, smaž.
[15] Representational State Transfer.

Kapitola 3

EET – elektronická evidence tržeb

V následující kapitole přiblížím princip elektronické evidence tržeb, kdy budu převážně čerpat z oficiálních stránek www.etrzby.cz, které jsou provozovány přímo Finanční správou[1]. Část kapitoly se bude zabývat samotným technickým řešením a popisu rozhraní pro komunikaci s Finanční správu. Nicméně budou vyzdvihnuty pouze důležité části, jelikož pro bližší porozumění je nutné studium samotné technické specifikace[2] a její přepis by zde byl bezpředmětný. Dále bude zmíněno několik existujících řešeních pro elektronickou evidenci tržeb.

3.1 Princip

Elektronická evidence tržeb (EET) je způsob evidence tržeb, kdy jsou údaje o každé transakci obchodníka online posílány na Finanční správu [3]. Díky tomu má Finanční správa potřebné informace, pomocí nichž může efektivně a cíleně provádět daňové kontroly, jejichž výsledkem budou následující přínosy [11]:

- zlepšení podnikatelského prostředí
- férovější podmínky pro zaměstnance (odstranění černých výplat)
- lepší fungování státu (účinnější finanční správa, spravedlivý a efektivní výběr daní).

Podnikatelé pro evidenci tržeb potřebují zařízení s připojením k internetu z důvodu online evidence. Může se jednat o specializovanou elektronickou pokladnu či pouze o pokladní SW běžící na osobním počítači, tabletu nebo chytrém telefonu. Dále musí zajistit předání daňového dokladu zákazníkovi. Zákon o evidenci tržeb neupravuje formát ani způsob předání účtenky zákazníkovi. Předání lze tedy učinit tištěnou formou pomocí tiskárny, nebo elektronickou – např. zaslání emailem, předání odkazu na účtenku, sms účtenka...

3.2 Kroky procesu evidence tržby

EET definuje dva způsoby evidence tržeb, a to *běžný režim (on-line)* a *zjednodušený režim (off-line)*. Jejich rozdíl bude následně přiblížen a dále bude popsána situace, která může nastat při výpadku internetového spojení, a tím nemožnosti tržbu zaevidovat on-line.

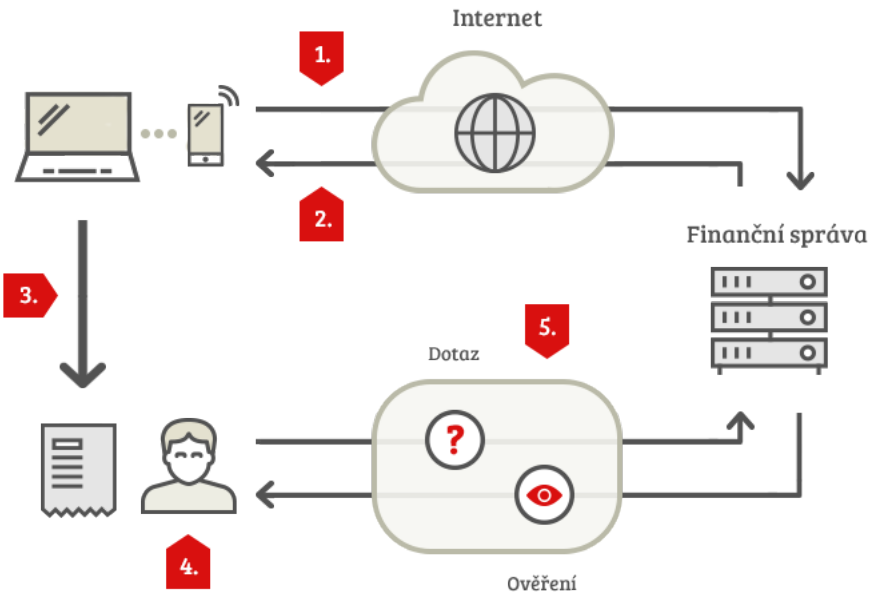
Nyní budu citovány možné kroky procesů včetně schémat, které je znázorňují [11].

[1] Webové stránky: http://www.financnisprava.cz.

[2] Dostupná zde: http://www.etrzby.cz/cs/technicka-specifikace.

3.2.1 Evidence tržeb v běžném režimu

Povinností poplatníka je nejpozději v okamžiku uskutečnění evidované tržby zaslat datovou zprávou údaje o této evidované tržbě správci daně a vystavit účtenku zákazníkovi.



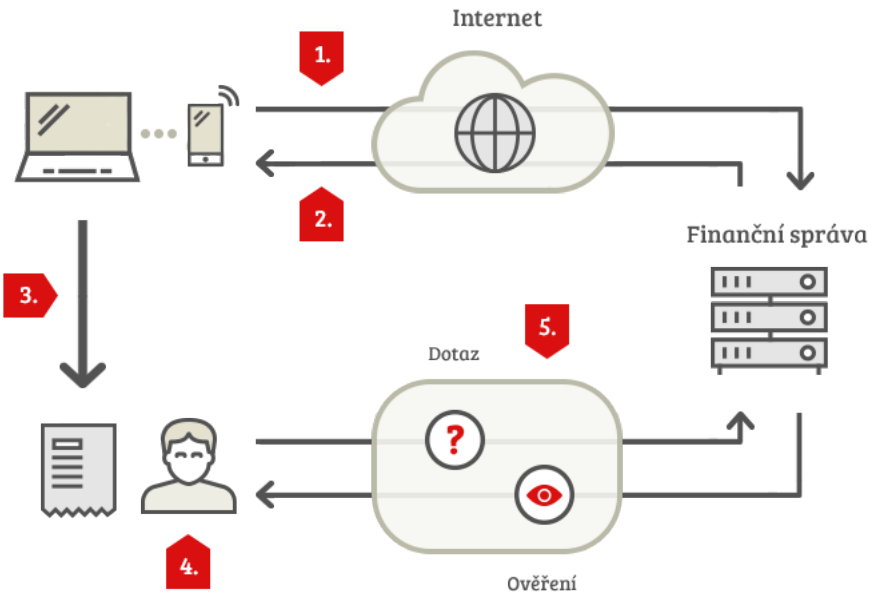
Obrázek 3.1: Kroky procesu evidence tržby v běžném režimu [11]

1. Podnikatel zašle datovou zprávu o transakci ve formátu `XML` Finanční správě.
2. Finanční správa potvrdí přijetí datové zprávy zasláním unikátního kódu `FIK`[3].
3. Podnikatel vystaví účtenku, která obsahuje náležitosti dle specifikace `EET` (např. získaný `FIK` kód) a předá zákazníkovi.
4. Zákazník obdrží účtenku.
5. Zákazník si může ověřit svoji účtenku na Daňovém portále, popřípadě evidovat účtenku do účtenkové loterie *Účtenkovka*[4].

3.2.2 Evidence tržeb při výpadku spojení

Pokud se pokladnímu zařízení nepodaří navázat spojení v nastavené mezní době odezvy (ze zákona nejméně na 2 sekundy), nemusí pokladní zařízení čekat na odpověď ze systému Finanční správy. Vystavená účtenka pak místo fiskálního identifikačního kódu (`FIK`) bude obsahovat podpisový kód poplatníka (`PKP`). Údaje o tržbě musí následně podnikatel odeslat nejpozději do 48 hodin od jejího uskutečnění (pokladní zařízení to zpravidla činí automaticky).

[3] Fiskální identifikační kód.
[4] Aplikace dostupná na https://www.uctenkovka.cz.



Obrázek 3.2: Kroky procesu evidence tržby při výpadku spojení [11]

1. Podnikatel zašle datovou zprávu o transakci ve formátu `XML` Finanční správě, ale zaslání se nezdaří.
2. Ze systému Finanční správy není zaslán unikátní kód `FIK`.
3. Podnikatel vystaví zákazníkovi účtenku s kódem `BKP` (bez `FIK`).
4. Zákazník obdrží účtenku, která není doposud evidovaná v systému Finanční správy.
5. Jakmile dojde k obnově spojení, pokladní zařízení odešle údaje o neúspěšně zaslané tržbě.
6. Finanční správa potvrdí přijetí datové zprávy zasláním unikátního kódu `FIK`.

3.2.3 Evidence tržeb ve zjednodušeném režimu

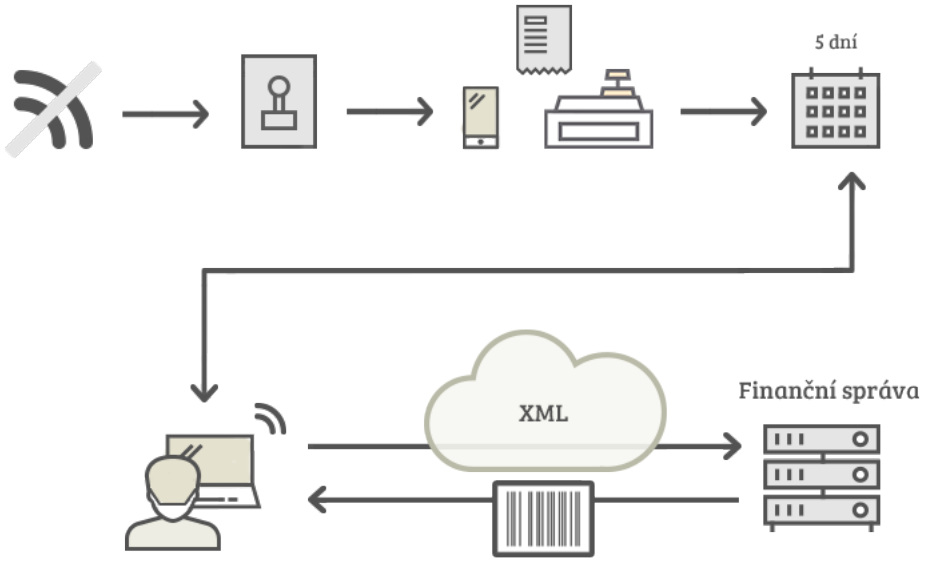
Pro některé podnikatele je nemožné evidovat tržby v online režimu například z důvodu absence internetové připojení. Z tohoto důvodu byl zaveden zjednodušený režim, který poplatníkovi ukládá odeslat informace o tržbě nejpozději do 5 dnů od uskutečnění evidované tržby. Na účtence tedy není povinností uvádět unikátní kód `FIK`, ale je nutné uvést podpisový kód poplatníka (`PKP`).

Tento režim není mezi podnikateli příliš rozšířen, jelikož ho může využívat pouze ten podnikatel, kterému bylo na jeho žádost vydáno povolení, nebo spadá do výjimek definovaných zákonem od `EET`[5].

1. Podnikatel prostřednictvím pokladního zařízení vystaví účtenku bez `FIK`, jež je následně uložena do paměti pokladního zařízení.
2. Zákazník obdrží účtenku s kódem `PKP` a `BKP`.

[5] Zákon č. 112/2016 Sb., o evidenci tržeb.

3. Do pěti dnů zajistí podnikatel odeslání datových zpráv o tržbách do systému Finanční správy (např. tak, že přemístí pokladní zařízení do místa, kde je dostupný internet).
4. Finanční správa potvrdí přijetí datové zprávy zasláním unikátního kódu `FIK` pro jednotlivé tržby.



Obrázek 3.3: Kroky procesu evidence tržby ve zjednodušeném režimu [11]

3.3 Technické řešení

Daňový portál pro evidenci tržeb je poskytován jako webová služba 2.4. Konkrétně využívá kombinaci protokolu standardních webových služeb (`SOAP1.1`) s využitím standardu `WS-Security`[6]. Ten rozšiřuje zprávu `SOAP` tak, aby poskytovala *kvalitní ochranu* zajištěním integrity, důvěrnosti a autenticity jedné zprávy [16].

3.3.1 Typy datových zpráv

Rozlišujeme následující typy datových zpráv:

Datová zpráva evidované tržby

`SOAP` tělo obsahuje všechny údaje, které jsou stanoveny technickou specifikací pro odeslání údajů o evidované tržbě. Samotná data evidované tržby jsou uložena ve vnořené struktuře *e-tržby*. `SOAP` hlavička obsahuje `XML` podpis a certifikát poplatníka, jehož privátní klíč byl použit k vytvoření `XML` podpisu. Viz obrázek 3.4 vlevo.

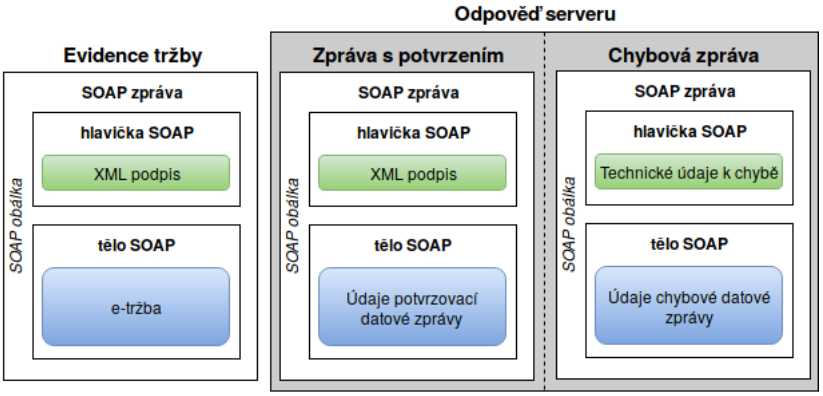
Potvrzovací datová zpráva

`SOAP` tělo obsahuje potvrzovací údaje o přijetí evidované tržby – hlavně potřebný `FIK`. `SOAP` hlavička obsahuje `XML` podpis a certifikát správce daně, jehož privátní klíč byl použit k vytvoření `XML` podpisu. Viz obrázek 3.4 uprostřed.

[6]Specifikace dostupná online: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

Chybová datová zpráva

Může nastat v případě kritické chyby v přijaté datové zprávě evidované tržby nebo z důvodu dočasné technické chyby na straně správce daně. `SOAP` tělo obsahuje chybový kód a textový popis chyby. Tato zpráva není na rozdíl od předchozích podepsána. Viz obrázek 3.4 vpravo.



Obrázek 3.4: Typy datových zpráv [13]

Kapitola 4

Analýza požadavků

Pokladna má za cíl snížit podnikateli náklady na pořízení potřebného hardwaru a umožnit používání bez ohledu na platformu, a to tak, že celé řešení bude postaveno na webových technologiích. K provozu tedy postačí běžný webhosting a internetový prohlížeč na straně klienta. Dalším záměrem je minimalizovat náklady na samotný provoz. V současnosti je velká kritika na zjevné dopady na životní prostředí z důvodu vyšší spotřeby papíru na „zbytečně“[1] vydané účtenky, které jsou navíc delší než dříve z důvodu nutnosti splnit náležitosti dané zákonem (po 40 dnech od zavedení `EET` vzrostla o 40 % spotřeba termopapíru [18]).

Pokladna rovněž nabídne funkce i pro samotné zákazníky s cílem ulehčit správu vlastněných účtenek, a dále nabídne tiketovací systém podpory pro řešení problémů se zakoupeným zbožím, zadání objednávky apod. Celý systém si lze představit jako internetový obchod s tím rozdílem, že obsah nákupu netvoří zákazník, ale prodejce (pokladní) na základě požadavků zákazníka. Zákazník má pouze přístup k realizovaným objednávkám (nákupům) skrz portál pro zákazníky.

Požadavky na vývoj vycházejí ze současné verze pokladny a dále z nedostatků, které jsem zaznamenal u první verze a u konkurenčních řešení.

4.1 Pokladna

Původní verze pokladny byla vytvořena jako jednouživatelská aplikace, tedy pro jednoho pokladního. Umožňovala tak v jeden okamžik tvorbu pouze jednoho účtu a nebylo možné mít rozpracovaných více účtů současně. Dále veškeré nastavení pokladny (údaje o podnikateli, certifikáty pro `EET`, ...) bylo nutné realizovat při instalaci pokladny skrze konfigurační soubory. Pokud podnikatel chtěl změnit nějaké údaje, neobešel se tak bez zásahu vývojáře.

Vyvíjená pokladna bude již víceuživatelská, což umožní práci několika pokladních v jeden okamžik. Podnikatel si bude moci definovat několik pokladen, nicméně každá bude vedena pod tentýž subjektem (firmou). Podnikatel si například takto bude moci vytvořit pokladnu pro každého zaměstnance či pro každou směnu. Dále bude pokladna umožňovat mít rozpracovaných několik účtů – tzn. při tvorbě účtu může pokladní kdykoliv jeho tvorbu přerušit („zaparkovat“ účet) a vytvořit nový či otevřít již „zaparkovaný“ účet a pokračovat v něm.

[1] Obchodník pro splnění povinnosti dané zákonem vytiskne účtenku a předává ji zákazníkovi. Ten ji ale většinou nechce, a tak papír končí v koši.

4.1.1 Režimy DPH

Původní verze pokladny pracovala pouze s jednou sazbou DPH (základní 21 %) a v případě potřeby změny její hodnoty by byl nutný zásah do konfiguračního souboru.

Vyvíjená pokladna bude podporovat veškeré sazby DPH dané zákonem a v případě jejich změn půjde jejich hodnota upravit bez zásahu do zdrojového kódu či konfiguračních souborů. Pro neplátce DPH půjde pokladna přepnout do režimu, který tuto skutečnost zohlední jak v samotné aplikaci, tak i v údajích na dokladu.

4.1.2 Realizace prodeje

Původní verze pokladny umožňovala tvorbu účtu (přidávání položek) dvěma způsoby. Prvním z nich bylo manuální vložení položky, kdy si pokladní doplnil název produktu, cenu vč. DPH a počet kusů. Druhý způsob byl realizován pomocí katalogu zboží, kdy se pokladnímu v modálním okně zobrazil seznam zboží v adresářové hierarchii. Pokladní si mohl vybrat jeden konkrétní produkt, nebo pomocí zaškrtávacího pole vybrat několik produktů. Katalog zboží nabízel také vyhledávání pro rychlejší nalezení produktu či kategorie (složky). K jednotlivým položkám na účtu bylo dále možné doplnit poznámku (např. pro sériové číslo zboží).

Realizaci prodeje u vyvíjené pokladny bude možné provést více způsoby, kde každý z nich bude vhodný pro různé typy provozů. Dále bude umožněna práce se čtečkou čárových kódů, která podstatně urychlí markování. Při přidávání produktů bude prodejci k dispozici „našeptávač“ produktů, který na základě historie realizovaných nákupů nabídne položky, které by se na účtě mohly také vyskytnout. Například pokud většina realizovaných nákupů bude v případě výskytu položky *rohlík* obsahovat také položku *máslo*, tak v případě vložení *rohlíku* nabídne našeptávač produkt *máslo*. Tímto by mělo dojít k urychlení markování, jelikož prodejce nebude nucen produkt vyhledávat v katalogu.

Obsah účtenky bude pokladnímu v průběhu markování vizualizován na obrazovce. Pokladní zde bude mít možnost měnit počty položek, upravit položku (cena, název, sazba DPH), odstranit položku z účtu či popřípadě vložit vlastní (novou) položku. U každé položky se pro informaci zobrazí mezisoučet a cena za jednotku.

4.1.3 Výdej dokladu

Kromě běžného tisku dokladu pomocí tiskárny pokladna dále nabídne vydání elektronického dokladu. Jeho předání bude realizováno jedním z následujících způsobů:

- elektronickou poštou
- vyzvednutím dokladu na portálu pro zákazníky pomocí obdrženého unikátního kódu v případě neregistrovaného zákazníka
- zobrazením dokladu v historii dokladů v portálu pro zákazníky v případě registrovaného zákazníka a přiřazení dokladu pod jeho zákaznický účet.

4.1.4 Správa zboží

Zboží bude možné umísťovat do kategorií (složek) s neomezenou možností zanoření (adresářová hierarchie). Pro rychlé nalezení produktu či kategorie nabídne katalog možnost vyhledávání, které bude probíhat v aktivní složce směrem níže v adresářové hierarchii. Při

aktivním vyhledávání půjde změnit úrovně zanoření (tzn. vynořit se či zanořit se v hierarchii), čímž dojde i ke změně oblasti, v níž bude vyhledávání prováděno.

U kategorií bude možné kromě povinného názvu vybrat výchozí sazbu DPH a výchozí měrnou jednotku. Tímto při vložení produktu do dané kategorie dojde k předvyplnění výchozích hodnot. U samotných produktů budou povinnými údaji název, cena, sazba DPH (v případě plátce DPH), měrná jednotka a množství produktu ve zvolené měrné jednotce. Mezi nepovinné údaje bude patřit čárový kód, který půjde zadat automaticky pomocí čtečky čárových kódů či manuálně, hodnota marže (v procentech, nebo korunách), barva pro odlišení a poznámka.

4.2 Portál pro zákazníky

Pokladna se bude v této části odlišovat od konkurenčních řešení, které tuto možnost nenabízí. Jedná se o část aplikace určenou pro zákazníky, kde zákazník bude mít přístup ke svému zákaznickému účtu. Přístup bude umožněn na základě registrace ze strany zákazníka či přidělení přístupů od prodejce. Zákazník zde primárně nalezne historii vydaných dokladů a informaci o svém účtě.

Další funkcionalitou bude sekce podpory neboli tiketovací systém, který bude sloužit k jednodušší komunikaci mezi zákazníkem a prodejcem či pro interní záležitosti. Obě strany přehledně uvidí celou komunikaci na jednom místě bez nutnosti dohledávání komunikace v elektronické poště.

Dále bude k dispozici sekce *ke stažení*, v které může prodejce zákazníkovi poskytnout různé soubory, jako jsou návody, reklamační řád apod. Další část nabídne rychlé informace ve formě často kladných otázek (FAQ). Poslední sekcí budou *aktuality* pro informování zákazníků.

4.3 Skladová evidence

Skladová evidence bude volitelná součást, jelikož pro některé podnikatele je vedení skladu nepotřebné. Nabídne možnost naskladnění a zobrazení skladových pohybů a realizaci inventury.

Naskladnění

Při naskladnění položek bude mít uživatel možnost vyplnit dodavatele, od kterého zboží zakoupil. Položky týkající se naskladnění půjdou vybrat pomocí katalogu zboží či naskenováním čárového kódu. U každé položky uživatel vyplní nákupní cenu, prodejní cenu a množství. Průvodce bude rovněž umožňovat hlídání marže v případě požadavku na tuto kontrolu.

Skladové pohyby

Uživatel zde nalezne veškeré skladové pohyby, které se budou dělit na *prodej*, *naskladnění* a *inventura*. V detailu skladového pohybu budou obecné informace týkající se samotného pohybu a seznam položek. U každé položky bude uveden počet kusů a cena, za kterou byla položka nakoupena, či prodána.

4.4 Existující aplikace

V současnosti již na trhu existují řešení poskytované jako služba zdarma či placené verze, které jsou bezplatné za cenu omezení funkcionality. Nevýhodou těchto řešení je velká závislost na provozovateli, jelikož tyto služby mohou kdykoliv zaniknout či se stát zpoplatněnými. Další nevýhoda je ve způsobu uložení dat, která jsou ve většině případů uložena na straně provozovatele služby. Klient služby si takto nemůže být nikdy jistý, kde všude se jeho data objeví, zda jsou uložena bezpečně či zda je provozovatel neanalyzuje ve prospěch konkurence.

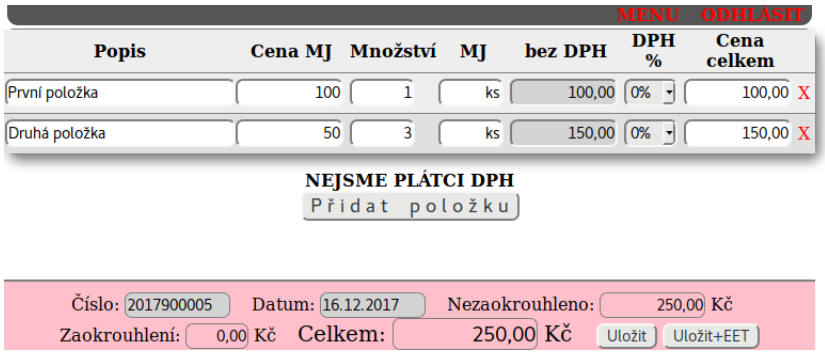
V následujících odstavcích bych zmínil pokladny, které jsou zdarma poskytovány jako služba pomocí webových technologií.

4.4.1 EET ZDARMA

Pokladna na návštěvníka nepůsobí příliš přívětivě z důvodu velmi jednoduchého a archaického uživatelského rozhraní. Po registraci je možné vytvořit několik firem (subjektů) pod jedním účtem. Pokladna nenabízí katalog zboží či vedení skladu. Tvorba účtu je pojata ve stylu vyplňování klasického tištěného paragonu (viz obrázek 4.1). Pokladní musí u každé položky vyplnit název, cenu, množství, měrnou jednotku a vybrat sazbu DPH. Což je značně zdlouhavé – celý proces vyplňování by například ulehčil našeptávač, který by pokladnímu nabízel názvy zboží z již existujících dokladů. Spodní část stránky slouží pro zobrazení informací o mezisoučtu a k dokončení tržby. Jako další funkcionalitu nabízí evidenci tržby z jiné aplikace pomocí `API` rozhraní. Tímto může podnikatel evidovat tržby např. z eshopu, který evidenci tržeb nepodporuje. Vydané či uložené doklady lze dohledat v historii dokladů, kde je pokladnímu k dispozici filtr pro hledání dokladu dle různých atributů.

Nedostatkem této služby je, že nelze mít současně otevřených několik účtů nebo účet „zaparkovat“ pro pozdější dokončení. Nabízí se sice možnost mít otevřené rozpracované doklady v různých oknech (panelech) prohlížeče, ale zde pak narazíme na problém s číslováním dokladu, jelikož každé okno bude mít předvyplněné totožné pořadové číslo – nutný manuální zásah, jinak uložení skončí s chybou týkající se duplicity.

Pokladna rovněž není dobře vyřešena po implementační stránce. Před zaevidováním tržby si může pokladní zvolit pořadové číslo paragonu. Před uložením tržby systém kontroluje, zda nedošlo k duplicitě a v případě duplicity dojde k chybě. Což je samozřejmě v pořádku, ale nedostatek spočívá v tom, že aplikace umožňuje smazání již vydaných paragonů. Takto se může jednoduše stát, že dojde k vydání paragonu s duplicitním pořadovým číslem, což odporuje zákonu o evidenci tržeb.



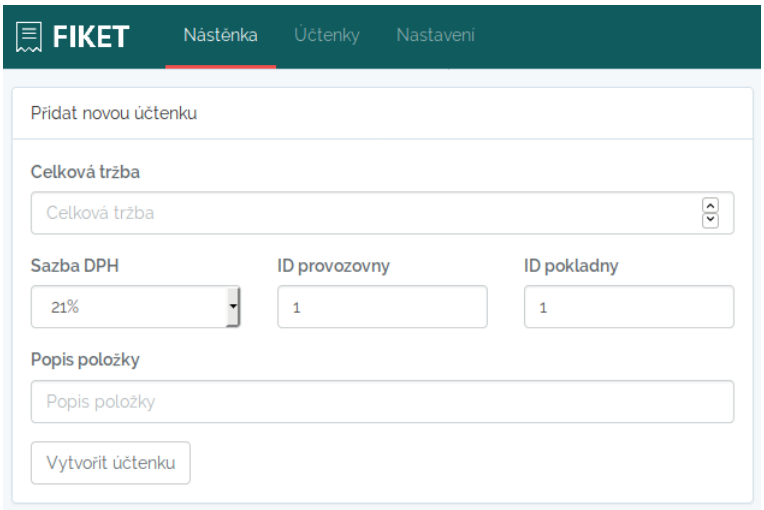
Obrázek 4.1: Tvorba účtu v aplikaci *EET ZDARMA*, zdroj: www.zdarma-eet.cz

4.4.2 Fiket

Pokladna nabízí velmi jednoduché a přívětivé uživatelské rozhraní. Pokladna podporuje nastavení pouze jedné firmy (subjektu). Počet pokladen a provozen je neomezen, jelikož při vystavení dokladu lze vyplnit vlastní název pokladny a název provozovny (nicméně je zde riziko chybného vyplnění).

Tvorba dokladu je pojata velmi minimalistickým způsobem, kdy pokladnímu postačí vyplnit pouze hodnotu celkové tržby, sazbu DPH a volitelně popis tržby (viz obrázek 4.2). Po uložení je možné předat doklad zákazníkovi v elektronické formě čímž celé řešení vyniká. Předání dokladu v elektronické formě spočívá v unikátním pětimístném kódu, jenž je každému dokladu přiřazen. Pokladní tedy předá zákazníkovi pouze kód účtenky, kdy po jehož zadání na stránce aplikace *Fiket* dojde k zpřístupnění dokladu. Pokladna rovněž nabízí přehledné statistiky, ve kterých poskytuje informace o počtu vydaných účtenek a sumě tržeb.

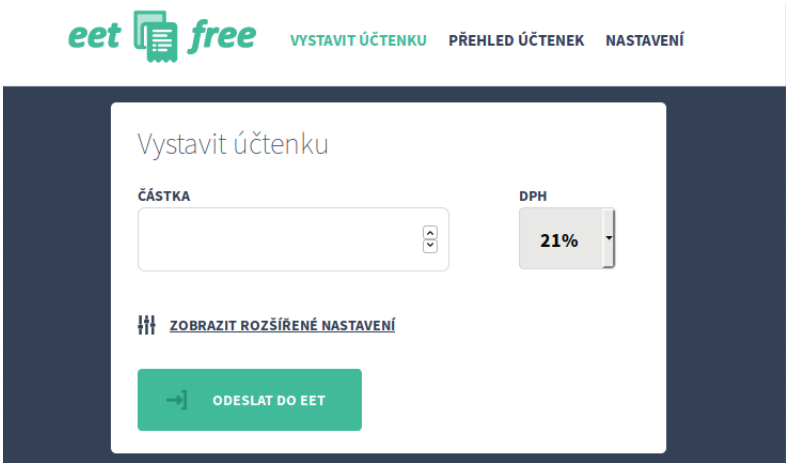
Celé řešení vyniká svou jednoduchostí. Avšak tento minimalistický přístup přináší i své nevýhody. První nedostatek spočívá v tom, že při tvorbě dokladu nelze tržbu rozdělit do několika sazeb DPH. V případě markování položek s různou sazbou DPH je nutné vystavit několik dokladů.



Obrázek 4.2: Tvorba účtu v aplikaci *FIKET*, zdroj: www.fiket.cz

4.4.3 EET free

Pokladna se velmi podobá svou jednoduchostí a nevýhodami pokladně *Fiket* 4.4.2. Pro vystavení dokladu je nutné vyplnit pouze hodnotu celkové tržby a sazbu DPH. V případě zobrazení rozšířeného nastavení lze dále vybrat druh platby a uvést poznámku (viz obrázek 4.3). Po uložení dokladu je možné jej zákazníkovi předat v tištěné podobě či elektronicky. Pro tištěnou podobu pokladna nabízí účtenku v několika rozměrech, a to pro tisk na klasický A4 papír či papír do pokladní tiskárny (80mm, 58mm). Elektronické předání účtenky je realizováno pomocí elektronické pošty, kdy dojde účtenka ve formě `HTML` emailové zprávy.



Obrázek 4.3: Tvorba účtu v aplikaci *EET free*, zdroj: www.eetfree.cz

4.4.4 Společné nedostatky

Mezi společné nedostatky výše zmíněných pokladen patří nekomfortní sekce s historií vydaných dokladů. Nelze jednoduše vyhledat doklad s daným identifikátorem či doklady filtrovat dle primárních atributů. Výjimkou je aplikace *EET ZDARMA*, která sice filtrování nabízí, ale není příliš komfortní – pro vyhledání dokladu nestačí zadat pouze jeho identifikátor, ale je nutné zadat hodnotu identifikátoru intervalem (tedy místo jedné hodnoty vyplnit dvě).

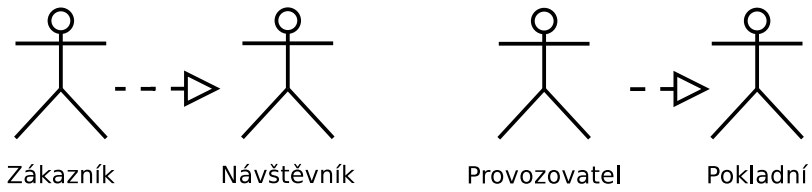
Kapitola 5

Specifikace požadavků

Specifikace požadavků vychází z analýzy uvedené v kapitole 4. V kapitole jsou nastíněny diagramy případů užití, jenž popisují případy užití softwarového systému z pohledu uživatele, definují typy aktérů (uživatelů) a posloupnosti událostí při užívání [32, s. 93].

Diagramy případů užití dále poslouží při návrhu, kde se pomocí nich eliminuje výskyt případů, kdy by se opomenulo na podstatný funkční požadavek, a v neposlední řadě při plánování implementace pro odhad rozsahu jednotlivých části systému.

V celém systému vystupují celkem čtyři aktéři, jejichž hierarchie je znázorněna na obrázku 5.1.



Obrázek 5.1: Role uživatelů systému a jejich hierarchie

5.1 Portál pro zákazníky

Portál pro zákazníky rozlišuje dva typy uživatelských rolí, a to *návštěvníka* a *zákazníka*.

5.1.1 Návštěvník

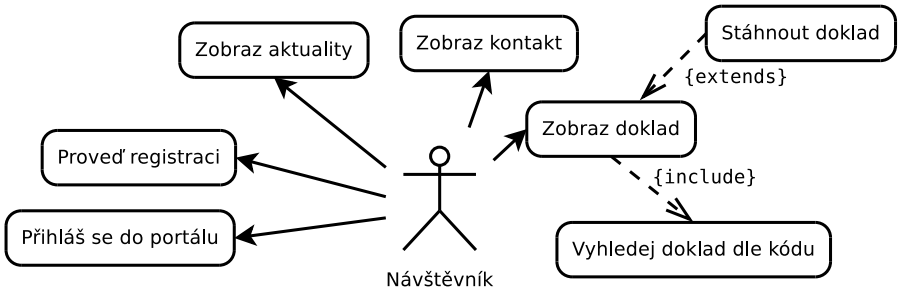
Návštěvníkem je myšlen uživatel, který není vůči portálu autentizován. Z toho důvodu jsou mu funkce portálu oproti *zákazníkovi* omezené. Primární funkcí portálu pro *návštěvníka* je zobrazení stránky s dokladem pomocí unikátní `URL` adresy. Na stránce se zobrazí základní informace o dokladu a seznam zakoupených položek. Doklad lze rovněž stáhnout ve formátu `PDF`.

Vytvoření zákaznického účtu proběhne na základě registrace. Registrační formulář se skládá ze tří částí, a to obecných informací (jméno, telefon, email), přihlašovacích údajů (uživatelské jméno a heslo) a kontaktních údajů (možné vyplnit automaticky na základě `IČ`[1]). Aby nebyl proces registrace příliš zdlouhavý, postačí pro dokončení registrace zadat

[1] Identifikační číslo osoby.

pouze jméno, email, uživatelské jméno a heslo. Volba uživatelského jména není formátem nikterak omezena – uživatel si může zadat jak přezdívku, tak i emailovou adresu. Po úspěšné registraci je nutná aktivace účtu pomocí odkazu uvedeného v zaslané emailové zprávě.

Další funkcionalitou je zobrazení výpisu aktualit, které podnikatel uveřejní skrze administraci, a dále zobrazení kontaktních údajů podnikatele. Výsledný diagram viz obrázek 5.2.



Obrázek 5.2: Diagram případu užití pro roli *návštěvník*

5.1.2 Zákazník

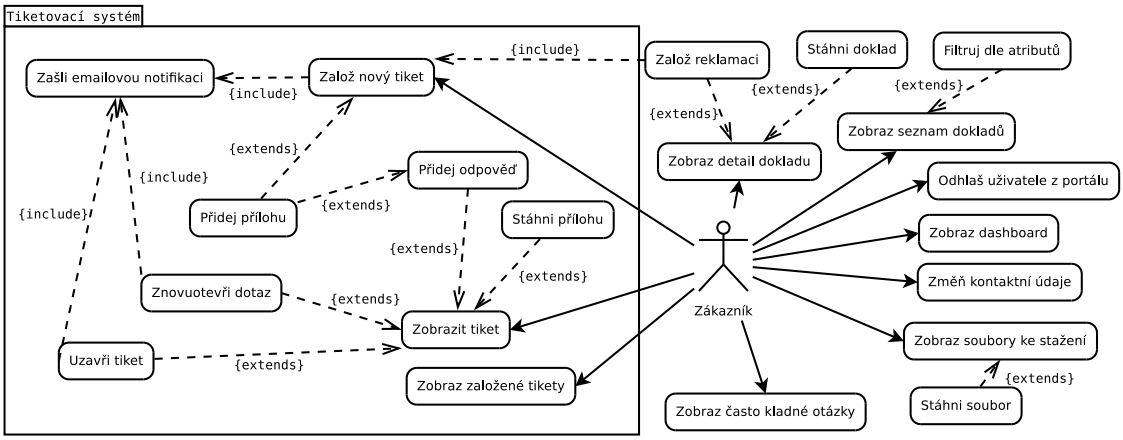
Zákazníkem je myšlen uživatel, který je vůči portálu autentizován. Takový uživatel je autorizován přistupovat k dalším funkcionalitám portálu. Dvěma hlavními funkcionalitami je zpřístupnění dokladů vydaných pro daný zákaznický účet a tiketovací systém podpory. Po přihlášení se *zákazníkovi* jako úvodní stránka zobrazí rychlý přehled, kde nalezne důležité informace o svém zákaznickém účtě, poslední vydané doklady a seznam nedořešených dotazů z tiketovacího systému.

Zákazník má přístup k historii vydaných dokladů. Výpis dokladů lze filtrovat a řadit dle zobrazených atributů. Detail dokladu obsahuje podrobný výpis položek, možnost jejího stažení a funkci oznámení problému, jež *zákazník* využije v případě, pokud by shledal s dokladem/zbožím nějaký problém (např. požadavek na reklamaci zboží). Oznámení problému je realizováno založením tiketu.

Založení tiketu lze realizovat i jiným způsobem, a to přímo v tiketovacím systému, který je znázorněn v levé části diagramu případu užití na obrázku 5.3. Pro založení tiketu je po *zákazníkovi* požadováno zadání předmětu a obsahu zprávy, výběr typu (reklamace, objednávka, . . .) a volitelně může být přiložena příloha.

Založený tiket se následně zobrazí v seznamu založených tiketů. V detailu tiketu jsou jednotlivé zprávy chronologicky zobrazeny v časové ose. Z důvodu lepší přehlednosti jsou zprávy seskupovány po dnech a barevně odlišeny dle jejich autorů. *Zákazník* může kdykoliv tiket uzavřít (označit za vyřešený), nebo opětovně otevřít. V případě, že nedojde k vyřešení tiketu v systémově definovaném časovém horizontu, je tiket automaticky uzavřen. Při jakékoliv změně jsou zainteresování uživatelé v rámci tiketu notifikování emailovou zprávou.

Pro zvýšení podpory zákazníku dále nechybí sekce často kladných otázek a sekce se soubory ke stažení. Obsah těchto sekcí je tvořen v administrační části. Poslední funkcionalitou je sekce nastavení, která slouží pro změnu kontaktních a přihlašovacích údajů. Výsledný diagram viz obrázek 5.3.



Obrázek 5.3: Diagram případu užití pro roli *zákazník*

5.2 Pokladna

V části pokladny se vyskytují dva aktéři – *pokladní* a *provozovatel*. Oběma aktérům se nabízí shodné funkcionality, proto je dále návrh pokladny zaměřen pouze na roli *pokladního*. Z důvodu přehlednosti uživatelského rozhraní nabízí pokladna pouze funkcionality, které jsou potřebné k realizaci prodeje. Vše ostatní, jako např. sklad, se nachází v administraci 5.3.

Pokladní má přístup jen k určitým pokladnám na základě přidělených oprávnění *provozovatelem*. Volba pokladny je umožněna v navigaci. V rámci zvolené pokladny může být otevřeno několik účtů, přičemž každý pokladní může mít aktivní pouze jeden účet (zbylé jsou zaparkované). V případě, kdy má *pokladní* rozpracovaný účet a chce založit nový, dojde k „zaparkování“ otevřeného účtu a založení nového. Otevření již *zaparkovaného* lze realizovat ze seznamu zaparkovaných účtů, který je v rámci dané instance pokladny. Nelze se tedy dostat k zaparkovaným účtům jiných pokladen. Pokud je daný účet již otevřen, jeví se ostatním pokladním jako uzamčený a lze jej otevřít pouze pro náhled. V případě, že není účet uzamčen, lze jej otevřít či smazat. Pro lepší přehlednost je možné účty pojmenovat.

Tvorba účtu probíhá pomocí dvou komponent umístěných v jednom okně. První komponenta slouží k vizualizaci účtu. U každé položky na účtě je vypsán její název, množství, mezisoučet a jednotková cena. Hodnotu množství lze operativně změnit pomocí tlačítek plus a mínus. Dále je možné položku odstranit z účtu nebo ji upravit. Úprava probíhá v modálním okně, kde *prodejce* může změnit název, množství, cenu, sazbu `DPH` a doplňující poznámku (např. pro sériové číslo produktu). Tyto úpravy mají vliv pouze na položky na účtě, nikoliv na produkt v katalogu. Přidání vlastní položky je realizováno pomocí modálního okna, kde uživatel vyplní údaje o položce (název, množství, cena, sazba DPH a poznámka). Aby se urychlil proces přidání vlastní položky, má *pokladní* možnost přidat „rychlou položku“. Zde je oproti předchozímu případu požadováno zadání ceny, množství a sazby DPH. V zápatí komponenty se dále nachází sumarizační údaje, kde je uvedena celková hodnota daně, počet položek a celková cena. Po jakékoliv změně dojde k automatickému překreslení vizualizovaného účtu.

Druhá komponenta slouží pro vkládání položek na účet. K dispozici je v následujících dvou variantách:

- **Režim katalogu**
  Režim nabízí procházení katalogu zboží ve formě adresářové struktury. V katalogu jsou barevně odlišené kategorie (složky) a produkty, jejichž barvu lze zvolit. U každé položky jsou zobrazeny informace o aktuálním stavu skladu a ceně za jednotku. Úroveň zanoření v hierarchii složek je uživateli znázorněna pomocí drobečkové navigace. Pod výpisem produktů se nachází našeptávač, který na základě vložených produktů na účtě nabídne při markování produkty, které by se mohly na účtě také vyskytnout.

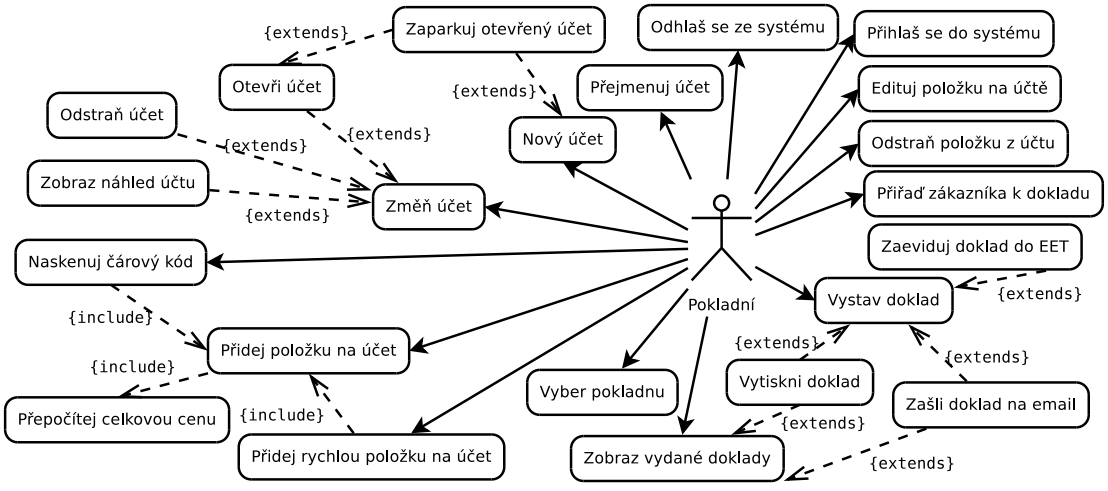
- **Režim okamžitého prodeje**
  Režim vhodný pro provozy bez zavedených položek v katalogu. *Pokladnímu* se zobrazí formulář pro přidávání vlastních položek.

Pro markování rovněž nechybí podpora čtečky čárových (`EAN`) kódů. Po naskenování kódu pokladna sama rozpozná tento úkon a v případě úspěšného nalezení položku přidá na účet.

Proces tvorby účtu je ukončen vystavením dokladu, kdy pokladna nabídne následující způsoby pro předání dokladu zákazníkovi:

- zobrazení `URL` adresy s unikátním kódem na doklad
- vytisknutí kompletního dokladu
- přiřazení dokladu k zákaznickému účtu
- vyplnění emailové adresy zákazníka pro zaslání dokladu na email.

Pokladní si dále může zobrazit vydané doklady v rámci pokladny a v případě potřeby zpětně doplnit zákazníka, opětovně vytisknout doklad či provést storno. Výsledný diagram viz obrázek 5.4.



Obrázek 5.4: Diagram případu užití pro roli *pokladní*

5.3 Administrace

Administrace slouží ke správě pokladny a portálu pro zákazníky. Autentizace do této části probíhá zcela nezávisle na portálu pro zákazníky a k přístupu je autorizována role *pokladního*

a *provozovatele*. Z důvodu rozsáhlosti této části aplikace je popis následně zaměřen pouze na některé případy užití.

5.3.1 Katalog zboží

Katalog zboží je koncipován jako hierarchická adresářová struktura, pomocí níž lze produkty v katalogu třídit do kategorií. Jeho vizualizace je realizována pomocí tabulky, ve které jsou prvně vypsány kategorie a následně produkty. Řádky tabulky jsou pro lepší orientaci uživatele barevně odlišeny štítky, jejichž barva může být zvolena u samotné kategorie/produktu. Pohyb v úrovních katalogů zboží je realizován zvolením kategorie, kdy se uživatel dostane v hierarchii níže. Pro vynoření z dané úrovně hierarchie může využít prvního řádku v katalogu, který tuto možnost signalizuje. Pro přehled uživatele o kompletní cestě zanoření je k dispozici drobečková navigace, v níž jsou vypsány názvy všech nadřazených kategorií. Katalog nabízí funkci vyhledávání, kdy v případě její aktivace dojde k vyhledávání kategorií a produktů pouze v úrovni hierarchie katalogu, kde se uživatel právě nachází. Pokud by uživatel chtěl vyhledat produkt dle `EAN` kódu, může tak učinit pouhým načtením kódu. Katalog se sám postará o vyhledání a v případě nalezení produktu přímo otevře jeho editaci. Jestliže by bylo v katalogu více produktů se shodným `EAN` kódem, dojde pouze k vyhledávání a uživateli se vypíší nalezené produkty s touto duplicitou.

Vytvoření nové kategorie či produktu (dále jen položka) může být realizováno kdykoliv při procházení katalogu. Položka je přidána do kategorie, která je v katalogu právě otevřena. V případě že není otevřena žádná kategorie, je položka přidána do kořene katalogu (nejvyšší úrovně). Uživatel je při vytváření položky vždy informován, kde bude položka umístěna.

Přidání nové kategorie se realizuje skrze modální okno, jehož formulář obsahuje pouze jedno povinné pole, kterým je název. Dále může uživatel vybrat výchozí sazbu `DPH` a měrnou jednotku. Tyto hodnoty jsou následně použity při vytváření nových položek pod danou kategorií. Pro volbu barvy štítku se zde dále nachází paleta barev, pomocí níž si uživatel vybere barvu dle svého uvážení.

Vytváření produktu vyžaduje po uživateli podstatně více informací. Mezi povinná pole patří název, cena, sazba `DPH` a měrná jednotka. Pro cenu jsou k dispozici dvě políčka, a to pro cenu s `DPH` a bez `DPH`. Systém automaticky při změně jedné hodnoty dopočítá druhou. V případě, že podnikatel není plátce `DPH`, je pole pro výběr sazby `DPH` skryté a zadání ceny probíhá pouze jedním polem. Sazba `DPH` a měrná jednotka se vybírá z předem definovaných hodnot, které je možné definovat ve správě číselníků. Mezi volitelná pole patří např. `EAN` kód, který může být automaticky zadán pomocí čtečky čárových kódů.

5.3.2 Správa skladu

Správa skladu se skládá z části pro naskladnění zboží a s historií skladových pohybů. Výběr položek pro naskladnění probíhá probíhá pomocí katalogu, kde si uživatel zaškrtává produkty nebo načtením `EAN` kódu produktu. Vybrané produkty se ihned zobrazí v oknu s položkami pro naskladnění. U každé položky je požadováno vyplnění nákupní ceny, ceny za jednotku a počet jednotek. Pro informaci se u každé položky zobrazuje stará pořizovací cena, aktuální prodejní cena a aktuální stav na skladě. V případě, že je u položky zapnuté hlídání marže a hodnota je nižší než požadovaná hodnota, je na tuto skutečnost uživatel upozorněn a naskladnění se provede až po zadání validních hodnot. Dále může uživatel uvést informace týkající se naskladnění, a to vybrat dodavatele, vyplnit číslo dokladu a uvést poznámku.

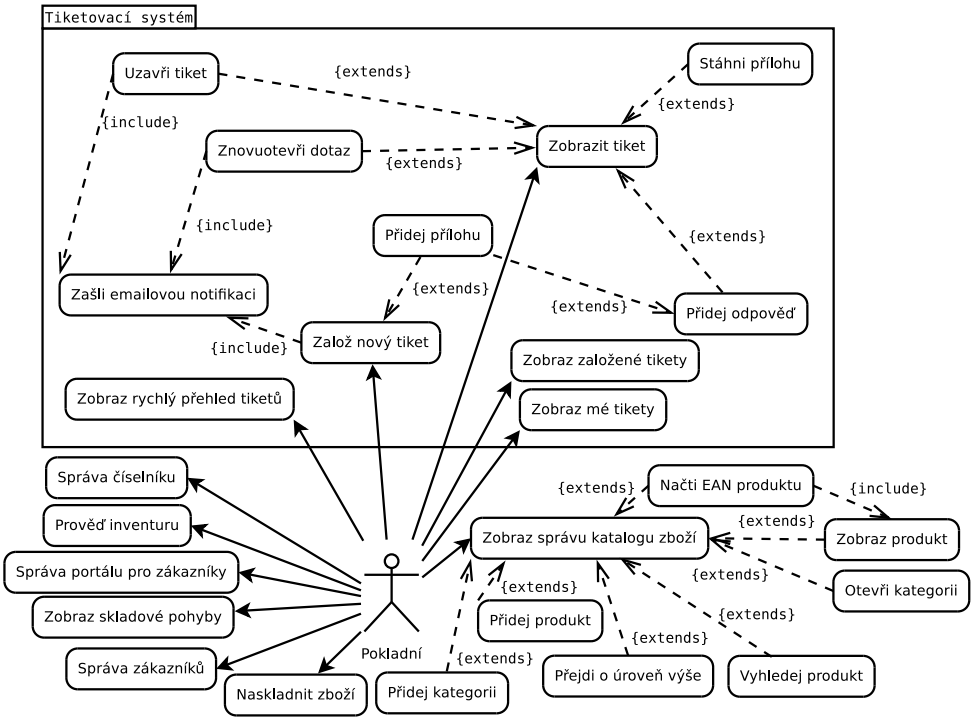
Historie skladových pohybů obsahuje veškeré pohyby zboží. U každého pohybu je uveden typ (nákup, prodej, inventura), kdo pohyb vykonal, počet položek a celková cena. Po zobrazení detailu pohybu lze zjistit položky, kterých se pohyb týká.

5.3.3 Tiketovací systém

Sekce správy tiketovacího systému plní téměř shodnou funkčnost jako sekce, jenž je přístupná *zákazníkovi* v portálu pro zákazníky. *Pokladní* zde nalezne jak tikety založené od zákazníků, tak i tikety založené pro interní komunikaci v rámci firmy. Pro lepší přehlednost systém nabízí možnost zobrazení jen těch tiketů, v nichž je přihlášený uživatel zainteresován (tzn. v komunikaci se nachází jím napsaná zpráva).

5.3.4 Správa číselníků

Správa číselníku obsahuje evidenci dodavatelů, sazeb `DPH` a měrných jednotek.



Obrázek 5.5: Diagram případu užití pro roli *pokladní* v administrační části

5.3.5 Provozovatel

Provozovatel má navíc ty funkcionality, které nemůže provádět neznalá osoba, jelikož by mohla špatným nastavením způsobit nefunkčnost pokladny, např. samotnou evidenci tržeb. Viz diagram na obrázku 5.6.

Nastavení systému

Podstatná část, ve které se nastavuje komunikace s daňovým portálem. Uživatel zde nahraje certifikát obdržený od daňového portálu a popřípadě zadá heslo k certifikátu, pokud bylo

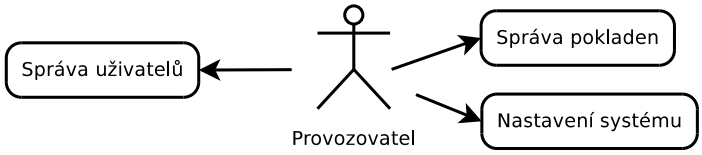
certifikátu nastaveno. Další nastavení se týká firemních údajů (název firmy, IČ, DIČ apod.). Aby si by uživatel jist, že je vše nastaveno v pořádku a komunikace s daňovým portálem funguje korektně, je zde k dispozici nástroj, který provede test komunikace a informuje uživatele o výsledku.

Správa pokladen

Provozovatel vytváří instance pokladen, které jsou na základě oprávnění zpřístupněny *pokladním*.

Správa uživatelů

Slouží pouze ke správě uživatelů pokladny a administrace. *Provozovatel* zde může vytvořit další uživatele s danou rolí. V systému musí vždy existovat nejméně jeden uživatel s rolí *provozovatele*.



Obrázek 5.6: Diagram případu užití pro roli *provozovatel* v administrační části

Kapitola 6

Technologie pro vývoj webových aplikací

Webové aplikace stoupají rychlým tempem na popularitě a stále větší škála desktopových aplikací se stěhuje do webového prostředí. Tímto jsou kladeny čím dál větší nároky na vývojové technologie. Existuje řada programovacích a skriptovacích jazyků pro vývoj webových aplikací. V tabulce 6.1 je uveden přehled nejpoužívanějších programovacích jazyků na straně serveru[1].

Jazyk	Využití [%]
PHP	83.1
ASP.NET	14.2
Java	2.5
ColdFusion	0.6
Ruby	0.6
JavaScript	0.4
Perl	0.3
Python	0.2
Erlang	0.1

(stav k 19. prosinci 2017)

Tabulka 6.1: Procentuální vyžití programovacích jazyků webovými aplikacemi, zdroj: http://w3techs.com/technologies/overview/programming_language/all

Jak je vidno, `PHP` se používá u 83.1 % webových aplikací na straně serveru. Může za to také fakt, že *PHP* je běžně dostupné u poskytovatelů webhostingu a jedná se o open-source řešení. Oproti tomu na klientské straně je nejpoužívanější *JavaScript* (94.9 %), jehož popularita dále poroste z důvodu zastavení vývoje *Adobe Flash* (5.6 %) [30].

6.1 Výběr vhodných technologií

Cílem vyvíjené pokladny je poskytnout ji jako otevřený software (open-source), kdy ji podnikatel bude moci provozovat na svém vlastním webovém serveru či běžně poskytovaném webhostingu. Z tohoto důvodu bylo žádoucí vybrat takové technologie, které jsou bezplatné

[1] Webová aplikace může využívat několik programovacích jazyků na straně serveru.

a běžně dostupné u poskytovatelů webhostingových služeb. Proto byly zvoleny pro serverovou část následující technologie: webový server *Apache*, skriptovací jazyk *PHP* a databázový server *MySQL* (trojice technologií známá jako `LAMP`[2]). Pro komunikaci s databází bude dále využit `ORM` framework *Doctrine* 6.4.1. Důvodem byly následující aspekty:

- technologie jsou běžně dostupné u poskytovatelů webhostingu
- open-source řešení, bezplatné
- multiplatformní
- nízké náklady na provoz
- silná komunita vývojářů – velmi malé riziko zastavení vývoje
- moderní technologie.

Pro klientskou část byl vybrán frontendový framework *Bootstrap* s využitím *JavaScriptové* knihovny *jQuery*. Zmíněné technologie budou dále popsány.

6.2 PHP

Skriptovací programovací jazyk běžící na straně serveru, určený především pro programování dynamických webových aplikací. Lze jej také použít k tvorbě konzolových a desktopových aplikací, u kterých je využita kompilovaná forma jazyka [31]. Jazyk je dynamicky typovaný, tzn. datový typ je vázán na hodnotu proměnné. Od verze `PHP 7` je možné deklarovat typy parametrů funkce/metody, a to i pro skalární typy. `PHP` podporuje objektový návrh pro složitější programové struktury, který v současnosti díky frameworkům převládá napříč webovými aplikacemi.

Jazyku `PHP` je také připisována řada nevýhod, kdy některé jsou již v současnosti eliminovány použitím dostupných nástrojů a novou verzí `PHP 7`. Dříve se jednalo hlavně o kritiku na volnost, kterou `PHP` dává programátorovi, díky čemuž je vývoj náchylný k chybám. V současnosti je chybovost eliminována možnosti zapnutí striktních typových kontrol. Další nevýhodou je horší týmová spolupráce, která může být nyní eliminována za použitím nástroje *Composer* 6.2.1. Některé nevýhody, jako např. neimplicitní ošetření vstupních hodnot od uživatele, lze eliminovat pouze na úrovni aplikace, k čemuž lze využít framework 6.2.2.

6.2.1 Composer

Jedná se o multiplatformní konzolový program (nástroj) pro správu knihoven a jiných zdrojů (tzv. závislostí, dependencies) v `PHP`. Programátor pro každý projekt manuálně či automatizovaně vytvoří soubor `composer.json`, ve kterém uvede informace o projektu (název, požadavky na `PHP` moduly, minimální verzi `PHP`, ...) a veškeré závislosti na knihovnách (název, požadovaná verze). Tento souboru si lze představit jako soubor *readme*, kterému je dodána formální specifikace.

Tímto mají vývojáři přehled o projektu a potřebných knihovnách. V případě nasazení projektu na nový server stačí spustit jeden příkaz, jenž automatizovaně zkontroluje veškeré požadavky na server a stáhne (aktualizuje) potřebné knihovny.

[2] Zkratka vytvořená z počátečních znaků použitých technologií – **L**iux, **A**pache, **M**ySQL a **P**HP.

6.2.2 Webový framework

Jedná se o aplikační framework, který je navržen pro vývoj webových aplikací. Framework programátorovi vytvoří striktnější prostředí a nutí ho tak používat určité konvence, což má za následek srozumitelný a snadno udržovatelný programový kód aplikace. Většinu frameworků lze díky jejich modulárnosti snadno rozšířit o další funkcionality pomocí knihoven. Programátor si doinstaluje pouze potřebné knihovny a nezatěžuje tak chod aplikace těmi nepotřebnými.

6.3 Nette framework

Český open-source framework, v počátku jehož vývoje stál *David Grudl*, a od roku 2009 je vývoj zaštítěn organizací *Nette Foundation*. V současnosti je šířen pod `BSD` licencí, která patří k těm nejvolnějším, jelikož umožňuje používání i v komerčních projektech. Je navržen čistě objektovým návrhem a pro svůj běh potřebuje nejméně verzi `PHP 5.3.1`. Nejaktuálnější verze `2.4` již využívá nových vlastností `PHP 7.1`. *Nette* se vyznačuje využitím populárního návrhového vzoru `MVP` 6.1. Jeho další vlastností je využívání událostmi řízeného programování a z velké části je založen na používání komponent, které představují základní kámen znovupoužitelnosti kódu. *Nette* klade mimořádný důraz na produktivitu, čistý kód a bezpečnost [21].

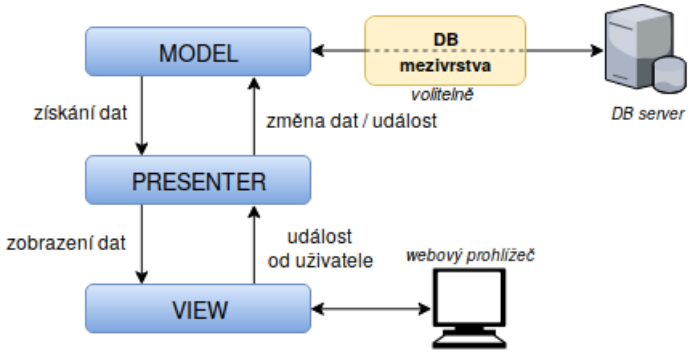
6.3.1 Důraz na zabezpečení

Při využití holého `PHP` je při vývoji velká šance zanesení bezpečnostní díry do aplikačního kódu. Velmi častými útoky na webové aplikace je využití neošetřeného vstupu dat od uživatele. K tomu lze využít techniky *SQL injection* pro napadení databázové vrstvy, kdy útočník pomocí neošetřeného vstupu docílí pozměnění `SQL` dotazu nebo techniky *Cross-Site Scripting* (`XSS`), kdy útočník do stránky podstrčí svůj kód. *Nette* proto klade velký důraz na bezpečnost, a tak veškerým vstupním datům od uživatele implicitně nedůvěřuje a před jejich zpřístupnění v aplikačním kódu je ošetří pomocí vestavěných mechanismů. Díky implicitnosti nemusí programátor řešit opakující se úkony vedoucí k ošetření dat a pouze na explicitní vyžádání může získat přístup k surovým datům. Ošetření vstupních dat lze také provádět *JavaScriptem* na straně uživatele pomocí validačních pravidel, která jsou automaticky generována u každého kontrolního prvku formuláře.

Nette podporuje i další mechanismy vůči různým útokům jako *Cross-Site Request Forgery*, *URL attack*, *control codes*, *invalid UTF-8*, *Session hijacking*, *session stealing*, *session fixation*...

6.3.2 Návrhový vzor MVP

Návrhový vzor odvození od `MVC` (**M**odel-**V**iew-**C**ontroller). Hojně používaný nejen pro tvorbu webových stránek, ale i desktopových aplikací. Vznikl z potřeby oddělit u aplikací s grafickým rozhraním kód obsluhy (*controller*/*presenter*) od kódu aplikační logiky (*model*) a od kódu definujícího grafické rozhraní (*view*).



Obrázek 6.1: Schéma návrhového vzoru `MVP`

Model

Model tvoří datový a zejména funkční základ celé aplikace. Je nezávislý na reprezentaci dat. Model si udržuje svůj vnitřní stav a vně nabízí pevně dané rozhraní, pomocí něhož můžeme zjišťovat či měnit jeho stav.

View

Převádí data získaná z modelu do podoby vhodné k interaktivní prezentaci uživateli [5]. V případě *Nette* se jedná o šablonu využívající šablonovacího systému *Latte*.

Presenter (controller)

Hlavní část, která řídí a spojuje dvě předcházející. Jejím primárním účelem je získat data z modelu a předat je šabloně, reagovat na události (typicky pocházející od uživatele), provádět změny v modelu (skrze jeho rozhraní) nebo pohledu [5].

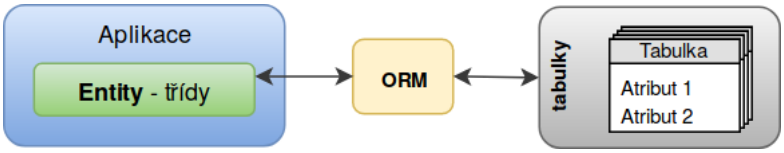
6.4 MySQL

MySQL[3] je systém řízení báze dat uplatňující relační databázový model [31]. Jedná se multiplatformní databázový systém využívající dotazovacího jazyka `SQL`[4]. Pro svou snadnou použitelnost, vysoký důraz na rychlost a výkon a především díky tomu, že se jedná o volně šiřitelný software, má v současné době vysoký podíl na používaných databázích – dle serveru *DB-Engines* je druhým nejpoužívanějším [4]. Vyšší rychlost je docílena za cenu některých zjednodušení, jako je např. jednoduchý způsob zálohování. Rovněž do páté verze *MySQL* nebyly podporovány pohledy, triggery a uložené procedury. Tyto funkce byly doplněny v nových verzích na popud vývojářů, kterým při vývoji scházely. Pro snadnou správu databází lze využít program *MySQL Workbench* či webovou aplikaci *phpMyAdmin*, která je hojně nabízena poskytovateli webhostingů [6].

[3] My Structured Query Language.
[4] Structured Query Language.

6.4.1 Doctrine

Jedná se o ORM[5] framework pro jazyk PHP, který tvoří mezivrstvu při komunikaci s databází. Mezivrstva zajišťuje mapování objektů (entit) na relační databázi, díky čemuž jsou záznamy v tabulce chápány jako objekty (viz obrázek 6.2). Návrh databáze je založen na moderním přístupu *Code First*. To znamená, že nejprve se vytvoří kód, konkrétně třídy představující entity, kde každá vlastnost (property) třídy obsahuje speciální anotaci. Na základě těchto entit dojde k vytvoření databázového schématu. Správa databázových objektů je postavena na návrhovém vzoru *Data Mapper*, která probíhá skrze fasádu[6] v podobě *Entity Manageru* [7].



Obrázek 6.2: Schéma ORM

6.5 Ostatní

Mezi další použité technologie patří balíčkovací systém *Bower*, který plní stejnou funkci jako *Composer* s tím rozdílem, že je určen ke správě knihoven, frontendových frameworků, grafických ikon apod. Dále JavaScriptový nástroj *Grunt.js*, jenž umožňuje automatizaci opakujících se úkonů a dále pomocí pluginů nabízí různé funkcionality, jako např. minifikace CSS či *JavaScriptových* souborů, spojení několika CSS souborů...

6.5.1 Bootstrap

Frontendový framework poskytující sadu nástrojů pro tvorbu moderních webových aplikací. Jedná se o návrhářské šablony založené na HTML a CSS sloužící pro úpravu typografie, formulářů, tlačítek, navigace a dalších komponent rozhraní, stejně jako další volitelná rozšíření *JavaScriptu* [2]. *Bootstrap* je kompatibilní se všemi verzemi hlavních prohlížečů a rovněž se přizpůsobuje použití na starších prohlížečích. Od verze 2.0 nabízí možnost responzivního designu, pomocí něhož dynamicky přizpůsobuje rozložení stránky s ohledem na používané zařízení (stolní PC, tablet, mobilní telefon) [2, 27].

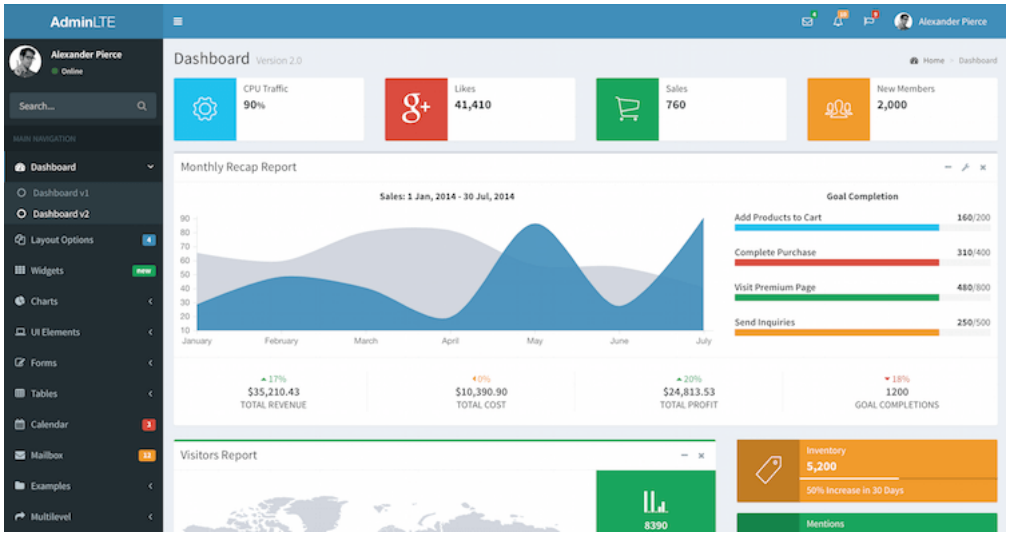
AdminLTE

Při vývoji bude využita šablona *AdminLTE*, která tvoří nadstavbu nad *Bootstrapem*. Šablonu lze různě přizpůsobit – podporuje několik způsobů rozložení stránky. Oproti holému *Bootstrapu* nabízí daleko více komponent.

6.5.2 jQuery

JavaScriptová knihovna běžící na straně prohlížeče, která slouží k manipulací s DOM strukturou. Oproti použití čistého JavaScriptu podstatně zjednoduší práci, zpřehlední zdrojový

[5] Object-Relational Mapping.
[6] Návrhový vzor, anglicky Facade.



Obrázek 6.3: Šablona *AdminLTE*, zdroj: https://adminlte.io

kód a abstrahuje od typu prohlížeče z hlediska vývoje. Kormě toho umožňuje další funkcionality jako zachytit na prvek v DOM struktuře událost a na základě jejího vyvolání reagovat, používat animace apod. [12]

Kapitola 7

Návrh aplikace

Tato kapitola nejprve seznámí čtenáře s částmi pokladního systému a jejich účelem. Dále bude zmíněn návrh modelové vrstvy a našeptávače produktů založeného na asociačních pravidlech. Závěr kapitoly se zaměřuje na rozložení uživatelského rozhraní jednotlivých částí.

7.1 Části pokladního systému

Na základě specifikace požadavků, uvedené v kapitole 5, byla aplikace dekomponována na tři části, a to:

- **Portál pro zákazníky**
  Sekce určená k zpřístupnění zákaznického účtu registrovaným zákazníkům a předání dokladu běžným návštěvníkům.
- **Pokladna**
  Obsahuje veškeré funkcionality pro realizaci prodeje. Primárně s ní budou pracovat pokladní.
- **Administrace**
  Slouží pro podporu a správu předchozích dvou části. Nachází se zde např. správa katalogu zboží a evidence skladu.

Dekompozice byla provedena z toho důvodu, že každá část je specifická pro odlišné aktéry systému a sdružuje funkcionality pro potřebu daných aktérů. Účel a návrh jednotlivých částí bude následně popsán.

7.2 Modelová vrstva

Návrh modelové vrstvy byl proveden technikou *Code First* s využitím `ORM` frameworku *Doctrine*, při které fyzické databázové schéma vzniká psaním aplikačního kódu. Modelová vrstva se skládá z tzv. entit, které jsou umístěny v adresáři `app/model/entities`. Entity jsou klasické objekty, kde každý atribut, který má být ukládán do databáze, musí být anotován v souladu s frameworkem *Doctrine*. Pomocí anotace definujeme, o jaký datový typ atributu se jedná apod. Na základě entit a uvedených anotací lze následně v databázi provést vygenerování nebo aktualizaci databázového schématu. Logické schéma databáze je uvedeno v příloze na obrázku A.3. Celkově je v aplikaci definováno následujících 27 entit.

Globální

- **User**
  Reprezentuje uživatele systému. Na základě role je uživatel dělen na zákazníka/pokladního/provozovatele.

- **SettingsGlobal**
  Určena k uložení globálního nastavení, mezi něž patří obsah stránek v portálu pro zákazníky a povolení funkcionality skladu.

- **SettingsEet**
  Kompletní nastavení funkcionality `EET` a její případné povolení/zakázání.

Pokladna

- **Receipt**
  Reprezentuje doklad vydaný pokladnou. Sama o sobě entita obsahuje pouze základní informace o dokladu – vše ostatní je uloženo pomocí vazeb na jiné entity. Doklad může být vázán na konkrétního zákazníka (entita `User`). U dokladu se eviduje, z jaké pokladny byl vystaven (entita `Cash`) a jakým uživatelem (pokladním). K dokladu lze nastavit fakturační údaje (entita `ReceiptCustomerData`) nezávisle od přiřazeného zákazníka. V případě zapnuté evidence dokladu do `EET` je na doklad vázána entita `ReceiptEetData`. Doklad je pevně svázán s nastavením (entita `SellerData`), jež bylo platné v době vystavení dokladu. Položky dokladu tvoří kolekce entit `ReceiptItem`. Viz diagram na obrázku A.1.

- **ReceiptItem**
  Reprezentuje položku na dokladu. Položka obsahuje název, počet jednotek, jednotkovou cenu, měrnou jednotku a poznámku. V případě, že prodej proběhl v režimu plátce `DPH`, je cena uvedena včetně `DPH` a dále je vyplněn údaj o hodnotě a typu sazby daně. Pokud byla položka vytvořena výběrem produktu z katalogu zboží, je tento produkt rovněž svázán s položkou.

- **ReceiptEetData**
  Entita obsahuje informace týkající se `EET` konkrétního dokladu, jako je stav zaevidování, čítač pokusů o zaevidování, `FIK` kód, označení pokladny a označení provozovny. Poslední dva údaje týkající se pokladny a provozovny by mohly být získány přímo z entity `Cash`, reprezentující pokladnu. Nicméně hodnoty v entitě `Cash` jsou variabilní a my potřebuje získat hodnoty, jež byly nastaveny v době vydání dokladu (hodnoty jsou potřebné pro budoucí vygenerování `PKP` a `BKP` kódu).

- **ReceiptCustomerData**
  Kontaktní údaje na zákazníka (odběratele).

- **ReceiptLogRecord**
  Log záznam týkající se pokusu o evidenci dokladu do `EET`.

- **SellerData**
  Kontaktní údaje provozovatele. Entity v systému tvoří jakýsi zásobník, kde na vrcholu je entita s nejaktuálnějšími údaji. Zbylé entity (s již neaktuálními údaji) jsou uchovávány z důvodu vazby na dříve vydané doklady.

- **DocumentIdSequence**
  Úložiště sekvencí identifikátoru pro jednotlivé typy dokladů.

- **Product**
  Produkt v katalogu zboží. Obsahuje základní údaje produktu, jako např. název, `EAN` kód, cena (bez `DPH`) atd. Dále produkt obsahuje vazbu na kategorii, v níž je přiřazen (entita `ProductCategory`); sazbu `DPH`, do které náleží (`TaxRate`) a měrnou jednotku (entita `MeasurementUnit`). Produkt nelze smazat v případě existence vazby s jinou entitou – v takovém případě je pouze označen jako archivovaný. Viz diagram na obrázku A.2.

- **ProductCategory**
  Entita představuje kategorii z katalogu zboží – tzn. seskupuje produkty do kategorie a tvoří hierarchickou strukturu katalogu zboží. U kategorie lze nastavit výchozí sazbu `DPH` a výchozí měrnou jednotku, která bude předvyplněna při vytváření produktu v dané kategorii.

- **Cash**
  Reprezentuje virtuální pokladnu v rámci pokladního systému. K pokladnám jsou pomocí vazby `cash_privileges` přiřazená oprávnění jednotlivých uživatelů. Pokladnu nelze smazat v případě existence vazby s jinou entitou – v takovém případě je pouze označena jako archivovaná.

- **MeasurementUnit**
  Entita definuje měrnou jednotku, prodejní jednotku a případně konverzní koeficient mezi nimi. Měrnou jednotku nelze smazat v případě existence vazby s jinou entitou – v takovém případě je pouze označena jako archivovaná.

- **TaxRate**
  Reprezentuje sazbu `DPH`. Mezi hlavní údaje patří název, typ sazby a hodnota. Sazbu nelze smazat v případě existence vazby s jinou entitou – v takovém případě je pouze označena jako archivovaná.

- **AprioriRule**
  Entita slouží pro uložení pravidla vygenerovaného pomocí algoritmu *Apriori* využitého v našeptávači produktů.

Sklad

- **StockMovement**
  Reprezentuje skladový pohyb, který obsahuje informace o pohybu (číslo dokladu, časové razítko provedení, vykonavatele, ...) a kolekci entit `StockMovementItem`. Skladový pohyb se dle typu dělí na *nákup zboží*, *prodej zboží* a *inventuru*. V případě typu *prodej zboží* je skladový pohyb spjat s dokladem, jehož se týká.

- **StockMovementItem**
  Položka skladového pohybu, jež je vázána na produkt z katalogu zboží, a dále obsahuje jednotkovou cenu, za kterou byl produkt zakoupen/prodán a počet jednotek.

- **Supplier**
  Číselník pro evidenci dodavatelů, jehož položky jsou využity u skladového pohybu.

Kromě názvu dodavatele jsou dále evidovány kontaktní údaje. Položku dodavatele lze ve výpisech skrýt.

Tiketovací systém podpory

- **Ticket**
  Reprezentuje tiket v tiketovacím systému podpory. Tiket obsahuje předmět, typ, datum vytvoření a stav. Pomocí typu se specifikuje, čeho se obecně tiket týká (*zboží*, *objednávka*, *reklamace*, *ostatní*). Stav vyjadřuje, v jaké fázi se tiket nachází (*nový*, *znovuotevřen*, *nová zpráva*, *odpovězeno*). Tiket může být dále vázán na určitý doklad (entita `Receipt`). Samotný obsah tiketu je tvořen kolekcí entit `TicketMessage`, jež představuje vlákno zpráv.

- **TicketMessage**
  Entita představující zprávu ve vlákně zpráv určitého tiketu. Zpráva je pomocí typu dělena na zprávu uživatelskou (obsah tvořen uživatelem) a systémovou, jež informuje o změně stavu tiketu. Na uživatelskou zprávu může být dále vázaná příloha (entita `TicketAttachment`).

- **TicketAttachment**
  Entita obsahuje informace o přiloženém souboru, jenž byl nahrán uživatelem při vytváření zprávy. Samotný soubor je uložen v lokálním souborovém úložišti.

- **UserTicketTypeNotification**
  Entita tvoří vazbu mezi typem tiketu a uživatelem. Na základě ní jsou uživatelům zasílány notifikace o vytvoření tiketu daného typu.

Portál pro zákazníky

- **Faq**
  Reprezentuje položku v sekci *Často kladené otázky*. Každá položka obsahuje název a obsah pro odpověď. Dále je možné přiložit přílohu (např. šablona reklamačního protokolu).

- **News**
  Aktuality, jejichž výpis je uveden v sekci *Aktuality*. Aktualita obsahuje název, datum publikace, obsah a příznak, jenž určuje, zda je aktualita publikovaná (tzn. viditelná).

- **DownloadsFile**
  Nahraný soubor v sekci *Ke stažení*. Entita obsahuje popis souboru a jeho původní název.

7.3 Našeptávač produktů s využitím asociačních pravidel

Jak již bylo zmíněno v návrhu části pro realizaci prodeje 5.2, součástí systému bude „našeptávač“ produktů, který pokladnímu nabídne rychlejší a efektivnější způsob přidávání položek na účet. Logika našeptávače bude fungovat na základě asociačních pravidel, jež jsou určena k hledání zajímavých asociací nad velkým množství dat [17]. Odkryjí nám tak zajímavé vztahy mezi položkami u již realizovaných prodejích za dobu provozování

pokladny. Při nedostatku informací o realizovaných prodejích (např. z důvodu nevyužití katalogu zboží) bude našeptávač prodejci skryt.

Pro hledání asociací mezi produkty bude použit algoritmus *Apriori* 7.3.2, jenž je pro účel použití v této práci dostačující.

7.3.1 Získávání asociačních pravidel

Z transakčních dat budou konkrétně získávány jednoúrovňová booleovská asociační pravidla, u nichž nás zajímá pouze přítomnost nebo nepřítomnost položky. Získaná asociační pravidla budou obecně ve tvaru:

$$A \Rightarrow B \qquad [podpora, spolehlivost]$$

s následujícím významem: Jestliže si zákazník chce koupit (na účtě je) množina položek A, pak by si mohl chtít dále koupit (mohlo by být dále namarkován) produkt z množiny položek B s danou *podporou* a *spolehlivostí*. Konkrétním příkladem může být:

$$\text{rohlík} \wedge \text{chléb} \quad \Rightarrow \quad \text{máslo, mléko} \qquad [\text{podpora} = 10\%, \text{spolehlivost} = 80\%]$$

Zmíněna *podpora* (support) a *spolehlivost* (confidence) jsou metriky udávající zajímavost asociačního pravidla $X \Rightarrow Y$. *Podpora* udává podíl počtu záznamů (dokladů) obsahující všechny produkty určené množinou položek $X \cup Y$ a všech záznamů (dokladů). Naproti tomu *spolehlivost* je poměr počtu záznamů (dokladů) s produkty určené množinou položek $X \cup Y$ vůči počtu záznamů (dokladů) obsahující pouze položky z množiny X. Asociační pravidlo je považováno za zajímavé, je-li splněna podmínka minimální *podpory* a minimální *spolehlivosti*. S využitím pravděpodobnosti lze tedy napsat:

$$podpora(A \Rightarrow B) = P(A \cup B)$$
$$spolehlivost(A \Rightarrow B) = P(B \mid A)$$

Získávání asociačních pravidel probíhá v následujících krocích [20, 17]:

1. Nalezení množin položek, které splňují podmínku minimální podpory (= frekventované množiny).
2. Generování asociačních pravidel z frekventovaných množin probíhá dle následujících kroků:
   (a) Pro každou frekventovanou množinu l se vygenerují všechny neprázdné podmnožiny.
   (b) Pro každou podmnožinu s množiny l je vygenerováno pravidlo $s \Rightarrow (l - s)$ a vypočtena jeho spolehlivost dle rovnice:

   $$spolehlivost(A \Rightarrow B) = P(B \mid A) = \frac{s(A \cup B)}{s(A)}$$

   Pokud je hodnota spolehlivosti pravidla vyšší než minimální spolehlivost s_{min}, pak je asociační pravidlo označeno za silné.
3. Výsledkem algoritmu generování asociací je seznam všech silných asociačních pravidel.

7.3.2 Algoritmus Apriori

Apriori je klíčový algoritmus pro získávání frekventovaných množin s použitím generování kandidátů. Algoritmus využívá předchozí znalost o dříve získaných frekventovaných množinách – tzv. induktivní přístup. V každé iteraci získané frekventované k-množiny jsou použity pro generování $(k+1)$-množin [17]. Každá iterace tak vyžaduje průchod databází. K vyšší efektivitě přispívá tzv. *Apriori vlastnost*. Tato vlastnost říká, že jestliže máme množinu k určitých položek, která není frekventovanou množinou, pak přidání jakékoliv další položky z ní frekventovanou množinu neučiní (tzn. její podpora nevzroste). Platí tedy, že každá podmnožina frekventované množiny je rovněž frekventovanou množinou.

Každá iterace algoritmu se skládá ze dvou kroků, a to:

- **Spojovací krok:** Slouží k nalezení všech frekventovaných množin (L_k) na základě kandidátů na frekventované množiny (C_k), jenž jsou vygenerováni spojením množin z L_{k-1}[1]. Nechť $l_1, l_2 \in L_{k-1}$. Spojení dvou (k-1)-množin ($l_1 \bowtie l_2$) je proveditelné pouze tehdy, pokud jejich prvních k-2 prvků je shodných. Výsledná množina vzniklá spojením l_1 a l_2 bude mít prvky $l_1[1], l_1[2], \ldots, l_1[k-1], l_2[k-1]$. Notace $l_y[x]$ označuje x-tý prvek množiny l_y. Dále musí platit podmínka $l_1[k-1] < l_2[k-1]$, která zamezí vzniku duplicit.

- **Vylučovací krok:** C_k je nadmnožinou L_k, z čehož plyne, že prvky C_k mohou, ale nemusí být frekventované. Zde se využívá *Apriori vlastnosti*, kdy v případě, že nějaká její (k-1)-podmnožina není frekventovaná, pak není frekventovaná ani kandidátní k-množina. V takovém případě můžeme k-množinu z C_k odstranit.

Pseudokód algoritmu *Apriori* je znázorněn v algoritmu č. 1. V prvním kroku jsou vygenerovány *1-množiny*. Dále v jednotlivých iteracích je volána funkce *aprioriGen*, která vykonává spojovací a vylučovací krok. Po vygenerování kandidátů dojde k průchodu databází a výpočtu výskytů jednotlivých kandidátů. Kandidáti, jejichž podpora je vyšší než minimální (s_{min}), jsou uloženy do L_k [17].

Výhody a nevýhody

Algoritmus vyniká poměrně jednoduchou implementací a použitím. Avšak jeho hlavní nevýhodu je efektivita, jelikož v každém cyklu prochází celou databázi pro nalezení všech frekventovaných množin, což je časově a výpočetně náročné. U větších databází může nastat problém s generování kandidátů, kterých může být velmi mnoho. Tyto problémy se snaží eliminovat např. metoda *FP-growth*, která doluje frekventované množiny bez generování kandidátů v datové struktuře zvané *FP-tree*. Její bližší popis lze nalézt například v literatuře *Data mining* [15, s. 257] nebo *Studijní opora k předmětu Získávání znalostí z databází* [17, s. 82].

Zvýšení efektivity

Problém s efektivitou bude v našeptávači produktů řešen využitím databáze, do které se budou ukládat vygenerovaná asociační pravidla. Jejich generování bude probíhat automatizovaně pomocí softwarového démona v operačním systému po určeném časovém období (nejspíše jednou za den), nebo po vložení určitého počtu dokladů od posledního vygenerování.

[1] Algoritmus předpokládá, že množina obsahuje lexikograficky seřazené položky.

Algoritmus 1 Apriori

Vstup: *databáze* D, s_{min}
Výstup: *L-frekventované množiny v* D

1: $L_1 \leftarrow$ nalezniFrekventovane_1-mnoziny(D)
2: $k \leftarrow 2$
3: **while** $L_{k-1} \neq \phi$ **do**
4: $C_k \leftarrow$ APRIORIGEN(L_{k-1}) *//generování nových kandidátů*
5: **for all** transakce $t \in D$ **do** *//projdi D pro zjištění počtů výskytů*
6: $C_t \leftarrow \{c \mid c \in C_k \wedge C \subseteq t\}$
7: **for all** kandidát $c \in C_t$ **do**
8: count[c]++
9: **end for**
10: **end for**
11: $L_k \leftarrow \{c \mid c \in C_k \wedge count[c] \geq s_{min}\}$
12: $k \leftarrow k + 1$
13: **end while**
14: **return** $L \leftarrow \bigcup_k L_k$

1: **function** APRIORIGEN(L_{k-1} :frekventované (k-1)-množiny)
2: **for all** množina $l_1 \in L_{k-1}$ **do**
3: **for all** množina $l_2 \in L_{k-1}$ **do**
4: **if** $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2])$
5: $\wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ **then**
6: c $\leftarrow l_1 \bowtie l_2$ *//spojovací krok – generování kandidátů*
7: **if** máNefrekvPodmnož(c, L_{k-1}) **then** *//vylučovací krok*
8: odstraň c *//odstranění nefrekventovaných kandidátů*
9: **else**
10: přidej c do C_k
11: **end if**
12: **end if**
13: **end for**
14: **end for**
15: **return** C_k
16: **end function**

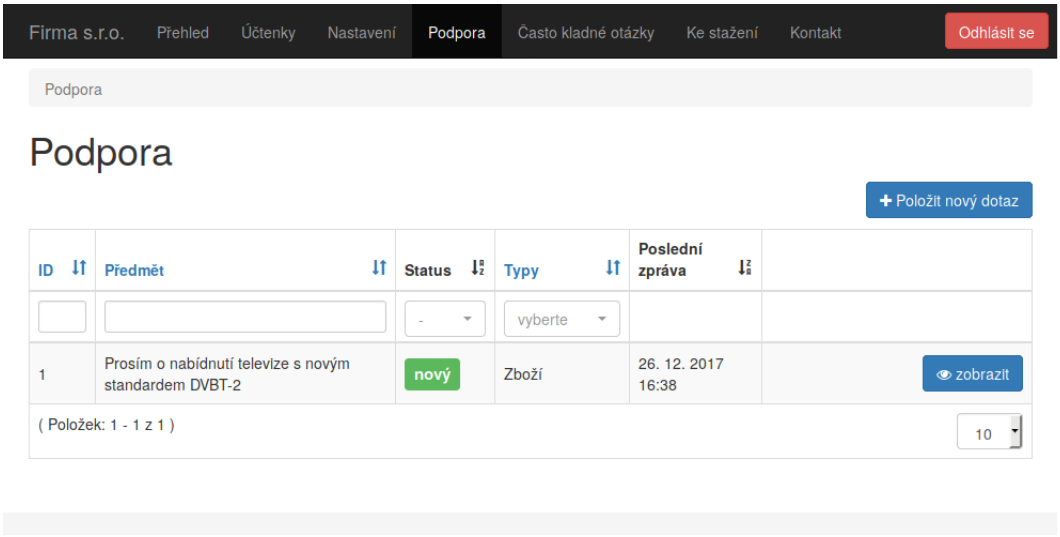
zdroj: Studijní opora k předmětu Získávání znalostí z databází [17]

7.4 Rozložení uživatelského rozhraní aplikace

Návrh je založen na použití již existujících šablon. Každá část je postavena na jiné šabloně z důvodu odlišných požadavků na uživatelské rozhraní. Použité šablony plně podporují responzivní zobrazení, díky čemuž se vzhled aplikace přizpůsobí velikosti obrazovky použitého zobrazovacího zařízení. Jednotlivé části jsou dále popsány na vybraných demonstračních obrazovkách.

7.4.1 Portál pro zákazníky

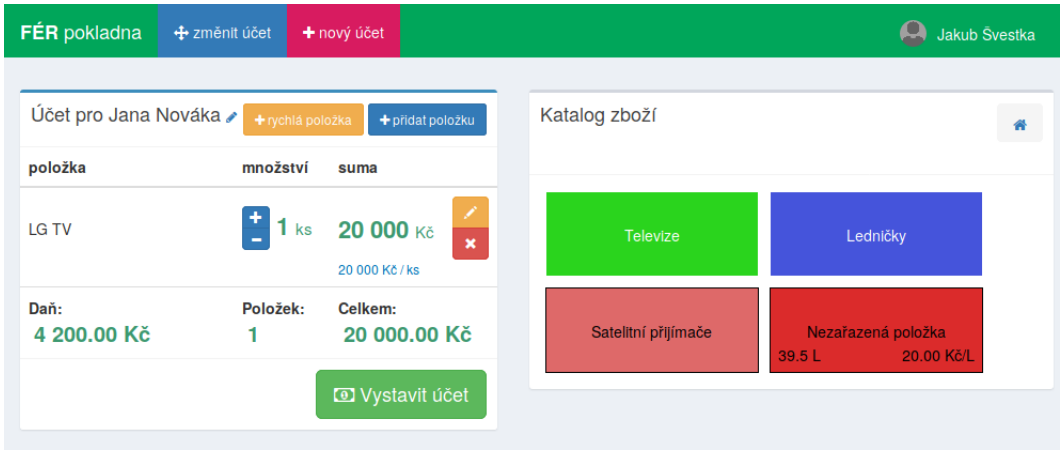
Portál pro zákazníky je založen na frameworku *Bootstrap* s využitím výchozí šablony. Snahou bylo šablonu nijak nerozšiřovat a zachovat tak její přehlednost a jednoduchost. Uživatelskému rozhraní dominuje horizontální navigace, kde v levé části je umístěn název firmy a v pravé části tlačítko pro odhlášení. Pod navigací je zobrazen nadpis stránky a dále drobečková navigace, která uživatele informuje o tom, kde se právě nachází. Zbylou část stránky tvoří obsah. Ukázka návrhu je na obrázku 7.1.



Obrázek 7.1: Obrazovka se správou tiketů v portálu pro zákazníky

7.4.2 Pokladna

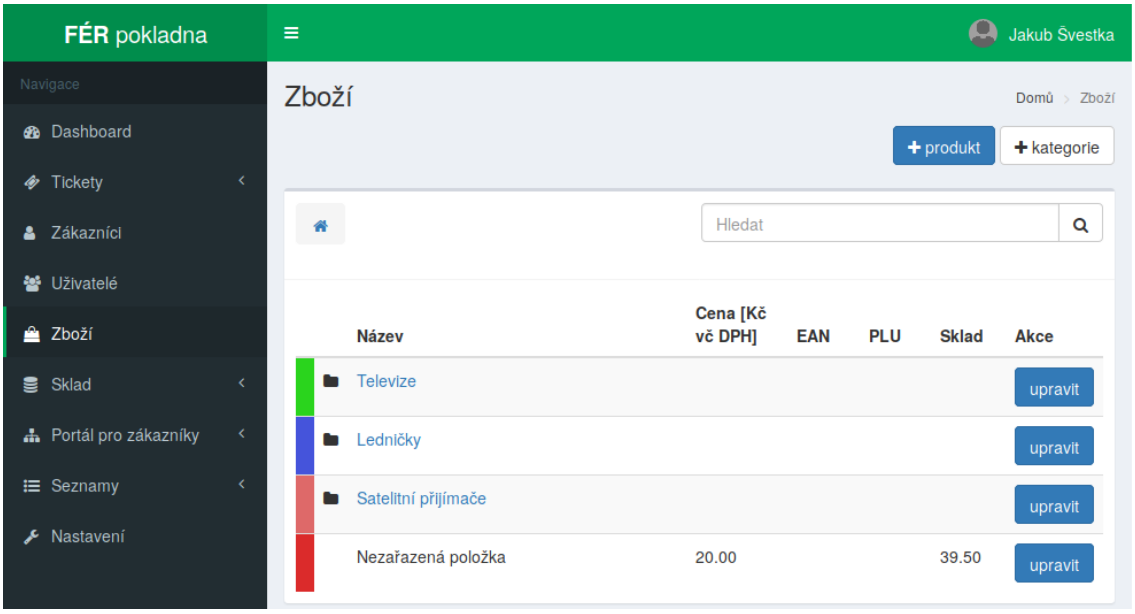
Pokladna je již postavena na šabloně *AdminLTE*, jelikož nabízí větší škálu předdefinovaných komponent. Z důvodu využití celé šířky okna je využita verze šablony s horizontální navigací. Jednotlivé položky v navigaci jsou pro lepší orientaci barevně odlišeny. Položky v navigaci jsou proměnlivé v závislosti na tom, na jaké stránce se uživatel nachází. Ukázka návrhu je na obrázku 7.2.



Obrázek 7.2: Hlavní obrazovka pokladny

7.4.3 Administrace

Administrace pro zachování stylu rovněž využívá šablony *AdminLTE*. Nicméně je zde použita varianta s vertikální navigací, kterou lze v případě potřeby zúžit pouze na zobrazení ikon. Položky navigace jsou z důvodu vysokého počtu seskupovány do kategorií. V obsahové části je rovněž zobrazena drobečková navigace, tak jako u portálu pro zákazníky. Ukázka návrhu je na obrázku 7.3.



Obrázek 7.3: Obrazovka s katalogem zboží v administrace

Kapitola 8

Implementace

Tato kapitola se již zabývá samotnou implementací *EET pokladny*, jež byla navržena v kapitole 7. Celá implementace je založena na `PHP` frameworku *Nette* s maximálním využitím prostředků, jenž nabízí, a dále knihoven, které doplňují specifické funkcionality nutné pro chod aplikace. Veškeré závislosti jsou uvedené v konfiguračních souborech balíčkovacích systémů (viz 8.1), pomocí nichž lze automatizovaně provést instalaci nebo případně aktualizaci. Díky tomu je možné jednoduše udržovat knihovny v aktuální verzi, a snížit tak riziko eventuální hrozby z důvodu možné zranitelnosti.

Celá aplikace je dle návrhu dekomponována na moduly se striktním dodržením návrhového vzoru `MVP` 6.1. Ve většině případů jsou větší celky či často používané části realizované pomocí komponent, díky čemuž je aplikační kód komponenty oddělený, přehlednější a především znovupoužitelný. Úprava logiky dané funkcionality se tak neprovádí na několika místech, ale pouze v samotné komponentě, která za funkcionalitu zodpovídá.

Z důvodu větší rozsáhlosti aplikace je následující text zaměřen na podstatné části, které jsou implementačně zajímavé, představují stěžejní funkci nutnou pro chod pokladny nebo u nich byly řešeny problémy, ze kterých bych některé rád vyzdvihl.

8.1 Adresářová struktura

Adresářová struktura je navržena tak, aby odpovídala zvyklostem použitého frameworku. Tím je docíleno oddělení souborů jednotlivých vrstev použitého návrhového vzoru `MVP`, zvýšení přehlednosti aplikačního kódu a zejména zajištění bezpečnosti celé aplikace. Z důvodu větší rozsáhlosti adresářové struktury je v příloze B znázorněna pouze její část, a to první tři úrovně adresářové hierarchie.

V kořenovém adresáři se nachází čtyři důležité konfigurační soubory, a to:

- `composer.json`
  Konfigurace pro nástroj *Composer* (viz 6.2.1). Obsahuje závislosti na `PHP` knihovny a rovněž na samotnou verzi `PHP` včetně potřebných rozšíření. Stažené knihovny jsou umístěny ve složce `vendor`.

- `bower.json`
  Konfigurace pro nástroj *Bower*. Obsahuje závislosti na frontendové balíčky, jako jsou *JavaScriptové* či `CSS` knihovny (např. Bootstrap 6.5.1). Stažené balíčky jsou umístěny ve složce `bower_components`.

- `package.json`
  Konfigurace pro nástroj *NPM*. Podobný nástroji *Bower*, nicméně oproti němu obsahuje dále závislosti na podpůrné pluginy pro nástroj *Grunt*. Stažené balíčky jsou umístěny ve složce `node_modules`.

- `Gruntfile.js`
  Konfigurace pluginů a procedur pro nástroj *Grunt*.

Dalším adresářem je `assets`, který obsahuje veškeré *JavaScriptové* soubory, `CSS` styly a obrázky. Soubory v adresáři `assets`, `bower_components` a `node_modules` jsou dále zpracovávány pomocí nástroje *Grunt*.

Adresář `data` slouží jako úložiště souborů pro samotnou aplikaci. Jsou zde uloženy přílohy vložené pomocí tiketovacím systému, soubory ke stažení a certifikát pro `EET`. Všechny zachycené chybové stavy jsou ukládány do logovacích souborů do adresáře `log`. Veškeré dočasné soubory se ukládají do adresáře `temp`. Adresáře zmíněné v tomto odstavci musí být přístupné webovým serverem pro zápis (a též čtení).

Předposledním adresářem je adresář `www`. Ten je jako jediný veřejně přístupný skrze webový server. Nacházejí se zde pomocné soubory jako `CSS` styly, *JavaScriptové* soubory, písma, grafické soubory a především soubor `index.php`. Ten slouží jako vstupní soubor pro každý požadavek a předává řízení do aplikace (tj. do adresáře `app`).

Adresář `app` obsahuje zdrojové kódy s aplikační logikou a konfigurační soubory. Je zde umístěn důležitý zaváděcí soubor aplikace `bootstrap.php`, jenž provádí inicializaci prostředí a vytvoření `DI`[1] kontejneru. Dále v něm nalezneme adresář `config`, který obsahuje veškeré nastavení aplikace. Soubor `config.neon` obsahuje globální nastavení, mezi něž patří nastavení rozšíření frameworku, konfigurační parametry pokladny a definice služeb, které chceme mít dostupné z `DI` kontejneru. Soubor `config.local.neon` rozšiřuje předchozí soubor a obsahuje specifické nastavení pro danou instanci aplikace – např. přístupové údaje k databázi. Adresář `components` obsahuje komponenty, které jsou společné pro všechny moduly – např. `Datagrid`. Adresář `forms` obsahuje třídu `BaseForm`, která slouží k vytváření a zpracování webových formulářů. Třída `BaseForm` je pouze potomek třídy `\Nette\Application\UI-\Form`, kterou nalezneme v samotném frameworku *Nette*. Nicméně oproti ní přidává další kontrolní prvky formulářů, které se nacházejí v podadresáři `Controls` a jsou jimi:

- `CKeditor` – `WYSIWYG` editor
- `ColorPicker` – výběr barvy
- `DatePicker` – kalendář pro výběr data.

Routování `URL` (tzn. obousměrné překládání mezi `URL` a aplikačním požadavkem) je realizováno na základě továrny `RouterFactory` v adresáři `router`, která definuje způsob překladu. Adresář `presenters` obsahuje třídu `BasePresenter`, od které dědí všechny další třídy, jež představují prezenční vrstvu. Adresář `model` a adresáře modulů (`*Module`) budou popsány dále.

8.1.1 Moduly pokladního systému

Dle dekompozice provedené v části návrhu 7.1, byl pokladní systém rozdělen na moduly. Modulem se rozumí adresář, který rozčleňuje *pohledy (views)* a *řadiče (controllers)* do spolu

[1] Dependency Injection – vkládání závislostí mezi jednotlivými komponentami.

souvisejících celků. Každý z modulů dále obsahuje adresář `components`, jenž obsahuje implementované komponenty. Z důvodu znovupoužitelnosti komponent jsou některé z nich mezi moduly sdílené. V systému se vyskytují celkem čtyři moduly, jejichž funkcionality budou dále nastíněny.

Pokladní modul – adresář `CashModule`

V modulu se nachází veškerá funkcionalita nutná pro realizaci prodeje, a to:

- markování účtu
- správa otevřených účtů (na aktuálně otevřené pokladně)
- správa vydaných dokladů (na aktuálně otevřené pokladně)
- detail vydaného dokladu (viz C.2)
- změna pokladny
- změna režimu zobrazení pro markování účtu (viz 8.2.1).

Administrační modul – adresář `AdminModule`

Hlavní modul, jenž slouží pro správu ostatních modulů a zejména správu veškerých dat pokladního systému. Modul obsahuje řadu funkcionalit, mezi které patří např.:

- správa vydaných dokladů ze všech pokladen
- správa katalogu zboží
- správa zákazníků a uživatelů
- tiketovací systém podpory
- správa skladu (naskladnění, skladové pohyby)
- správa číselníků – sazby `DPH`, dodavatelé, měrné jednotky
- správa pokladen vč. přiřazení práv
- nastavení kontaktních údajů provozovatele a `EET`
- statistiky (viz 8.6)
- log s historií evidence pokusů o zaslání tržeb do `EET`.

Modul portálu pro zákazníky – adresář `PortalModule`

Modul primárně nabízí funkcionality pro přihlášené zákazníky, a to:

- přístup k vlastněným dokladům
- zobrazení aktualit
- komunikace s prodejcem v rámci podpory
- zobrazení často kladených otázek
- stažení nahraných souborů.

Modul podpory (tiketovací systém) – adresář `TicketsModule`

Modul je využíván předchozími moduly. Slouží pouze k oddělení aplikačního kódu a komponent od ostatních modulů.

8.2 Realizace prodeje

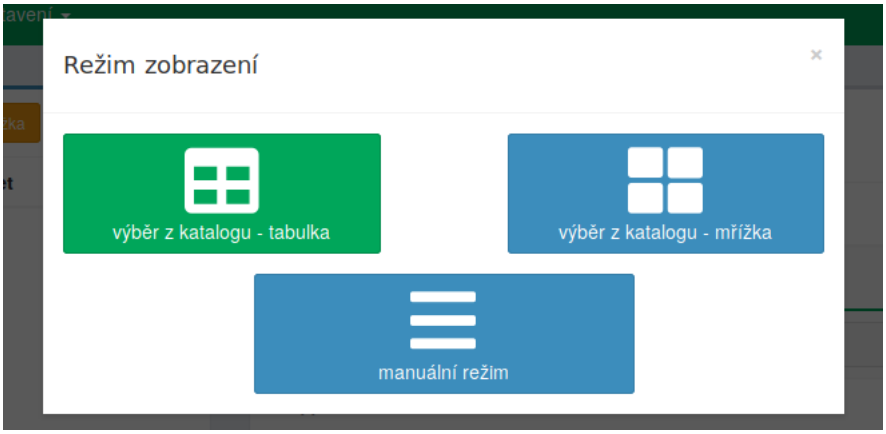
Realizace prodeje probíhá v rámci pokladního modulu. Pro realizaci prodeje musí mít pokladní přiděleno oprávnění k nejméně jedné pokladně. V opačném případě je pokladnímu přístup odepřen.

Změna aktivní pokladny je umožněna pomocí komponenty `SelectCashModal`. Aby nebylo nutné při každém přístupu do pokladní části vyžadovat po pokladním zvolení pokladny, je zvolená pokladna „zapamatována“ pomocí objektu `CachedCashierData`, jenž je uložen v keš paměti. Pokud dojde k přístupu do modulu bez zvolené pokladny, je automaticky zvolena první pokladna, ke které má pokladní oprávnění.

8.2.1 Režimy pro tvorbu účtu

Tvorba účtu probíhá pomocí tzv. režimu pohledu. Ve skutečnosti se jedná o komponentu, jež nabízí prostředí pro vkládání položek na otevřený účet. Aplikace na základě specifikace požadavků nabízí požadované režimy, jež budou dále popsány. Všechny režimy umožňují vložení produktu za pomocí čtečky čárových kódů.

Každý pokladní může mít v rámci jedné pokladny zvolen odlišný režim pohledu. Jeho změna probíhá pomocí komponenty `ChangeView` (viz obrázek 8.1). Vybraný pohled je na



Obrázek 8.1: Vizuální podoba komponenty `ChangeView`

základě sezení (tzn. pokladního a zvolené pokladny) uložen objektem `CachedCashSettings`, který je umístěném v keš paměti. V objektu se dále nachází identifikátor otevřeného (rozpracovaného) účtu, na základě něhož je pro ostatní pokladní uzamčen.

8.2.2 Režim katalogu

Režim se skládá ze dvou komponent, kde první z nich slouží k vizualizaci účtu (`ReceiptVisualization`) a druhá k výběru produktů. Komponenta pro výběr produktů byla oproti specifikaci požadavků rozdělena na dvě varianty, na základě čehož došlo k dekompozici režimu na dva režimy, a to na zobrazení katalogu pomocí *mřížky*, nebo *tabulky*.

Obě varianty komponent pro výběr produktů jsou potomky třídy `ProductSelect-\ProductSelect`. Komponenty přímo nepracují s objektem aktuálně tvořeného účtu. Z tohoto důvodu bylo nutné zajistit komunikaci s objektem účtu na vyšší úrovni, a to pomocí rodičovské komponenty daného režimu. Přidání produktu na účet je realizováno pomocí události `onProductSelected`, na kterou naslouchá rodičovská komponenta. Událost je vyvolána s argumentem, jenž obsahuje produkt pro přidání.

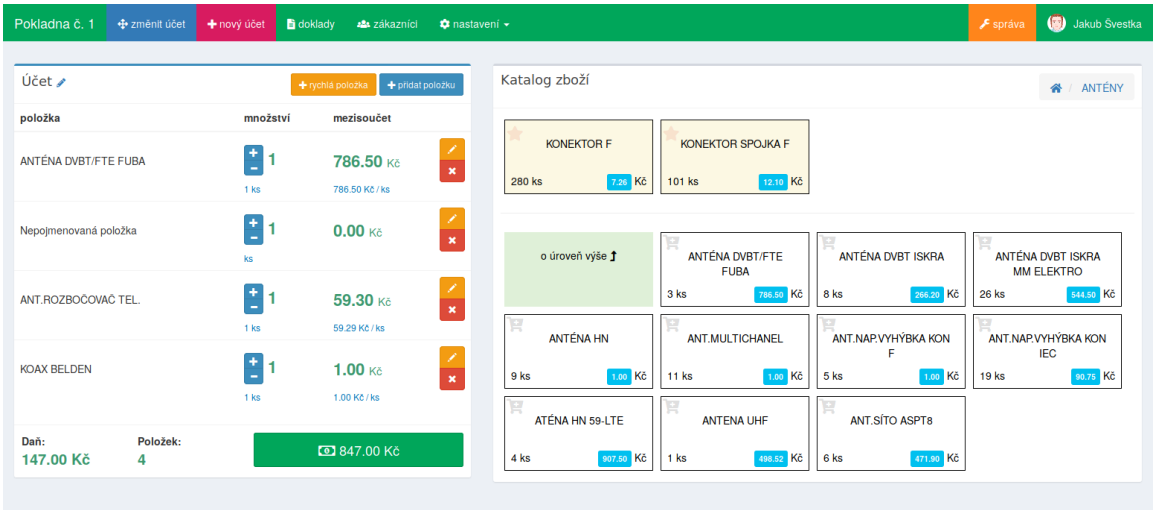
Aby bylo možné v katalogu odlišit produkty, které již byly přidány na účet, je nutné, aby rodičovská komponenta předala atributem `isProductInReceiptCallback` *callback* funkci, která parametrem přijímá objekt produktu (entita `Product`). Pomocí ní se komponenta při výpisu produktů dotazuje a dle výsledku případě produkt odliší od produktů, které ještě na účtě nefigurují.

Veškeré akce při práci s katalogem zboží probíhají asynchronně za pomocí *ajaxu*. Díky tomuto přístupu nedochází zbytečně k překreslení celé stránky, ale pouze určitých částí. Při překreslení je zobrazen tzv. *spinner*, jenž uživateli indikuje načítání dané části.

Mřížka

Komponentu implementuje třída `ProductSelect\GridView`. Varianta je uzpůsobená pro dotyková zařízení, kdy jsou položky z katalogu zobrazeny pomocí boxů, díky čemuž je práce s katalogem dotykově přívětivější – viz obrázek 8.2. V horní části se nacházejí našeptané produkty, jež jsou od ostatních odlišené ikonou ★.

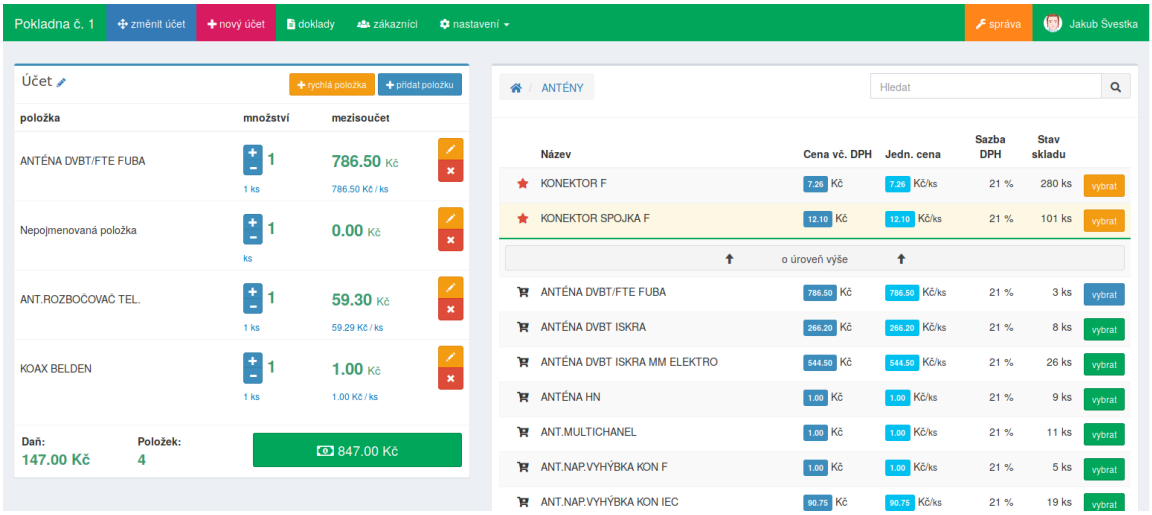
Boxy se složkami a produkty mohou být podbarveny. Barva podbarvení je definována uživatelem ve správě katalogu zboží. Aby se zajistila čitelnost textu uvnitř boxu, musí být barva textu variabilně určena na základě barvy pozadí. K tomuto účelu slouží statická metoda `getContrastYIQ(string $hexcolor)` ze třídy `ColorHelper`, která převede barvu pozadí z `RGB` modelu do modelu `YIQ` a na základě algoritmu určí, zda barva textu bude *bílá*, nebo *černá*.



Obrázek 8.2: Tvorba účtu – výběr z katalogu pomocí komponenty „mřížka“

Tabulka

Komponentu implementuje třída `ProductSelect\TableView`. Výpis katalogu zboží je realizován pomocí tabulky, tudíž je možné oproti předchozí komponentě vypsat více informací o produktu – viz obrázek 8.3. Komponenta nabízí funkci vyhledávání, kdy v případě její aktivace dojde k vyhledávání kategorií a produktů pouze v úrovni hierarchie katalogu, kde se pokladní právě nachází. V případě změny zanoření v hierarchii zůstane vyhledávání aktivní. Na prvních řádcích se nacházejí našeptané produkty, jež jsou od ostatních odlišené ikonou ★.



Obrázek 8.3: Tvorba účtu – výběr z katalogu pomocí komponenty „tabulka“

8.2.3 Režim okamžitého prodeje

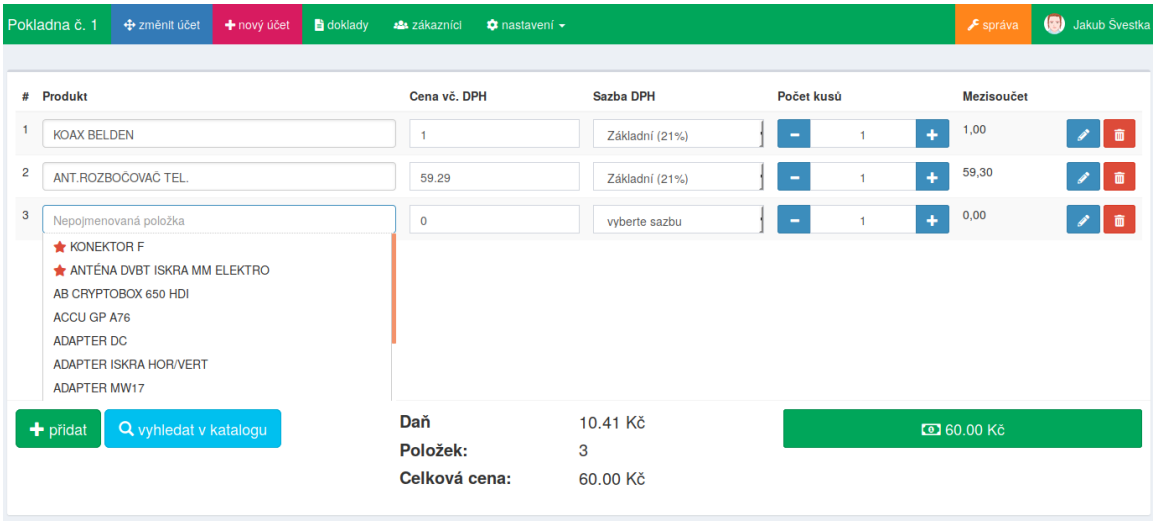
Poslední režim se skládá pouze z jedné komponenty, a to `ReceiptImmediateSale`. Komponenta slouží jak k vizualizaci účtu, tak i k přidávání položek. Hlavní část komponenty tvoří tabulka, kde řádek odpovídá jedné položce na účtu – viz obrázek 8.4. Řádek je přímo tvořen několika formuláři, kde každý z nich slouží k editaci jednoho atributu.

V řádku se editují základní atributy položky, jako její název, cena (vč. `DPH`), sazba `DPH` (pouze pro plátce) a počet kusů. Rozšířené atributy (jako poznámka) se editují v modálním okně pomocí komponenty `AddEditReceiptItemModal`.

U atributu název je použit *JavaScriptový* plugin *EasyAutocomplete*[2], který slouží k automatickému doplňování názvu produktu na základě katalogu zboží. Je-li vybrán název položky z nabídnutých produktů, je tento úkon považován za výběr položky z katalogu. Na základě toho dojde k automatickému doplnění zbylých atributů, které si pokladní může případně změnit.

V tomto režimu se rovněž využívá našeptávač produktů s mírně upravenou logikou, která spočívá v tom, že našeptaný produkt musí v názvu obsahovat řetězec, jenž byl uživatelem zadán ve formuláři. Našeptané produkty jsou zobrazeny na prvních pozicích v nabídnutých produktech a jsou odlišené ikonou ★.

[2]Dostupný z http://easyautocomplete.com.



Obrázek 8.4: Manuální tvorba účtu

Ve specifikaci požadavků bylo zmíněno, že tento režim je určen pro provozovatele bez zavedeného katalogu zboží. Nicméně výsledná implementace je vhodná i v případě jeho využívání.

8.2.4 Doklad

Pokladna podporuje základní typy dokladů, jejichž výčet je definován ve třídě `TypeOfDocument`. Jejich nabídka je přizpůsobena na základě toho, zda je či není provozovatel plátcem `DPH`, a to následovně:

- **Neplátce `DPH`**
  - pokladní doklad
  - faktura
- **Společné**
  - zálohová faktura
  - storno
- **Plátce `DPH`**
  - zjednodušený daňový doklad
  - faktura - daňový doklad

Na základě vybraného typu je pokladnímu při dokončení účtu nabídnuto nastavení odlišných atributů. U každého typu dokladu lze vyplnit údaje o zákazníkovi. Nicméně u *zjednodušeného daňového dokladu* a *pokladního dokladu* je tato informace nadbytečná a slouží pouze jako informace pro pokladního/provozovatele.

Číslování dokladů

Pro číslování dokladu není použit pouhý číselný identifikátor vytvořený databázovým systémem, jelikož sám o sobě neobsahuje žádnou další informaci, která by se dala využít pro jeho bližší identifikaci. Proto byla zvolena možnost složeného identifikátoru, tedy identifikátoru skládajícího se z několika údajů, a to následujících:

`{číselný typ dokladu}` / `{rok vytvoření}` / `{měsíc vytvoření}` / `{ID}`

Takto z čísla dokladu ihned zjistíme, o jaký typ se jedná a přibližné období jeho vytvoření. Zároveň je pomocí hodnoty `ID` zajištěna jeho unikátnost.

Aby bylo možné pro každý typ dokladu vést zvláštní sekvenci pro generování `ID`, bylo nutné rozšířit návrh o entitu `DocumentIdSequence`, jež si pro každý typ dokladu uchovává hodnotu čítače `ID`[3].

Generování dokladu do PDF

Pro generování dokladu do `PDF` je určena služba `ReceiptPDFGenerator`, jež při své instanciaci očekávává objekt dokladu (entita `Receipt`). Služba využívá knihovnu *mPDF*[4], která umožňuje vytvářet `PDF` dokumenty pomocí jazyka `HTML`.

Služba využívá dvě šablony dokladů, a to:

- **Doklad bez hlavičky odběratele**
  Využit pro *pokladní doklad*, *zjednodušený daňový doklad* a *storno*. Ukázka na obrázku D.1.
- **Doklad s hlavičkou odběratele**
  Využit pro *fakturu*, *faktura - daňový doklad* a *zálohovou fakturu*. Ukázka na obrázku D.2.

Při generování dokladu se zohledňuje to, zda je provozovatel plátcem `DPH`. Na základě toho je struktura dokladu přizpůsobena tak, aby doklad splňoval veškeré náležitosti dané zákonem. Doklad kromě textové reprezentace čísla dokladu obsahuje rovněž čárový kód typu `CODABAR`[5], v němž je číslo dokladu zakódováno.

8.3 Detail účtu

Detail účtu je realizován pomocí komponenty `ReceiptDetail`. Získání instance komponenty probíhá skrze továrnu `ReceiptDetailFactory`, jež obsahuje dvě veřejné metody, a to:

- `create(Receipt $receipt)` a
- `createCustomerView(Receipt $receipt)`.

Metody se liší pouze v tom, že pomocí druhé je vytvořena komponenta v zákaznickém režimu, který má omezenou funkcionalitu. Obě metody očekávají v parametru objekt dokladu, který má být komponentou vizualizován.

Komponenta se skládá ze čtyř částí, jež budou dále popsány. Kompletní vizualizace komponenty je vyobrazena na obrázku C.2.

8.3.1 Tlačítka s akcemi

Tlačítka s akcemi jsou rozděleny do dvou skupin, kde první skupina (umístěna vlevo) je určena k úkonům, jež jsou spjaty s předáním dokladu zákazníkovi (viz obrázek 8.5). Tlačítko pro *tisk* je obslouženo signálem `handlePrintReceipt()`. Pomocí něj dojde k otevření nového okna s dokladem v `PDF` a vyvolání pop-up dialogu pro tisk. Oproti tomu tlačítko *stáhnout*, jež je obslouženo signálem `handleDownloadReceipt()`, provede přímé stažení

[3] Z důvodu restrikcí použitého `ORM` nebylo možné vytvořit složený identifikátor na straně databáze.
[4] Viz https://github.com/mpdf/mpdf.
[5] Bližší specifikace dostupná z https://en.wikipedia.org/wiki/Codabar.

dokladu v `PDF` (bez jeho zobrazení). Oba signály využívají službu `ReceiptPDFGenerator` (viz 8.2.4). Tlačítko pro webový doklad slouží k zobrazení modálního okna s odkazem na doklad. Modální okno je tvořeno komponentou `ShowReceiptLink`. Po kliknutí na tlačítko dojde k vyvolání signálu `handleShowReceiptLink()`, jenž zobrazí modálního okno. Poslední tlačítko v této skupině je určeno k zaslání dokladu na email. Po kliknutí dojde ke zpracování akce signálem `handleSendReceipt()`, na základě čehož se zobrazí modální okno vytvořené komponentou `SendReceiptModalDialog`. V modálním okně se nachází textové pole pro email, kde je automaticky předvyplněn email, jenž byl vyplněn v kontaktních údajích odběratele.

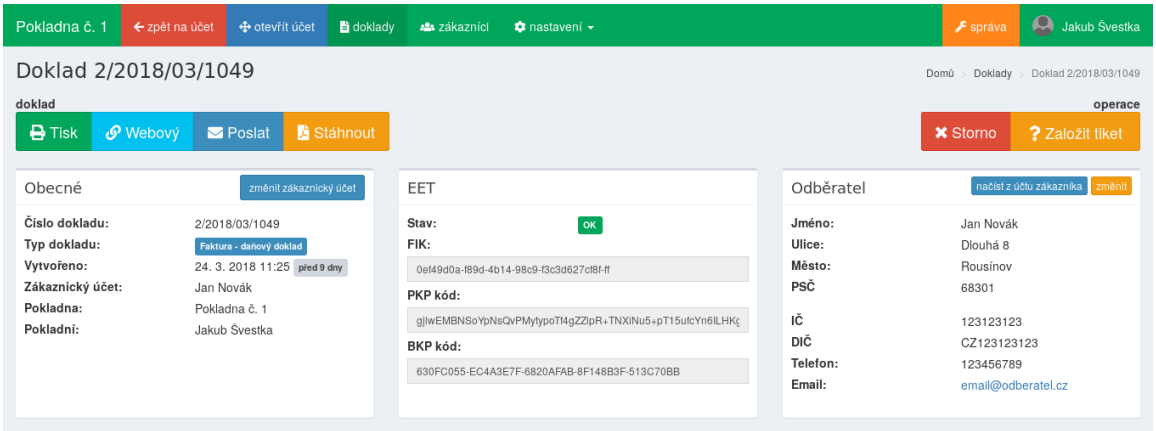
Druhá skupina tlačítek tvoří akce, které slouží k provedení storna dokladu a založení tiketu. Storno dokladu představuje modální okno tvořené komponentou `CancelReceiptDialog`. V modálním okně se nachází zaškrtávací tlačítko, pomocí něhož pokladní určí, zda má být zboží zpětně naskladněno. Založení tiketu je obslouženo signálem `handleCreateTicket()`, který pouze vyvolá událost `onCreateTicket`. Na tuto událost naslouchá rodičovská komponenta, která provede přesměrování na stránku pro založení tiketu, kde je předvyplněn daný doklad.

V režimu zákazníka jsou zobrazeny pouze tlačítka pro tisk a stažení dokladu.

8.3.2 Informační boxy

Informace o dokladu jsou rozděleny do tří boxů (viz obrázek 8.5). První box obsahuje obecné informace o dokladu, jako např. číslo, datum a čas vytvoření, jméno přiřazeného zákazníka atd. Pokladní může pomocí komponenty `AssignCustomer` změnit přiřazeného zákazníka či jej od dokladu odebrat. Změna přiřazeného zákazníka je možná pouze v režimu pokladního.

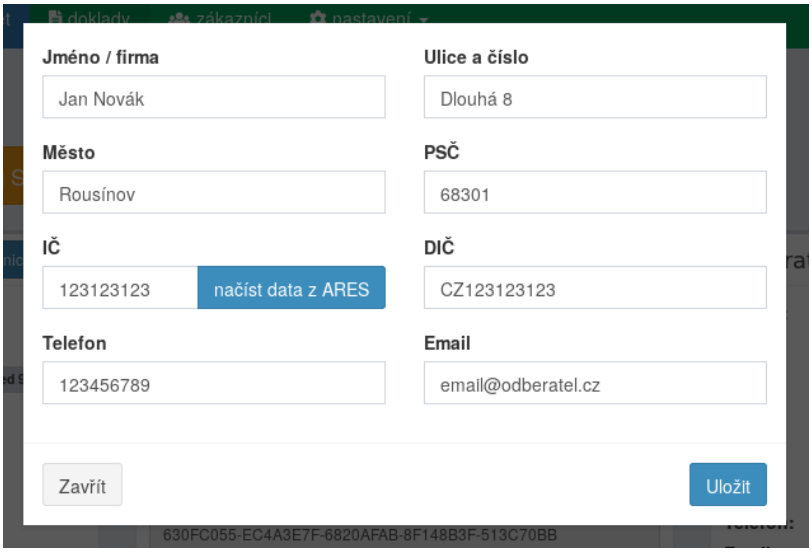
Druhý box obsahuje informace o stavu evidence do `EET`. Jestliže se nepodařilo zaevidovat tržbu, má zde pokladní možnost pokus o zaevidování manuálně vynutit. Pokud doklad nebyl vytvořen se zapnutou funkcionalitou evidence tržeb do `EET`, je box skrytý.



Obrázek 8.5: Detail dokladu – akce a informační boxy

Poslední box slouží k nastavení kontaktních údajů odběratele. Ty mohou být automaticky načteny z účtu zákazníka, jenž je k dokladu přiřazen (signál `handleLoadCustomerDataFromAccount()`) nebo přímo upraveny v modálním okně pomocí komponenty `ChangeCustomerData` (viz obrázek 8.6). V případě firemního zákazníka mohou být kontaktní

údaje načteny automaticky z ARES[6] databáze na základě IČ. Změna údajů je možná pouze v režimu pokladního.



Obrázek 8.6: Změna kontaktních údajů odběratele – komponenta ChangeCustomerData

8.3.3 Položky

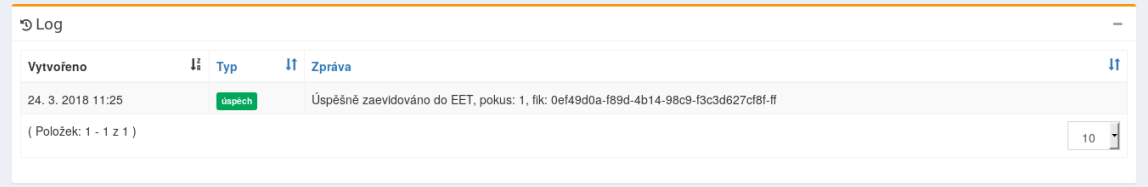
Tato část se výrazně odlišuje od toho, zda byl doklad vydán plátcem DPH. U každé položky jsou zobrazeny atributy, jež se týkají počtu jednotek, sazby DPH a cen. Ve spodní části vlevo se nachází tabulka s vyčíslením jednotlivých sazeb DPH a vpravo celková suma. Vypočtení a správné zaokrouhlení je realizováno pomocí metod v entitě dokladu (Receipt) a entitě položky na dokladu (ReceiptItem). Viz obrázek 8.7.

Obrázek 8.7: Detail dokladu – položky

[6] Administrativní registr ekonomických subjektů.

8.3.4 Log

Poslední část je zobrazena pouze v režimu pokladního a slouží k zobrazení log záznamů. Zatím jsou zaznamenávány pouze informace týkající se pokusů o evidenci tržby do `EET`. Viz obrázek 8.8.



Obrázek 8.8: Detail dokladu – log

8.4 Webový doklad

Termín označující předání dokladu pomocí `URL` adresy. Při volbě schématu `URL` adresy bylo prioritou, aby byla co nejkratší, jelikož ve většině případů bude zákazníkem manuálně přepisována do webového prohlížeče, a dále zabezpečena proti lámání. Lámáním je myšlena situace, kdy by uživatel získával adresy jiných dokladů, a to na základě zjištěné logiky, jež je využita při generování adresy. Příkladem může být adresa, která by obsahovala číselný identifikátor dokladu (např. `10`). Takto by mohl uživatel postupně zkoušet menší či vyšší čísle než `10`, a dostal by se tak k dokladům, jenž mu nenáleží. Z výše uvedených důvodů bylo nutné zvolit sofistikovanější přístup, jenž spočívá v identifikaci dokladu pomocí unikátního tokenu.

Jako první možnost se nabízelo využít hashovací funkci `md5`, pomocí niž by se vytvořil hash z konkatenace identifikátoru a tzv. kryptografické soli[7]. Kryptografická sůl by zabránila hádání hashe jiných dokladů – tzn. útočník by si nemohl bez znalosti soli vypočítat hash jiného identifikátoru dokladu. Nicméně toto řešení nesplňuje požadavek na krátkou délku, jelikož je hash příliš dlouhý (32 znaků v hexadecimální soustavě). Což by bylo při manuálním přepisu uživatelsky nepřívětivé.

Druhá možné řešení spočívalo v převedení identifikátoru dokladu z desítkové soustavy do soustavy o základu 62, jež z celého čísla vytvoří řetězec se znaky 0-9, a-z a A-Z. Výhoda tohoto řešení spočívá v možné reverzním převedení zpět do desítkové soustavy a získání původního identifikátoru. Ovšem problém spočívá v možném hádání identifikátorů jiných dokladů, jelikož výsledný řetězec je sekvenční – např.:

$$21_{10} \rightarrow 3LF_{62}$$
$$22_{10} \rightarrow 3LG_{62}$$
$$23_{10} \rightarrow 3LH_{62}$$

Po bližším zkoumání dalších možných řešeních byla nalezena knihovna `Hashids`, jež přímo splňovala požadavky. Knihovna umožňuje generování krátkých, unikátních a nesekvenčních řetězů z číselných identifikátorů náležících do $\mathbb{N}_0$. Při práci s knihovnou lze nastavit vlastní abecedu, kryptografickou sůl a minimální délku vygenerovaného řetězce. Díky konfigurovatelnosti knihovny je zajištěna její unikátnost. Útočník ji tak nemůže zneužít pro

[7] Náhodně zvolený řetězec, jež uživateli není znám.

vygenerování identifikátorů jiných dokladů bez znalosti použitého nastavení. Vygenerované řetězce lze opět jednoduše dekódovat na původní číselný identifikátor (nutné použít shodnou abecedu a kryptografickou sůl).

8.4.1 Komunikace s knihovnou

Veškerá komunikace s knihovnou je realizována pomocí třídy `ReceiptTokenHelper`, která zajišťuje konfiguraci knihovny, a dále obsahuje následující dvě veřejné statické metody:

- `encode(Receipt $receipt) : string`
  Metoda provede vygenerování unikátního tokenu (řetězce) z číselného identifikátoru dokladu, pod kterým je uložen v databázi (tzn. nejedná se o číslo, jež figuruje na samotném dokladu).
- `decode(string $token) : ?int`
  Metoda zajišťuje zpětné dekódování tokenu do číselné podoby. V případě neúspěchu je navrácena hodnota `NULL`.

Při konfiguraci knihovny byly z abecedy vynechány znaky, které jsou při manuálním přepisu lehce zaměnitelné (např. 1 a l), a dále nastavena minimální délka výsledného tokenu na 5 znaků.

8.4.2 Zobrazení webového dokladu

Je realizováno v modulu portálu pro zákazníky pomocí pohledu `ReceiptTokenPresenter`. Ten na základě tokenu předaného v `URL` adrese provede zobrazení dokladu pomocí komponenty `ReceiptDetail` (viz 8.3). Schéma `URL` je následující:

`http://domena/{TOKEN}`

V případě, že je zákazník přihlášen a požadovaný doklad je přiřazen k jeho účtu, dojde k přesměrování zákazníka na doklad v zákaznickém účtě.

8.5 Čtečka čárových kódů

Aplikace podporuje práci se čtečkou čárových kódů, která v `ASCII` módu funguje podobně jako periferie klávesnice. Čtečka po načtení kódu postupně zasílá dekódované znaky do zařízení a konec zasílání signalizuje escape sekvencí `\r`. Pro obsluhu této logiky bylo nutné implementovat *JavaScriptovou* funkci – konkrétně plugin `barcodeScannerListener` ve frameworku *jQuery*.

Inicializace pluginu probíhá na určitém elementu (objektu) z `DOM`[8] hierarchie stránky. Ve většině případů probíhá inicializace přímo na samotném objektu `document`, jenž reprezentuje kořenový element samotné stránky. Při inicializaci lze uvést konfigurační parametry, mezi které patří parametr `debug`. Pomocí něj lze zapnout debug mód, kdy plugin v průběhu činnosti zasílá logovací zprávy do konzole prohlížeče.

Při inicializaci dojde k zavolání metody `init`, která nastaví na daném elementu naslouchání na událost `keypress`. Tato událost vzniká, jestliže je stisknuta klávesa, a tedy i zaslán znak ze čtečky. Při obsluze vzniklé události se kontroluje, zda nevznikla přímo nad

[8] Objektově orientovaná reprezentace HTML dokumentu.

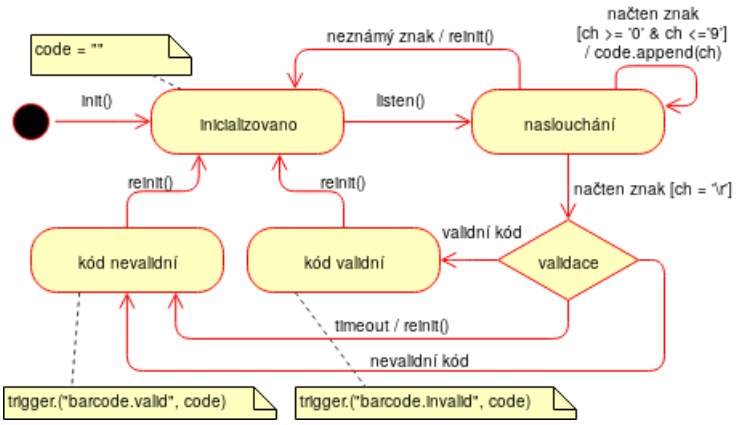
textovým polem (`input`), které nebylo specifikováno jako element. V takovém případě by to totiž bylo posuzováno jako vložení načteného kódu přímo do textového pole.

Obsluha validně zachycené události je obsloužena metodou `listen(e)`, které se přímo předá objekt události (`jQuery.Event`). V metodě dojde ke zjištění znaku a jeho validaci pomocí metody `validateKey($key)`. Na výsledek validace metoda následně zareaguje třemi možnými scénáři:

- *validní znak:* `0-9`
  Znak je dále zpracován metodou `appendChar($char)`, ve které dojde k uložení znaku a aktuálního času pro případnou detekci vypršení časového limitu.
- *validní znak:* `\r`
  Signalizován konec zasílání. Provede se validace načteného kódu metodou `validate`.
- *nevalidní znak*
  Dojde k reinicializaci knihovny pomocí metody `reinit`.

Validace načteného kódu provede ověření, zda kód splňuje náležitosti dané typem `EAN-13`[9] a zda byl dodržen časový limit, který zabraňuje situacím, kdy by se knihovna pokusila zpracovat znaky zadané přímo na klávesnici. Časový limit se vypočte jako rozdíl aktuálního času (načtení znaku `\r`) a času posledního přidaného znaku (načtení znaku `0-9`).

Po úspěšné validaci je vyvolána událost `barcode.valid` a při neúspěšné událost `barcode.invalid`. Rovněž v obou případech je parametrem předán načtený kód. Zjednodušené schéma výše uvedené logiky je znázorněno na obrázku 8.9.



Obrázek 8.9: Schéma logiky pluginu `barcodeScannerListener`

8.6 Statistiky

Část pro statistiky je určena pouze pro roli *provozovatele*. V případě potřeby zpřístupnění i pro roli *pokladního* by bylo nutné upravit záznam ve třídě `Authorizator`, jež slouží k ověření autorizace.

Rychlý přehled o stavu pokladny je k dispozici na samotném dashboardu (viz C.1), jenž je dostupný i pro roli *pokladního*. Uživatel zde nalezne sumu tržeb za aktuální den/měsíc a za předchozí den/měsíc, počet vydaných dokladů za aktuální den/měsíc a za

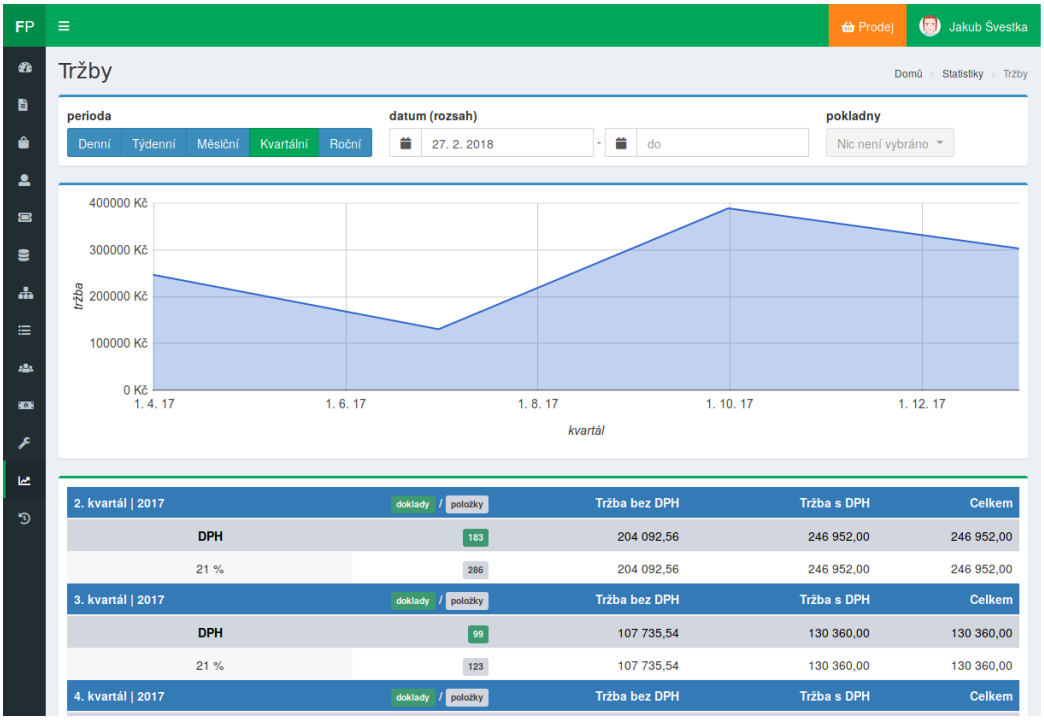
[9] Specifikace dostupná z https://en.wikipedia.org/wiki/International_Article_Number.

předchozí den/měsíc. Dále se zde nachází tabulka s posledními vydanými doklady a posledními skladovými pohyby, časová osa tiketů a počty tiketů dle jednotlivých stavů. Detailnější informace jsou dostupné v části se statistikami, jejíž funkce budou dále přiblíženy. Veškeré sekce statistik berou v potaz to, zda je provozovatel plátcem či neplátcem DPH.

8.6.1 Tržby

Slouží k zobrazení tržeb za určité období. Provozovatel si může zvolit periodu, dle které budou výsledky seskupeny. Dostupné periody jsou: *denní*, *týdenní*, *měsíční*, *kvartální* a *roční*. Ve výchozím nastavení filtru jsou výsledky zobrazeny bez rozlišení pokladen, na kterých byla tržba realizována. Pro jejich rozlišení musí provozovatel explicitně vybrat, které pokladny chce zobrazit.

Zobrazení tržeb je realizováno pomocí grafu s využitím knihovny *Google Charts*[10] a tabulky. Tržby v tabulce jsou seskupeny dle zvolené periody a případně pokladny. Každý záznam obsahuje kromě celkového součtu také detailnější pohled na součty dle jednotlivých sazeb DPH. Viz obrázek 8.10.



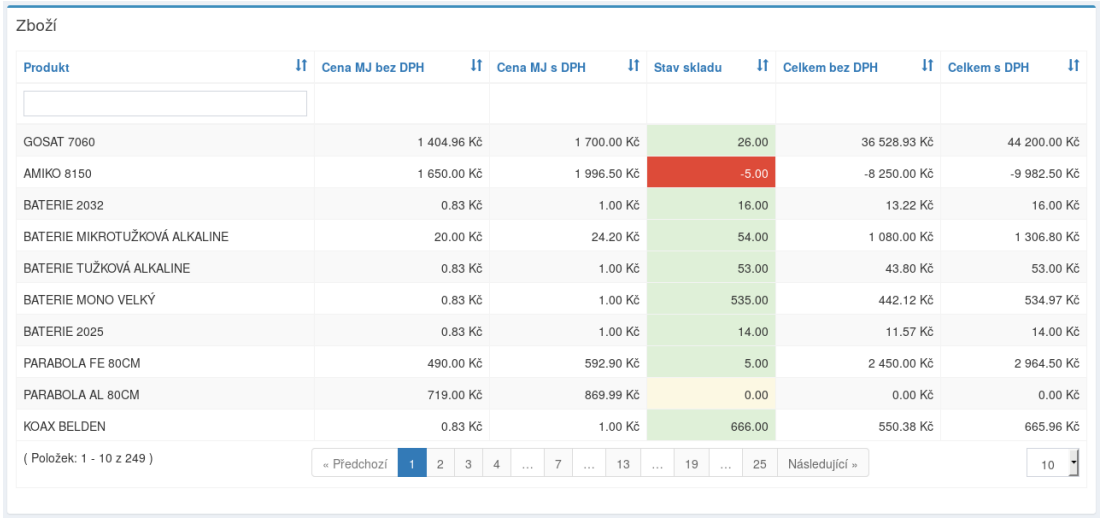
Obrázek 8.10: Statistiky – sekce tržby

8.6.2 Sklad

Zobrazuje celkovou částku (s DPH a bez DPH), kterou má provozovatel umořenou ve skladu. Dále je k dispozici tabulka s jednotlivými produkty, kde se u každého z nich nachází informace o aktuálním skladovém stavu, ceně za jednotku (s DPH a bez DPH) a umořené částce (s DPH a bez DPH). Pro lepší přehlednost je skladový stav podmíněně podbarven dle toho, zda je hodnota kladná (*zelená*), záporná (*červená*) nebo nulová (*oranžová*) (viz obrázek

[10] Knihovna je dostupná z https://developers.google.com/chart/.

8.11). V záhlaví tabulky se nachází kontrolní prvky, pomocí nichž lze výsledky filtrovat či řadit.



Obrázek 8.11: Statistiky – sekce sklad

8.6.3 Zboží

Zobrazuje celkový počet položek v katalogu a dále tabulku s prodanými produkty. U každého produktu je uveden počet prodaných jednotek a celková tržba. V záhlaví tabulky se nachází kontrolní prvky, pomocí nichž lze výsledky filtrovat či řadit.

8.6.4 Tikety

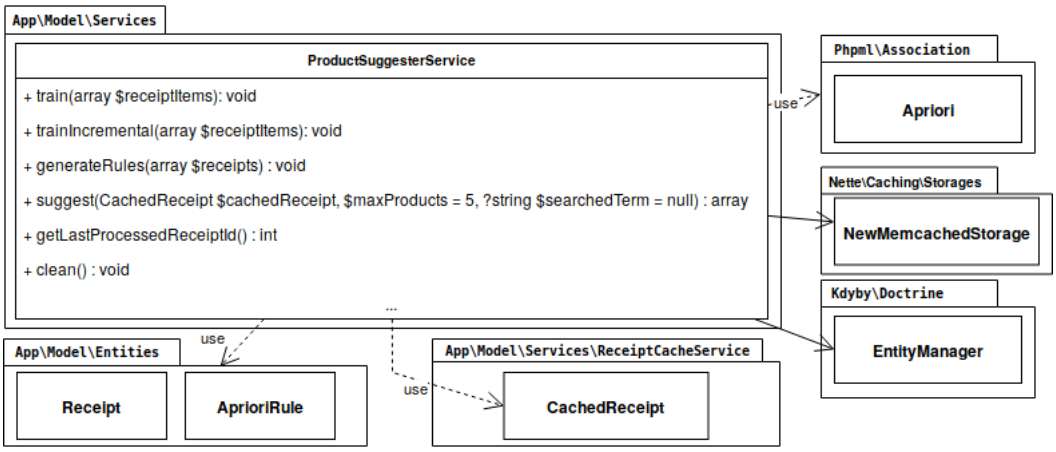
Rychlý přehled o tiketovacím systému, kde provozovatel nalezne počty tiketů dle jednotlivých stavů a graf s počtem tiketů za měsíční periodu. Měsíc a rok lze vybrat v záhlaví boxu, v němž je graf umístěn.

8.7 Našeptávač produktů

Veškerá logika našeptávání produktů je zajištěna službou `ProductSuggesterService`, jejíž část rozhraní je znázorněna na obrázku 8.12. Služba zajišťuje dvě hlavní funkce, a to **získávání asociačních pravidel z již vydaných dokladů** a samotné **našeptávání produktů**. Získávání asociačních pravidel je realizováno s využitím algoritmu *Apriori*, který byl již popsán v návrhu 7.3.2. Jeho implementace byla použita z knihovny `PHP-ML`[11], která nabízí velkou škálu algoritmů pro strojové učení.

Službu primárně využívají komponenty pro výběr zboží při markování účtu, jež byly popsány v části 4.1.2. Nicméně její využití by mohlo být daleko rozsáhlejší – např. pro marketingové účely.

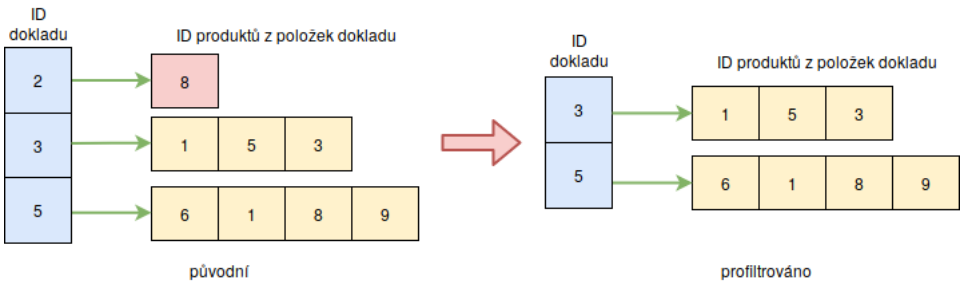
[11] Machine Learning library for PHP, dostupná z: https://github.com/php-ai/php-ml/.



Obrázek 8.12: Důležité rozhraní služby `ProductSuggesterService` a její závislosti

8.7.1 Získávání asociačních pravidel

Při získávání asociačních pravidel dochází k hledání vztahů mezi prodanými produkty. K tomuto účelu je určena metoda `train(array $receiptItems)`, jež v parametru očekává vstupní data. Vstupními daty se rozumí pole entit `ReceiptItem`, což jsou ve skutečnosti položky z již realizovaných dokladů (několika). Z těchto položek je následně vytvořeno dvourozměrné pole transakcí, kde klíč první dimenze odpovídá identifikátoru dokladu a její hodnota obsahuje pole s identifikátory produktů, jež figurují na daném dokladu (tzn. položky z dokladů jsou seskupeny po dokladech). V případě, že se jednalo o manuálně vytvořenou položku (tzn. bez vazby na produkt z katalogu), je tato položka ignorována. Pole transakcí vzniklé po této transformaci je znázorněno na obrázku 8.13, vlevo.



Obrázek 8.13: Dvourozměrné pole dokladů a jejich položky

Pole transakcí je dále profiltrováno od dokladů, které obsahují pouze jednu položku. Takové doklady jsou pro zpracovaní algoritmem bezpředmětné a zbytečně by zpracování zpomalovaly. Výsledek filtrace je znázorněn na obrázku 8.13 vpravo.

Následně je zavolána privátní metoda `generateRules(array $receipts)`, která parametrem převezme pole transakcí a provede instanciaci algoritmu *Apriori* z knihovny `Phpml\Association`. Po instanciaci je algoritmu předáno pole transakcí a explicitně vynuceno vygenerování asociačních pravidel. Po tomto kroku dojde k uložení pravidel do databáze pomocí entity `AprioriRule` (dříve uložená pravidla jsou smazána). Entita obsahuje hodnoty antecedentu a konsekventu pravidla. Tyto hodnoty obsahují pole identifikátorů produktů, které jsou oddělené symbolem „`,`“ a v případě antecedentu jsou vzestupně se-

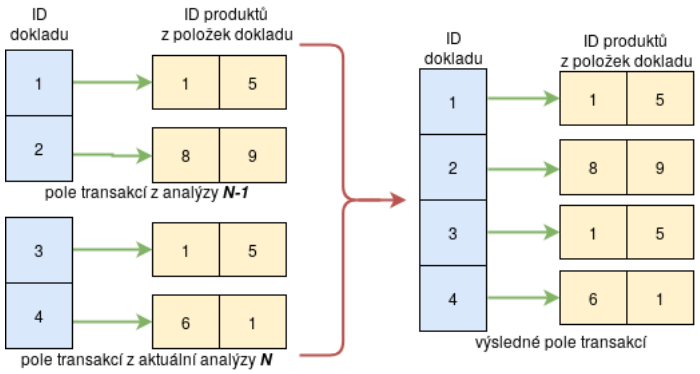
řazeny. Seřazení je důležité pro následné vyhledávání, kdy se nad hodnotami antecedentu vyhledává pole identifikátorů jako řetězec.

Návrh entity `AprioriRule` porušuje `1NF`, jelikož atributy výsledné tabulky v databázovém systému neobsahují pouze atomické hodnoty. Nicméně dekompozice entity by byla v tomto případě bezúčelná a dále by mohlo dojít ke zvýšení časové náročnosti na vkládání záznamu a na samotné vyhledávání záznamů (což je nežádoucí).

Optimalizace

Získávání asociačních pravidel je samo o sobě časově a výpočetně náročné. V rámci optimalizace byla implementace rozšířena o možnost inkrementální analýzy dokladů, která využívá skutečnosti, že nemůže dojít ke změně položek na již analyzovaných dokladech. Díky tomu stačí pole transakcí rozšiřovat pouze o transakce, jež vzniknou z nových (dosud nezpracovaných) dokladů.

K tomuto účelu slouží metoda `trainIncremental(array $receiptItems)`, která po vytvoření pole transakcí (shodně jako metoda `train(...)`) provede jeho sloučení s polem transakcí, jež se nachází v keš paměti[12] z minulé (inkrementální) analýzy. Pole vzniklé sloučením je opět uloženo do keš paměti pro budoucí analýzu. Služba si rovněž zapamatuje maximální `ID` dokladu, jež má ve vstupních datech. Tato informace je využita pro automatickou analýzu dokladů – viz dále.



Obrázek 8.14: Inkrementální analýza

Automatická analýza dokladů

Získané znalosti z dokladů je nutné udržovat aktuální – čím více dokladů je analyzováno, tím lepší výsledky může našeptávač nabídnout. Pro zautomatizování procesu je určen konzolový příkaz `cash:initProductSuggester`, umístěný v adresáři `app\model\commands`. Příkaz využívá optimalizovanou variantu analýzy, kdy se nejprve služby dotáže, jaké poslední `ID` dokladu zpracovala (označme jej `lastID`). Na základě něj jsou z databáze vybrány položky dokladu (entity `ReceiptItem`), jež náleží dokladům, jejichž `ID` je větší než `lastID`. Tímto probíhá získávání pouze nových dokladů z databáze. Příkaz se spouští v určitých časových intervalech pomocí plánovače úloh (*Cron*).

Spuštění příkazu v kořenovém adresáři aplikace:

```
$ php www/index.php cash:initProductSuggester [-f]
```

[12] Pro keš paměť byl zvolen kešovací systém *Memcached*, https://memcached.org/.

Příklad výstupu:

```
Processing receipt with ID from 42 to 80
Loaded: 67 items
```

Příkaz obsahuje jeden volitelný parametr `-f` (`--force`), pomocí něhož lze vynutit úplnou inicializaci, při které dojde ke smazání keš paměti a provede se zpracování všech dokladů.

Metriky asociačních pravidel

Při instanciaci algoritmu `Apriori` je nutné specifikovat hodnotu podpory a spolehlivosti. V konfiguračním souboru `app/config/config.neon` se definuje hodnota minimální podpory (proměnná `minSupport`) a hodnota minimální spolehlivosti (proměnná `minConfidence`). Hodnota minimální podpory je dynamicky přepočítána dle definované exponenciální funkce následovně:

$$minSupport(X) = e^{-aX-b} + c$$

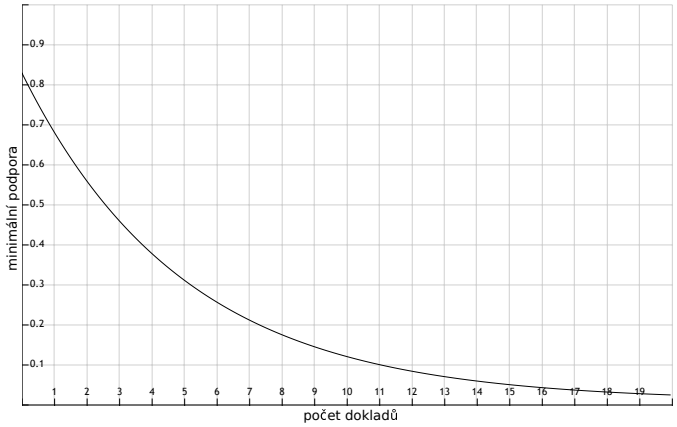
Proměnná `X` označuje počet transakcí a význam konstant je následující:

- `c` – hodnota minimální podpory, tzn. nejmenší hodnota, kterou funkce *minSupport* může vrátit
- `a`, `b` – ovlivňují, jak rychle křivka klesá, když se proměnná `X` (počet transakcí) zvýší.

Dynamický výpočet podpory byl převzat z článku *„How to auto-adjust the minimum support threshold according to the data size“* [22].

Zvolené hodnoty

Proměnná `X` odpovídá počtu dokladů ve vstupních datech, tedy $X = count(\$receipts)$. Hodnota minimální podpory byla nastavena na `0.01` (konstanta `c`) a minimální spolehlivost na `0.05`. Konstanty `a` a `b` byly zvoleny na základě několika pokusů o vykreslení funkce s různými hodnotami konstant. Nakonec byly zvoleny hodnoty $a = 0.2$ a $b = 0.2$. Hodnoty byly zvoleny s tím cílem, aby nastavená minimální podpora byla platná až od 20 dokladů – viz graf na obrázku 8.15. Z důvodu malého vzorku reálných dat pro testovaní (266 dokladů po vyfiltrování) se ukáže až při větším množství dokladů, zda byly hodnoty vhodně zvoleny.



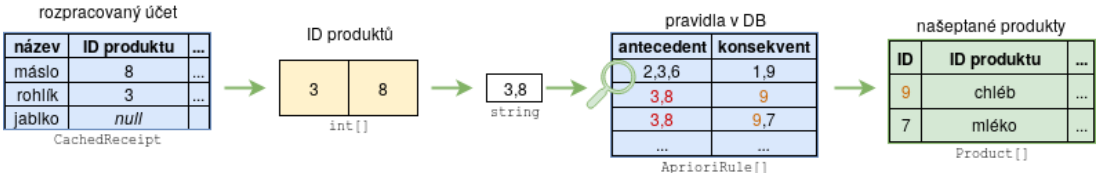
Obrázek 8.15: Závislost minimální podpory na počtu dokladů

8.7.2 Našeptávání produktů

K našeptání produktů slouží metoda `suggest(...)`, která přes parametry očekává:

- rozpracovaný účet, na základě jehož položek je našeptávání provedeno
- maximální počet vrácených produktů – volitelné (výchozí hodnota `5`)
- hledaný řetězec, který se musí nacházet v atributech produktu (název, `EAN` kód, ...) – volitelné.

V těle metody dojde k načtení `ID` produktů, které se nachází na předaném (rozpracovaném) účtu. Pole s identifikátory produktů je vzestupně seřazeno a převedeno do textového řetězce, kde identifikátory jsou oddělené znakem „,“. Na základě tohoto řetězce jsou vyhledány v databázi pravidla (entity `AprioriRule`), jež obsahují tento řetězec antecedentu. Z těchto pravidel jsou vybrány konsekventy, které obsahují `ID` produktů. S konsekventy se provede sloučení do jednoho pole, jež ve výsledku obsahuje `ID` produktů pro našeptání. Nicméně z hodnot několika konsekventů můžeme dostat shodné `ID` produktu, což pak ve výsledném poli způsobí duplicity. Před samotným odstraněním duplicit se nejprve provede zjištění četnosti jednotlivých hodnot v poli a na základě ní jsou položky seřazeny. Tímto je docíleno toho, že produkty s největší četností budou našeptány jako první. Popisovaný proces je znázorněn na obrázku 8.16.



Obrázek 8.16: Schéma našeptání produktů k rozpracovanému dokladu

Pokud není našeptávačem nalezen žádný produkt pro daný vzorek, je pokus o našeptání proveden pouze pro posledně vložený produkt.

8.8 EET – elektronická evidence tržeb

Jelikož povinnost zasílat tržby do `EET` neplatí pro všechny podnikatele, byla tato funkcionalita implementována jako volitelná. Díky tomu může pokladnu využívat širší okruh podnikatelů bez nutnosti zasílat tržby do `EET`.

8.8.1 Nastavení

Nastavení `EET` se nachází v administrační části v sekci *nastavení* → *EET* a je realizováno pomocí komponenty `EditEetSettings`. Komponenta obsahuje formulář, v kterém si provozovatel může vybrat, zda je funkcionalita `EET` povolena. Pro povolení je požadováno vyplnění údajů, mezi které patří: *DIČ* poplatníka, výchozí číslo provozovny, výchozí číslo pokladny, a pokud již dříve nebyl nahrán certifikát, tak i certifikát ve formátu *PKCS #12* a heslo k certifikátu. V opačném případě jsou tyto položky volitelné. Formát certifikátu *PKCS #12* je tzv. archiv, který obsahuje uložený binární certifikát a k němu odpovídající privátní klíč. Certifikát v tomto formátu lze získat přímo z portálu *Finanční správy*.

Před uložením je nejprve zavolána metoda `certificateValidate(BaseForm $form)` pro validaci certifikátu. Validace je vykonána pouze v případě, že byl certifikát vybrán. Validace má za cíl ověřit, zda zadané heslo je platné pro nahraný certifikát. K tomuto ověření je využita funkce `openssl_pkcs12_read` z `PHP` knihovny *OpenSSL*, která slouží k rozbalení certifikátu na samotný certifikát a privátní klíč. Návratová hodnota této funkce vypovídá o tom, zda se rozbalení povedlo, či ne. Tzn. pokud rozbalení neproběhlo korektně, je zadané heslo vyhodnoceno jako nesprávné. Tímto nedojde k uložení dat a rovněž nedojde k samotnému povolení funkcionality `EET`.

Jestliže odeslaný formulář projde veškerou validací, metoda `save(BaseForm $form)` zajistí uložení zadaných hodnot do databáze. Nastavení `EET` je reprezentovanou entitou `SettingsEet`. Z důvodu bezpečnosti není certifikát ukládán do databáze, ale přímo do souborového systému. Certifikát *PKCS #12* je pomocí zadaného hesla nejprve rozbalen na certifikát `X.509` a privátní klíč. K uložení slouží služba `EetCertificateStorage`, která obsahuje veřejnou metodu `save(string $cert, string $privateKey)`. Po jejím zavolaní dojde k uložení certifikátu a privátního klíče do adresáře `data/eetCertificate`. Adresář není veřejně přístupný skrze webový server. Pokud se v adresáři již nacházel certifikát/klíč, dojde k jeho přepsání.

Pro ověření či testovací provoz pokladny lze dále v nastavení zapnout:

- **Ověřovací mód**
  V případě zapnutého ověřovacího módu nejsou odeslané záznamy považovány za skutečné tržby. Zaslané tržby slouží pouze pro ověření, zda se tržby evidují korektně do portálu *Finanční správy*. Po odeslání tržby nedojde k obdržení `FIK` kódu.

- **Neprodukční prostředí** (též označováno jako *playground*)
  Neprodukční prostředí slouží pouze k testovacím účelům pokladny, tedy nikoli koncovým uživatelům pokladny. Pro jeho využívání je nutné nahrát speciální testovací certifikát, který byl pro toto prostředí veřejně vydán přímo *Finanční správou* a dále zadat údaje o poplatníkovi, pro kterého byl testovací certifikát vydán[13]. Po zaslání datové zprávy s tržbou do neprodukčního prostředí dojde k vrácení `FIK` kódu, který není platným fiskálním identifikačním kódem.

Test spojení

Jedná se o další funkcionalitu výše zmíněné komponenty, která slouží pro ověření uloženého nastavení. Test spojení je realizován pomocí metody `testConnection()` ze služby `EETService` (viz 8.8.3). V případě chybného nastavení je uživatel informován chybovou zprávou, která rovněž blíže specifikuje chybně nastavený údaj.

8.8.2 Uložení `EET` dat týkající se dokladu

Jak již bylo řečeno výše, funkcionalita `EET` je volitelná. Proto při návrhu entity `Receipt`, reprezentující doklad, byly atributy týkající se `EET` odděleny do samostatné entity `ReceiptEetData`. Asociace těchto entit je znázorněna na obrázku 8.18 a sémantika atributů je následující:

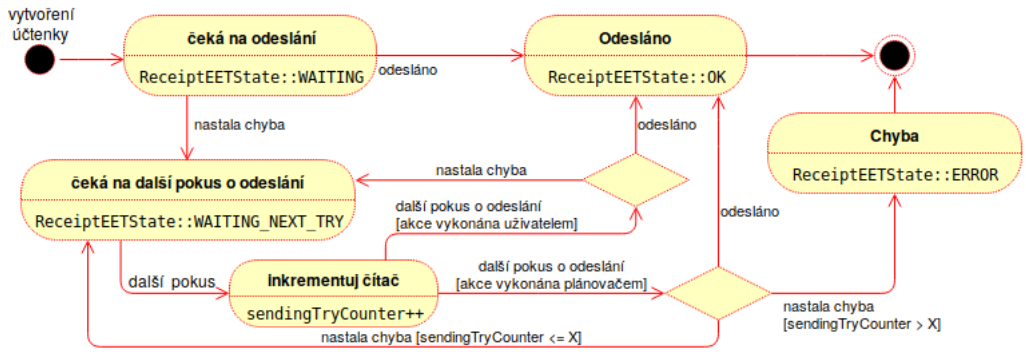
- `sendingTryCounter` – čítač pokusů o zaslání tržby

[13] Testovací certifikát a údaje o poplatníkovi jsou dostupné z http://www.etrzby.cz/cs/technicka-specifikace.

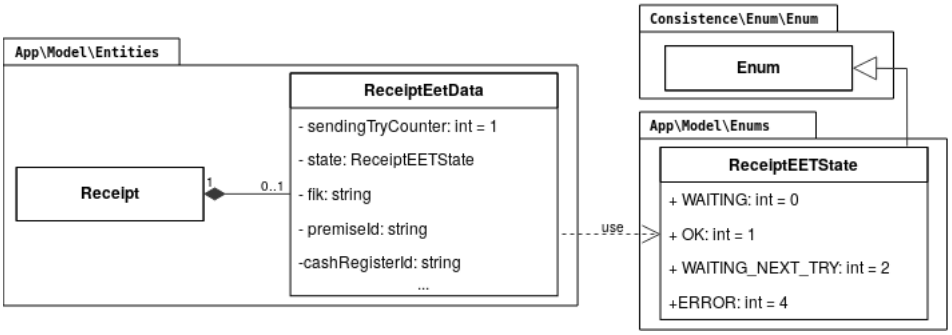
- `state` – stav zaevidování tržby (viz níže)
- `fik` – FIK kód, který se získá po úspěšné evidenci tržby
- `premiseId` – označení provozovny
- `cashRegisterId` – označení pokladního zařízení.

Možné stavy při evidenci tržby a přechody mezi nimi

Logika přechodů mezi stavy je znázorněna pomocí stavového diagramu na obrázku 8.17.



Obrázek 8.17: Stavový diagram znázorňující změnu stavu při evidenci dokladu do `EET`



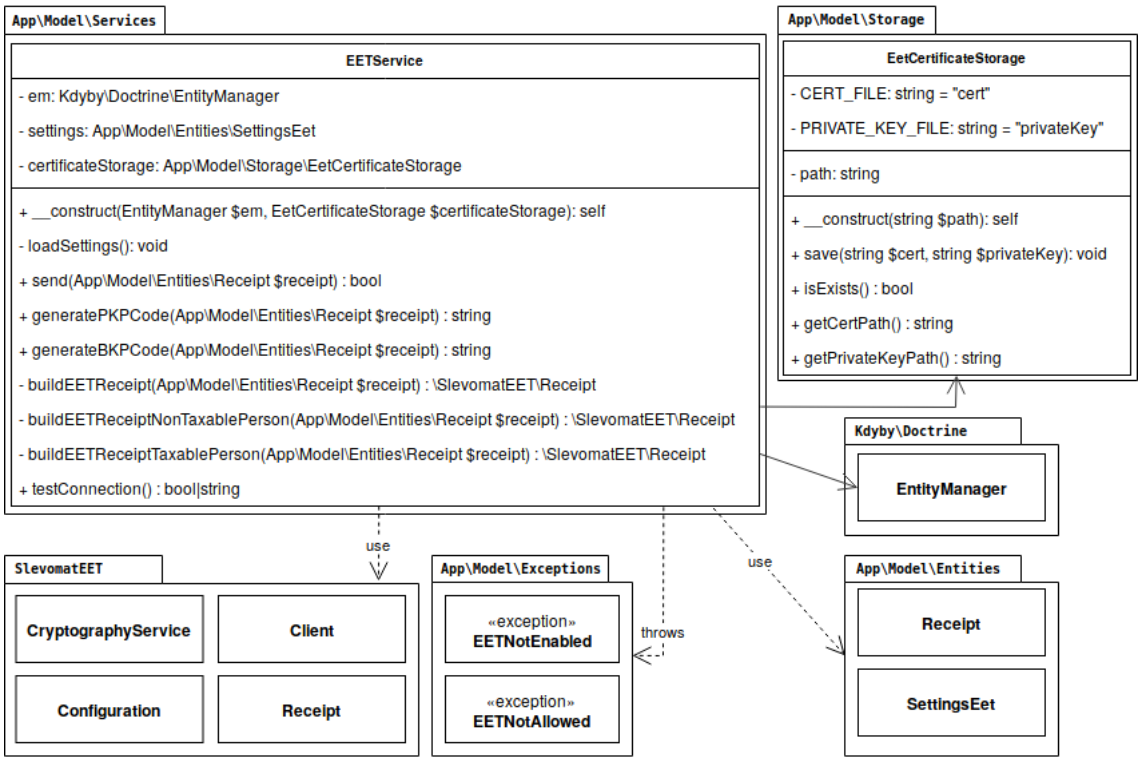
Obrázek 8.18: Diagram tříd entity `ReceiptEetData` a její závislosti

8.8.3 Zasílání tržeb

Implementace zasílání tržeb do `EET` musí být provedena v souladu s technickou specifikací vydanou *Finanční správou*, jež byla zmíněna v kapitole 3. V současné době již existuje řada knihoven, které na základě technické specifikace implementují logiku komunikace s portálem `EET`. Programátor je tak odstíněn od implementační logiky komunikace a využívá pouze jednoduché rozhraní dané knihovny. Pro implementaci byla zvolena knihovna `slevomat/eet-client`[14] z důvodu kvalitní dokumentace a hlavně pozitivních zkušeností s jejím použitím v první verzi již implementované pokladny.

[14]Knihovna je dostupná pod MIT licencí z https://github.com/slevomat/eet-client.

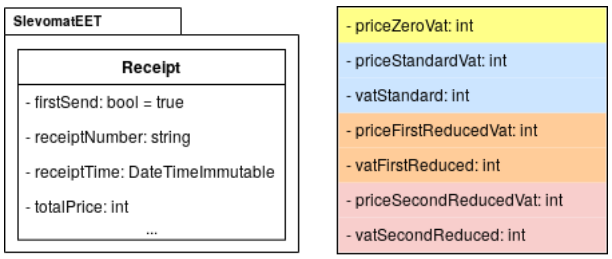
Komunikace s knihovnou, a tedy i s portálem *Finanční správy*, probíhá výhradně skrze implementovanou službu `EETService`, jejíž diagram je znázorněn zjednodušeným UML diagramem tříd na obrázku 8.19. Služba si při své instanciaci v `DI` kontejneru vyžádá službu `EetCertificateStorage`, která zpřístupní úložiště s certifikátem a privátním klíčem. Dále v konstruktoru voláním privátní metody `loadSettings()` dojde k načtení aktuálního nastavení `EET` (entita `SettingsEet`). Z důvodu důležitosti této služby budou dále blíže popsány metody, které implementuje.



Obrázek 8.19: Diagram tříd služby `EETService` a její závislosti

Metoda `buildEETReceipt(Receipt $receipt) : \SlevomatEET\Receipt`

Entita `Receipt` odpovídá dokladu/účtence, jež je uložena v databázi. Aby bylo možné pracovat s účtenkou pomocí využité knihovny, bylo nutné vytvořit konverzní metodu, která z entity `Receipt` vytvoří objekt instance třídy `SlevomatEET\Receipt`, jehož část atributů je znázorněna na obrázku 8.20.



Obrázek 8.20: Část atributů třídy `SlevomatEET\Receipt`

K tomuto účelu byla implementovaná výše uvedaná metoda, která z předané instance objektu `Receipt` převezme:

- číslo účtenky
  `receiptNumber = $receipt->getNiceId()`
- příznak, zda se jedná o první zaslání
  `firstSend = $receipt->getReceiptEetData()->isFirstTry()`
- datum vytvoření
  `receiptTime = $receipt->getCreationDate()`
- celkovou tržbu
  `totalPrice = $receipt->calculateReceiptSum() * 100`

Objekt `SlevomatEET\Receipt` vyžaduje veškeré zadávané částky včetně setin měny, a to uložené pomocí datového typu `integer`. Z tohoto důvodu je nutné částky datového typu `float` vynásobit konstantou *100* a přetypovat na datový typ `integer`[15].

V případě, že je poplatník plátce `DPH`[16], je dále nutné celkovou hodnotu tržby specifikovat do jednotlivých druhů sazeb `DPH`. K tomuto slouží atributy, které jsou znázorněny na obrázku 8.20 vpravo. Jednotlivé atributy pro daný druh sazby jsou pro lepší přehlednost barevně rozlišeny.

Metoda `send(Receipt $receipt) : bool`

Slouží k odeslání tržby do `EET`. Ihned po zavolání dojde k ověření, zda je `EET` funkcionalita zapnuta. Tímto ověřením je aplikačně garantováno, že nastavení `EET` se nachází v konzistentním stavu a rovněž nemůže dojít k případu, kdy by se pokladna pokusila o evidenci tržby, i když to poplatníkem nebylo chtěné – v tomto případě by došlo k vyhození výjimky `EETNotEnabled`. Dále je hlídáno, zda parametrem předaný objekt může být vůbec zpracován a považován za tržbu. Jak již bylo řečeno v části 8.2.4, v systému se vyskytuje několik typů dokladů a některé se za tržbu nepovažují – např. zálohová faktura. K tomuto ověření slouží statická metoda `isPossibleSendToEET(Receipt $receipt)` umístěná ve třídě `App\Model\Helpers\ReceiptHelper`. Při pokusu o zaslání tržby z dokladu nepovoleného typu, dojde k vyhození výjimky `EETNotAllowed`.

Po průchodu výše uvedenými validacemi může dojít k inicializaci knihovny pro komunikaci s `EET`, konkrétně:

- vytvoření šifrovací služby `SlevomatEET\Cryptography\CryptographyService`, které se předá certifikát, heslo k privátnímu klíči a samotný privátní klíč
- vytvoření konfigurace prostředí `SlevomatEET\Configuration`
- a vytvoření objektu klienta `SlevomatEET\Client`.

Před samotným odesláním tržby je nutné provést konverzi objektu entity `Receipt` (viz výše). Poté dojde k odeslání tržby metodou `send(SlevomatEET\Receipt $receipt)` v objektu klienta `SlevomatEET\Client`.

Pokud proběhne odeslání tržby bez chyb, provede se uložení `FIK` kódu a nastavení stavu `ReceiptEETState::OK` do dané entity `Receipt`. V případě chyby je nastaven stav

[15] Např. pokud odesíláme tržbu *55.52* Kč, předáme do objektu hodnotu *5552* ($55.52 * 100 = 5552$).
[16] Daň z přidané hodnoty.

`ReceiptEETState::WAITING_NEXT_TRY`, inkrementován čítač pokusu o zaslání tržby a opětovný pokus bude opakován pomocí příkazu, který je spouštěn plánovačem úloh (viz 8.8.4) nebo manuální vynucením.

Metoda `generatePKPCode(Receipt $receipt) : string`

Pomocná metoda, která slouží k vygenerování `PKP`[17] kódu, který musí být součástí účtenky v případě, že se nepovedlo tržbu zaevidovat. Tento kód si lze představit jako zjednodušený elektronický podpis.

Metoda `generateBKPCode(Receipt $receipt) : string`

Pomocná metoda, která slouží k vygenerování `BKP`[18] kódu, který je vždy součástí účtenky. Tento kód jednoznačně identifikuje příslušnou tržbu poplatníka. Z implementačního pohledu se jedná o otisk výše zmíněného kódu `PKP`, tedy zjednodušeně zapsáno `PKP = hash(BKP)`.

Metoda `testConnection() : bool|string`

Slouží pouze k ověření, zda provozovatel provedl nastavení `EET` validně. Logika ověření spočívá v zaslání uměle vytvořené účtenky do `EET`, jejíž tržba je nulová. V případě, že se odeslání nepovedlo či portál *Finanční správy* vrátil odpověď s chybou/upozorněním, je tato chyba/upozornění předáno návratovou hodnotou (datový typ `string`). Při úspěšném zaslání je vrácena hodnota `true` (datový typ `bool`).

8.8.4 Automatické zasílání tržeb

Pokladna se automaticky a autonomně pokouší o zaslání tržeb, u kterých se zaslání na první pokus nepovedlo. K tomuto účelu byl implementován konzolový příkaz `SendReceiptsToEETCommand`, umístěný v adresáři `app\model\commands`. Příkaz se spouští v určitých časových intervalech pomocí plánovače úloh (*Cron*).

Po spuštění příkazu dojde k získání všech účtenek (entit `Receipt`), jež mají v objektu `ReceiptEetData` nastaven stav `ReceiptEETState::WAITING_NEXT_TRY`. Následně jsou jednotlivě účtenky předány službě `EETService` metodou `send(...)` pro další pokus o zaslání.

Aby se předešlo situaci, kdy pokladna bude permanentně zasílat tržbu i přes několik desítek neúspěchů, byl implementován ochranný mechanismus, který v případě, že je počet pokusů > 20, označí účtenku v objektu `ReceiptEetData` stavem `ReceiptEETState::ERROR`. Tímto se aplikace již nadále nebudou pokoušet tržbu automaticky odeslat a bude nutné ji odeslat na základě vnějšího podnětu od uživatele.

Spuštění příkazu v kořenovém adresáři aplikace:

```
$ php www/index.php cash:sendReceiptsToEET
```

[17] Podpisový kód poplatníka.
[18] Bezpečnostní kód poplatníka.

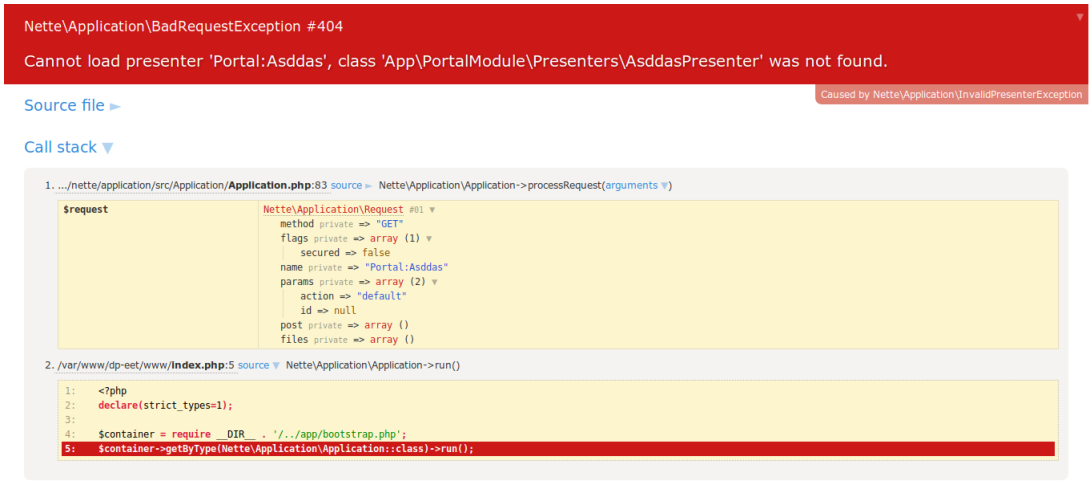
Příklad výstupu:

```
Count of receipts waiting for sending to EET: 2
+-------------------+--------+--------------------------------------+
| ID                | Result | FIK                                  |
+-------------------+--------+--------------------------------------+
| 1/2018/03/1052    | OK     | 81bacfa5-e795-4300-a47e-f7e659cc7fe3-ff |
| 1/2018/03/1053    | ERROR  |                                      |
+-------------------+--------+--------------------------------------+
```

Kapitola 9

Testování

Testování a ladění aplikace probíhalo již v průběhu implementace. Samotné PHP neobsahuje nástroj pro ladění a zpracování chyb, pouze nás upozorní na to, o jakou chybu se jedná a na jakém řádku k ní došlo. Proto *Nette* obsahuje knihovnu *Tracy*[1] (známá též pod názvem *Laděnka*) určenou k tomuto účelu. Při výskytu chyby dojde k zobrazení chybové stránky, která obsahuje úryvek z kódu se zvýrazněným řádkem, který chybu vyvolal (viz obrázek 9.1). Chybové stránce dominuje velmi užitečná komponenta, jež zobrazuje zásobník volání (call stack). Pomocí ní lze procházet metody v pořadí, v jakém byly volány včetně zobrazení hodnot předaných argumenty. Dále zde nalezneme veškeré hodnoty systémových proměnných jak PHP, tak i prostředí frameworku, parametry HTTP požadavku, HTTP hlavičky zaslané odpovědi...

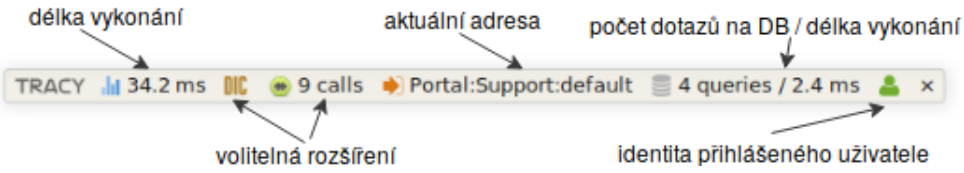


Obrázek 9.1: Chybová stránka vygenerovaná knihovnou *Tracy*

Pro ladění a získání informací o aktuálním chodu aplikace je určen ladící panel (*Tracy* debugger bar). Ten je zobrazen po každém HTTP požadavku včetně požadavků vykonaných asynchronně skrze Ajax. Pomocí něj zjistíme celkový čas zpracování dotazu, jenž je užitečný pro kontrolu optimalizace aplikace. Mezi další informace patří například aktuální adresa, počet dotazů na databázi včetně celkové délky vykonání dotazů apod. Po kliknutí na ja-

[1] Dostupná z https://tracy.nette.org/.

koukoliv položku se zobrazí rozšířené informace – např. po kliknutí na položku týkající se dotazů na databázi dojde k zobrazení okna s jednotlivými dotazy a časy vykonání.



Obrázek 9.2: *Tracy* debugger bar

9.1 Testování pokladny

Testování pokladny probíhalo probíhalo jeden týden v reálném provozu, během kterého bylo vydáno 78 dokladů (z toho 4 storna). Provozovateli přišla velmi užitečná funkce našeptávání, možnost provozovat více pokladen v rámci jedné instance aplikace a vícero možností pro předání dokladu. Ze strany provozovatele byly rovněž vzneseny následující návrhy na zlepšení.

Řazení položek v katalogu zboží

problém:
Položky v katalogu zboží (produkty a kategorie) nebyly abecedně řazeny. Provozovatel měl problém se v katalogu orientovat a vyhledání položky mu trvalo výrazně déle.

řešení:
Jednalo se o implementační chybu, kdy v `SQL` dotazu byla opomenuta klauzule pro řazení. Chyba byla operativně opravena před samotným testovacím obdobím.

Pojmenování účtu v manuálním režimu

problém:
Provozovateli chyběla možnost pojmenovat účet.

řešení:
Takto funkcionalita byla v pokladně k dispozici, ale byla opomenuta v manuálním režimu. Pokladní tak neviděl název účtu a nebylo jej možné přejmenovat. Funkcionalita byla dodělána i pro manuální režim.

9.2 Testování portálu pro zákazníky

Testování bylo provedeno jednotlivě na vzorku 10 lidí. Pro tento účel byl vytvořen testovací manuál, dle kterého účastníci testování vykonávali úkoly a odpovídali na otázky. Účastníci byli rovněž při práci pozorováni, abych zjistil, zda je uživatelské rozhraní dostatečně intuitivní. Manuál k testování je k dispozici v příloze E. Úkolem testování bylo ověřit, zda jsou základní funkcionality portálu pro účastníky srozumitelné a intuitivní.

Výsledky dotazníku jsou k dispozici v příloze E. Při testování vzešlo ze strany účastníků několik návrhů na zlepšení, některé z nich budou dále zmíněny.

Založení tiketu – výběr dokladu

problém:
Výběr dokladů byl realizován pomocí výběru ze seznamu dokladů. Od dvou účastníku přišel dotaz na to, zda nelze v seznamu nějak vyhledávat, jelikož v případě desítek dokladů se stane seznam nepřehledný.

řešení:
Na základě dotazu jsem výběr dokladu opatřil o možnost fulltextového vyhledávání dle čísla dokladu.

Zaslání dokladu na email

problém:
Pro zaslání dokladu na email bylo nutné vždy vyplnit emailovou adresu. Od jednoho účastníka vzešel návrh na předvyplnění emailové adresy, která byla již zadána při registraci.

řešení:
Návrh jsem akceptoval. Předvyplnění je implementačně snadné a zákazníkovi značně urychlí zaslání dokladu na vlastní emailovou adresu.

Webový doklad

problém:
Účastník zachytil v úvodním představení zmínku o webovém dokladu a byl od něj vznesen dotaz, jak tato funkcionalita funguje a jak lze odkaz na doklad získat.

řešení:
Funkcionalita byla účastníkovi objasněna a vysvětleno, že odkaz na doklad je uveden na staženém dokladu ve formátu `PDF`. Po delší diskusi jsme dospěli k závěru, že by bylo vhodné funkcionalitu pro získání odkazu doplnit do samotného detailu s dokladem. V detailu dokladu bylo dodělané tlačítko pro zobrazení modálního okna s odkazem.

Kapitola 10

Závěr

Cílem práce bylo navrhnout a implementovat systém pro prodej zboží s podporou evidence tržeb do EET, evidencí skladu a portálem pro zákazníky. Návrh systému vychází z nedostatků již implementované verze pokladny, která není natolik univerzální, aby se mohla použít u větší škály prodejců. Při návrhu byl kladen důraz na jednoduchost a univerzálnost rozhraní pokladny. Průzkum a výběr vhodných technologií byl soustředěn na ty, které jsou běžně dostupné u poskytovatelů webhostingů a jsou poskytovány zdarma.

Dle analýzy dostupných pokladen doposud neexistuje žádné řešení, které by bylo poskytnuto jako opensource. Všechna existující bezplatná řešení jsou poskytovaná jako služba, kdy uživatel nemá přehled o tom, co se děje s jeho daty a je plně závislý na poskytovateli služby. Cílem návrhu a implementace je vytvořit funkční prototyp pokladního řešení, který bude možné nasadit do ostrého provozu ve firmě rodinného příslušníka a poskytnout ho jako opensource řešení. Prioritou navrženého systému je snížit podnikateli náklady na pořízení potřebného hardwaru a na samotný provoz.

Pro implementaci byly zvoleny moderní technologie, mezi které patří *Nette* framework postavený na PHP 7.1, ORM framework *Doctrine* a JavaScriptová knihovna *jQuery*. Grafický vzhled aplikace byl kompletně postaven na CSS frameworku *Bootstrap*. Pro pokladní část byla použita šablona *AdminLTE*, která tvoří nadstavbu nad *Bootstrapem*. Všechny použité externí knihovny jsou spravované pomocí balíčkovacích systémů *Composer*, *Bower* a *npm*. Díky nim je zajištěna jednoduchá správa a instalace, kdy dojde k automatickému stažení všech závislostí pomocí několika příkazů.

Vytvořená aplikace splňuje stanovené požadavky. Aplikace byla vytvořena natolik konfigurovatelně, že nemusí být použita pouze jako EET pokladna. Funkcionalita EET není povinná a lze ji kdykoliv v nastavení zakázat. Díky tomu může pokladnu využívat širší okruh podnikatelů bez nutnosti zasílat tržby do EET. Aplikace plně rozlišuje, zda je, nebo není provozovatel plátcem DPH. Na základě této skutečnosti dojde k přizpůsobení uživatelského rozhraní a výsledných dokladů, aby splňovaly veškeré náležitosti dané zákonem.

Pokladní systém podporuje práci se čtečkou čárových kódů, jež je u těchto systémů standardně používána a pokladnímu urychlí proces tvorby účtu. Pro tvorbu účtu byly implementované tři režimy pohledů, kde každý z nich najde uplatnění u různých typů podnikatelů. Volba režimu pohledu záleží plně na pokladním a může být kdykoliv změněna. Velký potenciál má pokladní systém v našeptávači, který na základě analýzy již vydaných dokladů našeptává pokladnímu produkty, jež by mohly být dále přidány na účet.

Mezi nejobtížnější části návrhu patřilo přiřazení funkcionalit a zvolení vhodného rozložení rozhraní jednotlivých částí systému, které vznikly po dekompozici na základě analýzy požadavků. Dále bylo nutné studium textů, jenž se věnovaly EET, zákonným požadavků na

zaokrouhlování dokladů a samotným náležitostem, jež musí doklad splňovat. Velkou výzvou byl našeptávač zboží, u kterého bylo nutné provést implementaci co nejefektivněji, aby byla latence při našeptávání co nejmenší.

Možnosti dalšího rozvoje

Mapa stolů

Jednalo by se o další (čtvrtý) režim pohledu, který by byl určen pro restaurační provozy. Pokladní by zde měl zobrazenou mapu stolů, kdy po kliknutí na daný stůl by došlo k zobrazení otevřených účtů na daném stole.

Rozdělení účtu

V případě režimu mapy stolů by bylo nutné dále dodělat možnost rozúčtování účtů na několik separátních účtů (např. u stolu sedí několik osob a každý požaduje platit určité položky).

Využití našeptávače pro marketingové účely

Pomocí našeptávače by mohlo docházet k analýze nákupů konkrétního zákazníka a cílit na něj např. slevové kupóny na určitý sortiment zboží.

Věrnostní program

U každého produktu by bylo uvedeno, kolik věrnostního bodů bude za nákup obdrženo či strženo z účtu zákazníka. Zákazník by v portálu pro zákazníky viděl stav a historii čerpání věrnostních bodů.

`API` rozhraní

Pomocí `API` rozhraní by bylo možné vytvářet doklady a evidovat je do `EET`. Tato funkcionalita by měla využití např. při propojení s internetovým obchodem, kdy by docházelo k vystavení dokladů přes pokladnu.

Zpřístupnění katalogu zboží v portálu pro zákazníky

U produktů v katalogu zboží by bylo možné doplnit popis produktu a přiložit jeho fotografii. Zákazník by měl katalog zboží zpřístupněn v portálu pro zákazníky, kdy by si mohl katalog procházet a zjistit tak, zda daný sortiment prodejce prodává či zjistit bližší informace o produktu.

Využití `API` rozhraní aplikace *Účtenkovka*

Aplikace *Účtenkovka*, do které zákazníci vkládají své účtenky pro účast v loterii, nabízí `API` rozhraní, jež by se dalo využít. Zákazník by takto mohl účtenku zaevidovat do loterie pouhým kliknutím na tlačítko v detailu dokladu.

Literatura

[1] *Usage of HTTP/2 for websites*. W^{3}Techs Web Technology Surveys, [Online; navštíveno 4.12.2017].
URL https://w3techs.com/technologies/details/ce-http2/all/all

[2] *Bootstrap*. Wikipedie, 2017, [Online; navštíveno 21.12.2017].
URL https://cs.wikipedia.org/wiki/Bootstrap

[3] *Elektronická evidence tržeb*. Wikipedie, 2017, [Online; navštíveno 4.12.2017].
URL https://cs.wikipedia.org/wiki/Elektronick%C3%A1_evidence_tr%C5%BEeb

[4] *Knowledge Base of Relational and NoSQL Database Management Systems*. DB-engines, 2017, [Online; navštíveno 21.12.2017].
URL https://db-engines.com/en/ranking

[5] *Model-view-controller*. Wikipedie, 2017, [Online; navštíveno 21.12.2017].
URL https://cs.wikipedia.org/wiki/Model-view-controller

[6] *MySQL*. Wikipedie, 2017, [Online; navštíveno 21.12.2017].
URL https://cs.wikipedia.org/wiki/MySQL

[7] *Seriál: Doctrine 2 (12 dílů)*. 2017, [Online; navštíveno 21.12.2010].
URL https://www.zdrojak.cz/serialy/doctrine-2/

[8] *Web 2.0*. Wikipedie, 2017, [Online; navštíveno 1.12.2017].
URL https://cs.wikipedia.org/wiki/Web_2.0

[9] *Webová služba*. Wikipedie, 2017, [Online; navštíveno 4.12.2017].
URL https://cs.wikipedia.org/wiki/Webov%C3%A1_slu%C5%BEba

[10] *XML-RPC*. Wikipedie, 2017, [Online; navštíveno 5.12.2017].
URL https://cs.wikipedia.org/wiki/XML-RPC

[11] *Způsob fungování systému evidence tržeb*. Etržby – Generální finanční ředitelství, 2017, [Online; navštíveno 5.12.2017].
URL http://www.etrzby.cz/cs/jak-to-funguje

[12] Chaffer, J.; Swedberg, K.: *Mistrovství v jQuery*. Brno: Computer Press, první vydání, 2013, ISBN 978-802-5141-038.

[13] Finanční správa: *Formát a struktura údajů o evidované tržbě*. 2016, [Online; navštíveno 8.12.2017].
URL http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.pdf

[14] Gourley, D.; Totty, B.: *HTTP*. Sebastopol, CA: O'Reilly, první vydání, 2002, ISBN 15-659-2509-2.

[15] Han, J.; Kamber, M.: *Data mining*. San Francisco, CA: Morgan Kaufmann, druhé vydání, 2006, ISBN 978-1-55860-901-3.

[16] International Business Machines Corporation: *Web Services Security (WS-Security)*. 2017, [Online; navštíveno 8.12.2017].
URL https://www.ibm.com/support/knowledgecenter/en/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac55630_.htm

[17] Jaroslav Zendulka a kol.: *Studijní opora k předmětu Získávání znalostí z databází*. VUT FIT Brno, 2009.

[18] Krajské redakce České televize, Adnreas Papadopulos: *40 dnů s EET v obchodech – o 40 procent víc spotřeby termopapíru*. Reportáž ČT24, 2017, [Online; navštíveno 9.12.2017].
URL https://www.facebook.com/CT24.cz/videos/10155299377494009/

[19] Ludin, S.; Garza, J.: *Learning HTTP/2*. Boston: O'Reilly, first edition. vydání, 2017, ISBN 14-919-6244-5.

[20] Michal Burda: *Získávání znalostí z databází – Asociační pravidla*. 2004, [Online; navštíveno 1.1.2018].
URL http://www.fit.vutbr.cz/study/courses/ZZD/public/seminar0304/GUHA-text.pdf

[21] Nette Foundation: *Nette Framework – Rychlý a pohodlný vývoj webových aplikací v PHP*. 2017, [Online; navštíveno 21.12.2010].
URL https://nette.org/cs/

[22] Philippe Fournier-Viger: *How to auto-adjust the minimum support threshold according to the data size*. 2013, [Online; navštíveno 1.3.2018].
URL http://data-mining.philippe-fournier-viger.com/how-to-auto-adjust-the-minimum-support-threshold-according-to-the-data-size/

[23] Pitner, T.: *Webové služby*. Studijní materiály předmětu FI:PA165, 2004, [Online; navštíveno 6.12.2017].
URL https://is.muni.cz/el/1433/podzim2004/PA165/um/prednaska11/index.html

[24] Roy, F.; James, G.; Nielsen, H. F.; aj.: *RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1*. IETF, 1999, [Online; navštíveno 27.11.2017].
URL https://tools.ietf.org/html/rfc2616

[25] Satrapa, P.: *Jak funguje nový protokol HTTP/2*. ROOT.CZ, 2015, [Online; navštíveno 4.12.2017].
URL https://www.root.cz/clanky/jak-funguje-novy-protokol-http-2/

[26] Smith, N.: *Opinion: When Chrome, YouTube and Firefox drop it like it's hot, Flash is a dead plugin walking*. Science X, 2015, [Online; navštíveno 20.11.2017].
URL https://phys.org/news/2015-07-opinion-chrome-youtube-firefox-hot.html

[27] Spurlock, J.: *Bootstrap*. Beijing: O'Reilly, první vydání, [2013], ISBN 14-493-4391-0.

[28] Tari, Z.; Phan, A.; Jayasinghe, M.; aj.: *On the Performance of Web Services*. SpringerLink : Bücher, Springer US, 2011, ISBN 9781461419303.
URL https://books.google.cz/books?id=vXL-M73vdvUC

[29] Vrána, J.: *Webové služby v PHP: XML-RPC a SOAP*. Root.cz, 2007, [Online; navštíveno 5.12.2017].
URL https://www.root.cz/clanky/webove-sluzby-php-xmlrpc-soap/

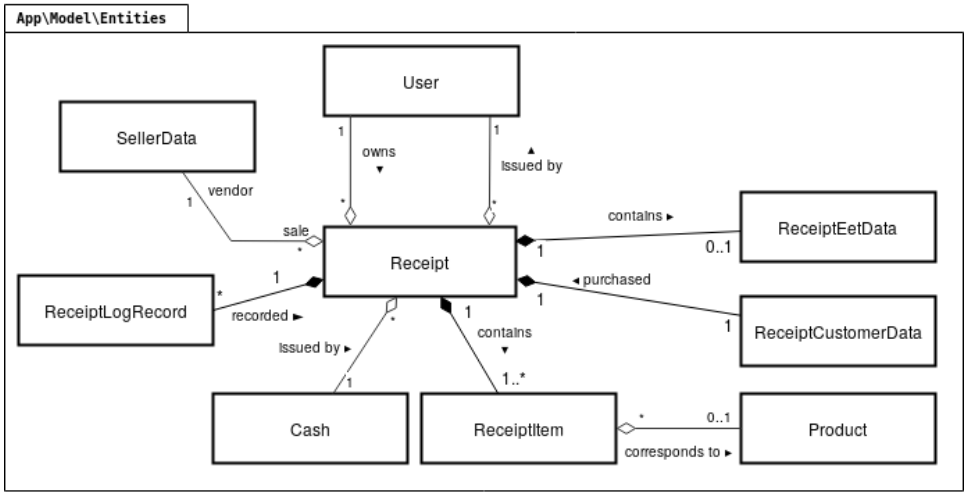
[30] W3Techs: *Usage of client-side programming languages for websites*. [Online; navštíveno 19.12.2017].
URL https://w3techs.com/technologies/overview/client_side_language/all

[31] Welling, L.: *PHP a MySQL*. Praha: SoftPress, první vydání, 2003, ISBN 80-864-9760-7.

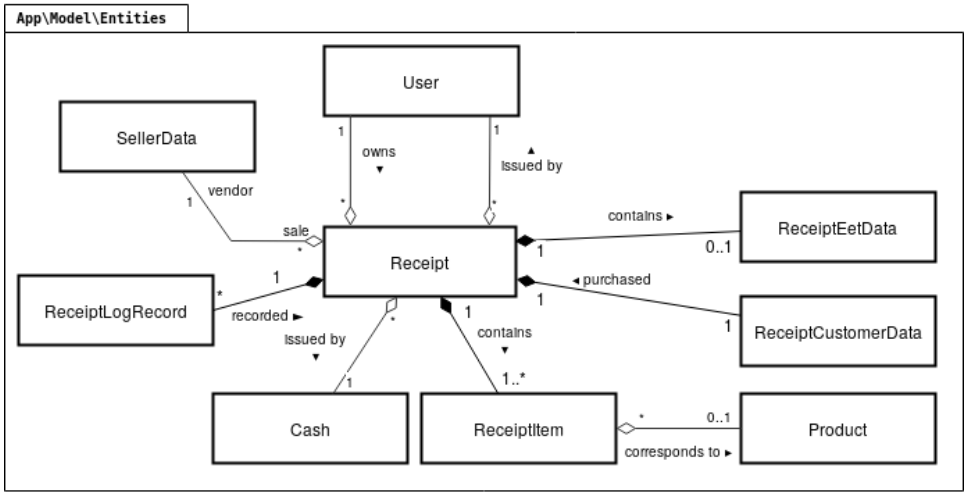
[32] Řepa, V.: *Procesně řízená organizace*. Grada, první vydání, 2012, ISBN 978-80-247-4128-4.

Příloha A

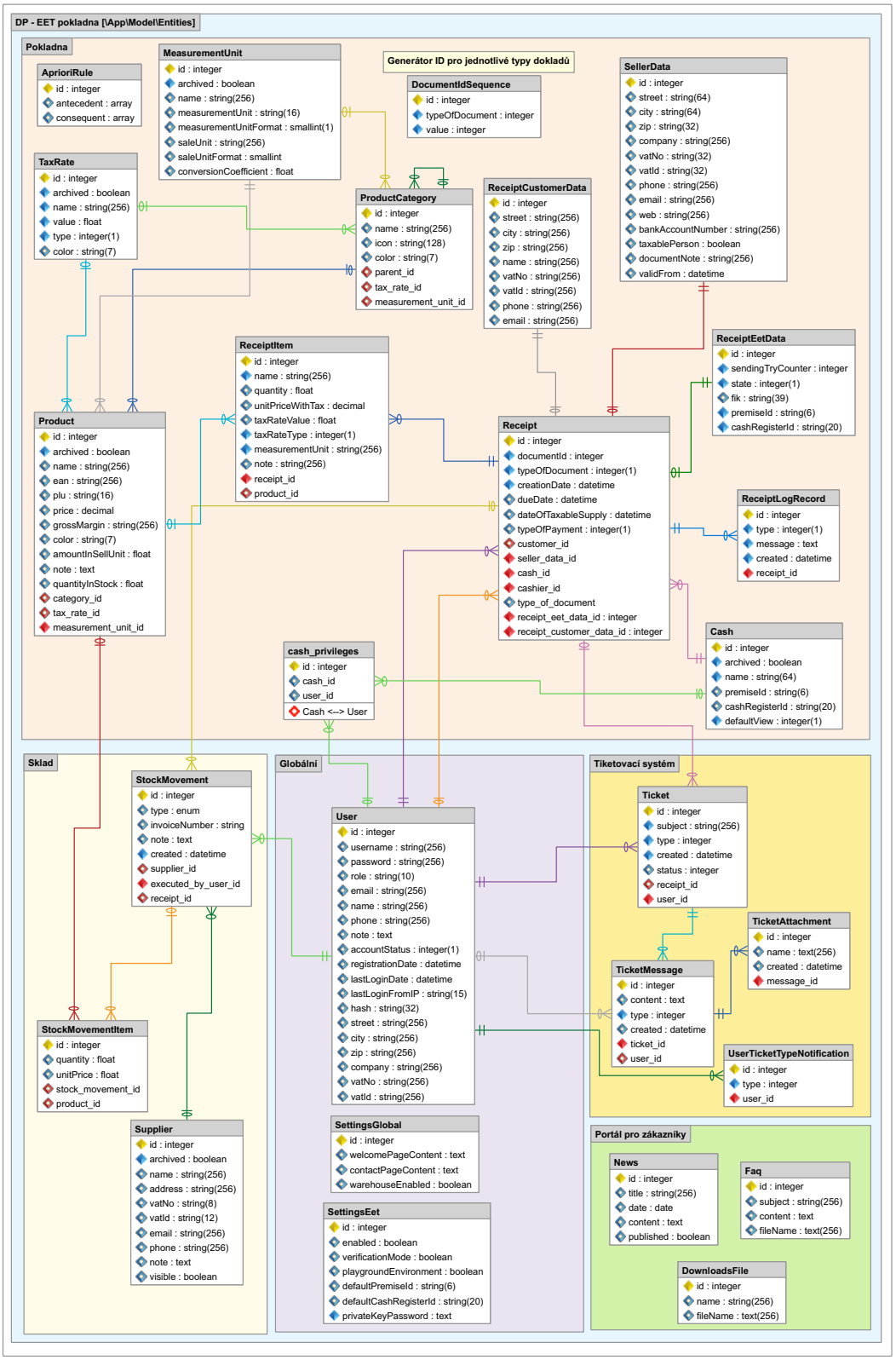
Návrhové diagramy



Obrázek A.1: Diagram třídy `Receipt`



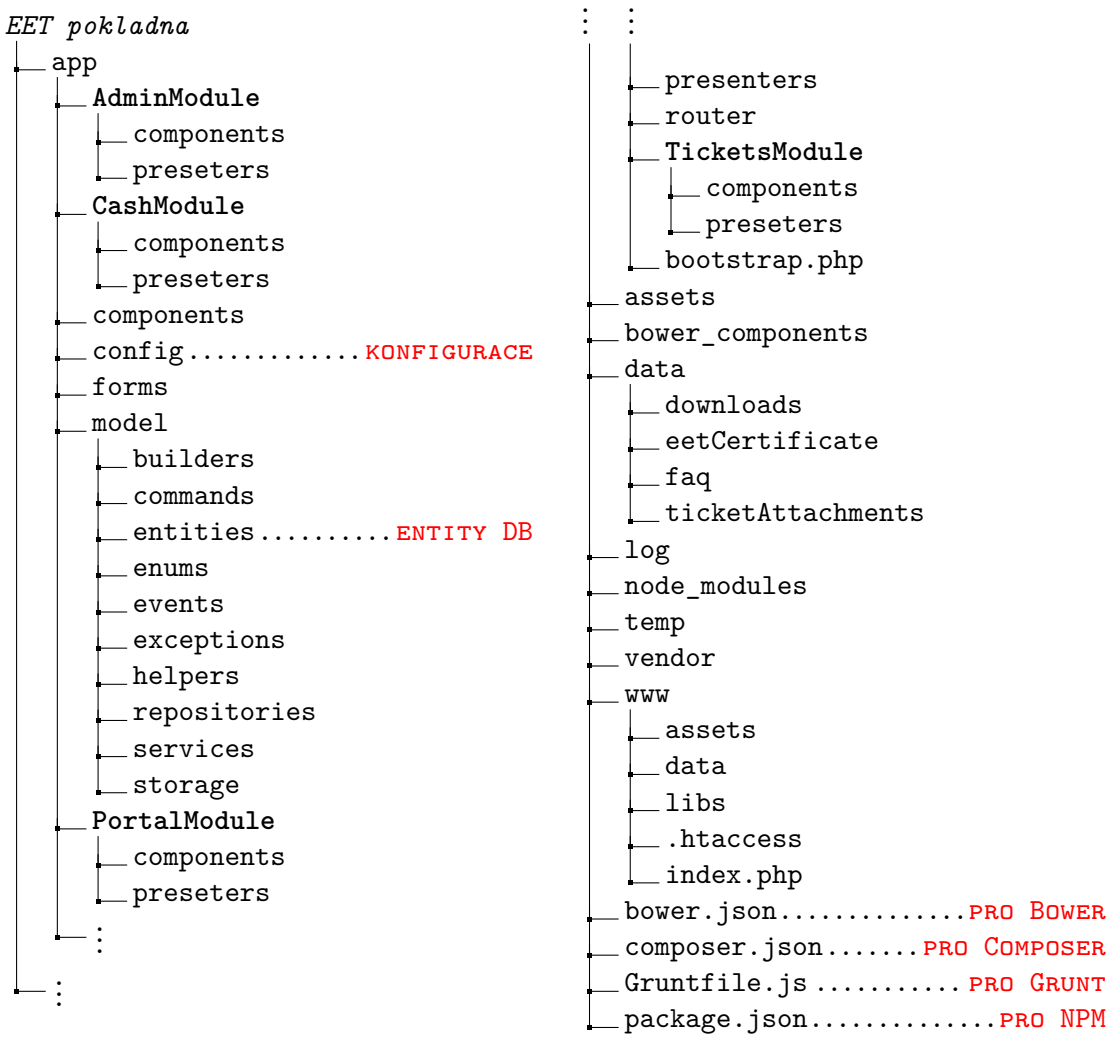
Obrázek A.2: Diagram třídy `Product`



Obrázek A.3: Logické schéma databáze

Příloha B

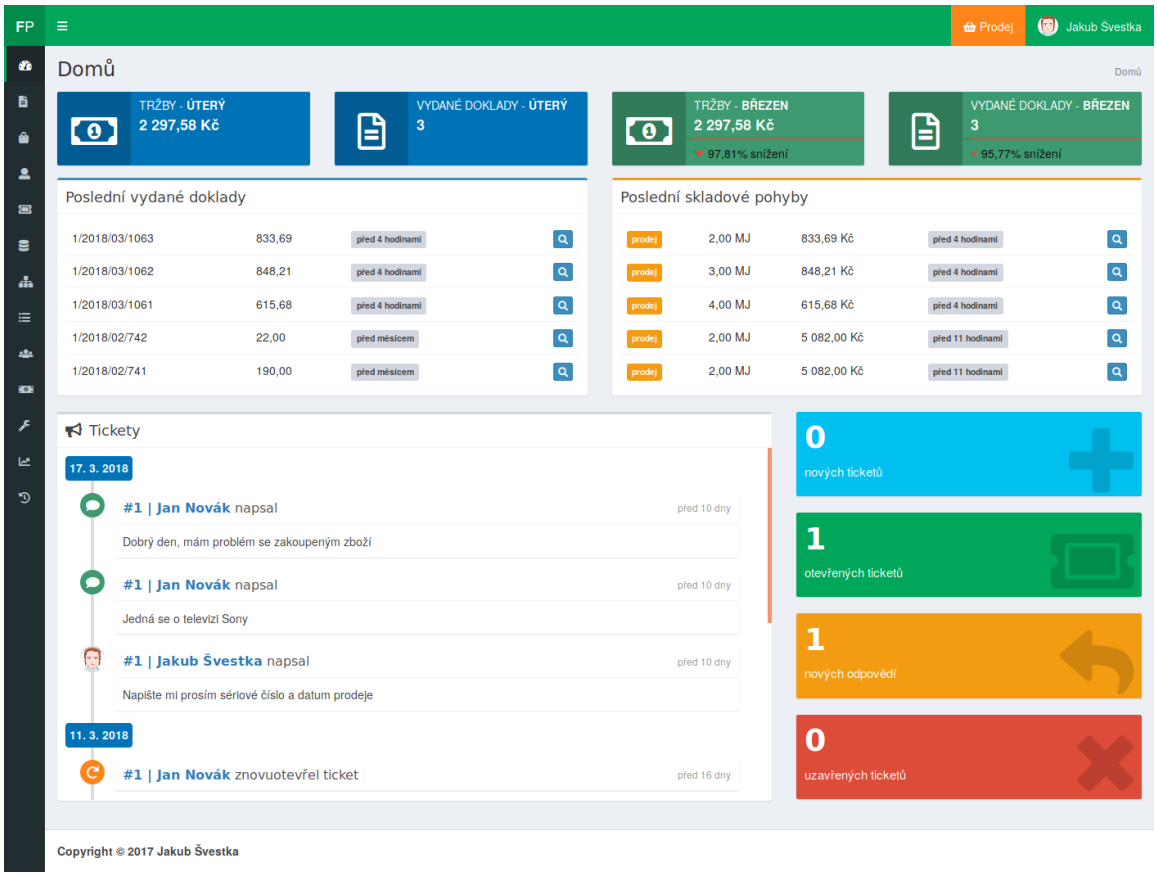
Adresářová struktura aplikace



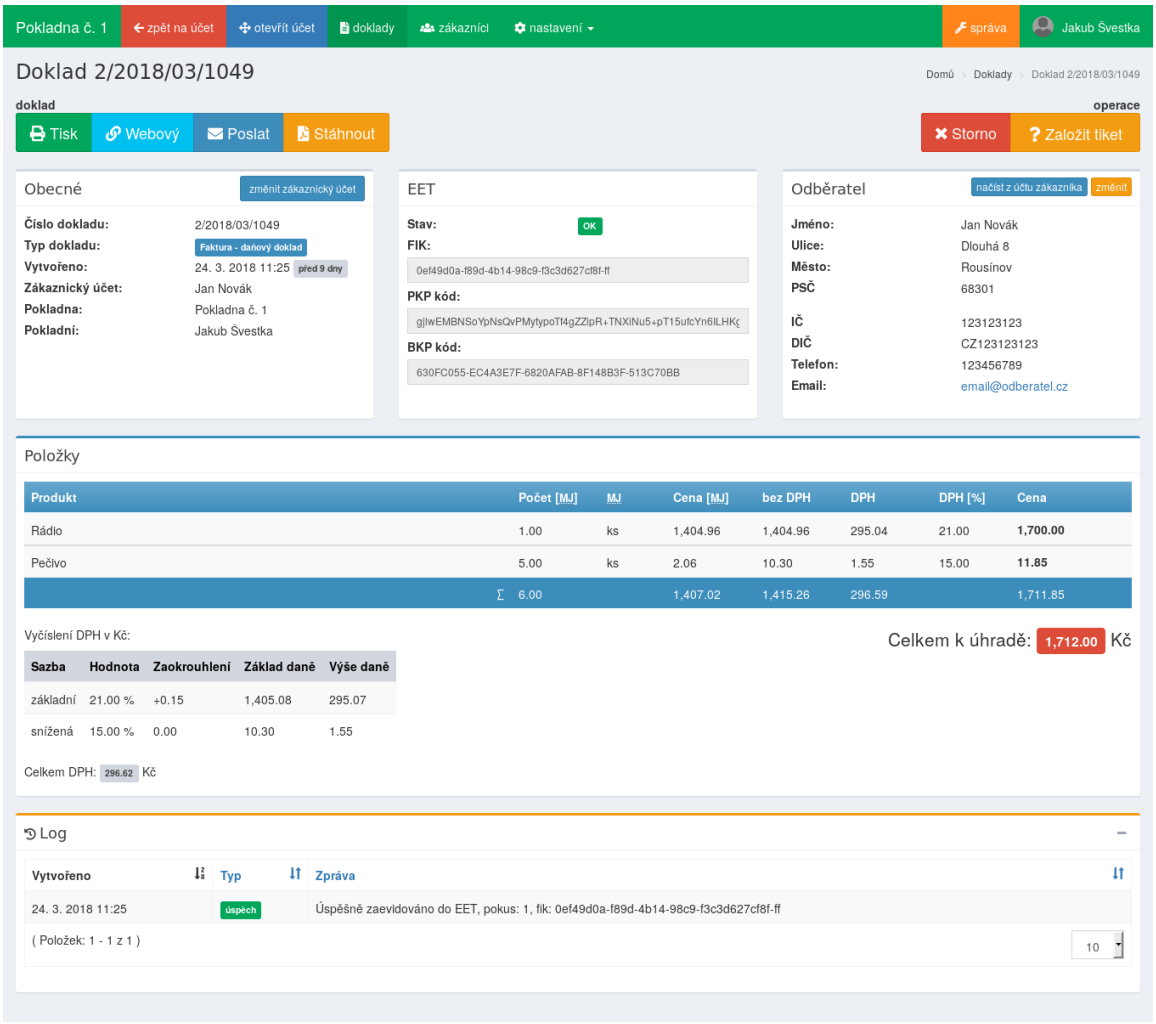
Obrázek B.1: Adresářová struktura aplikace *EET pokladna*

Příloha C

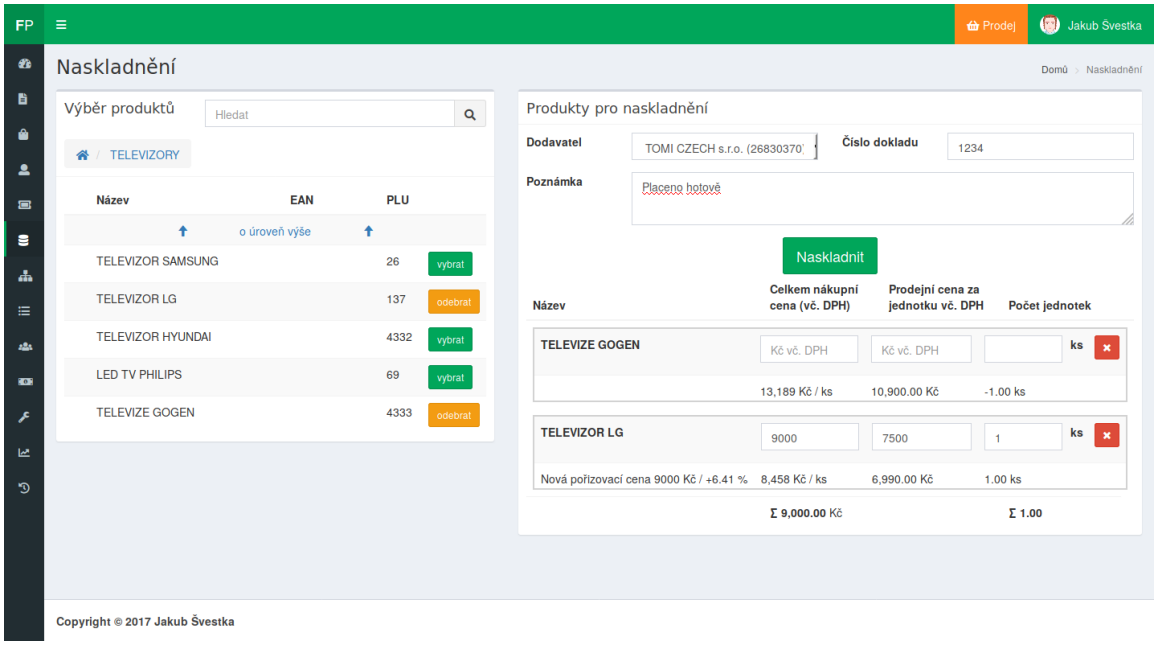
Snímky aplikace



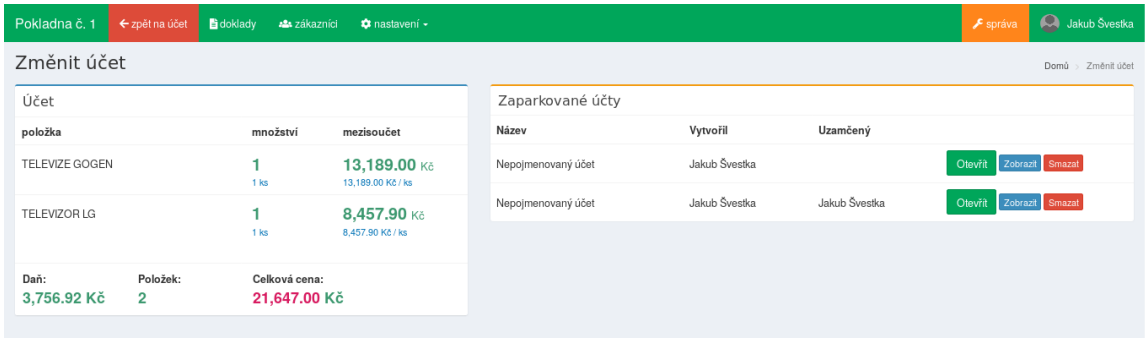
Obrázek C.1: Dashboard v administraci



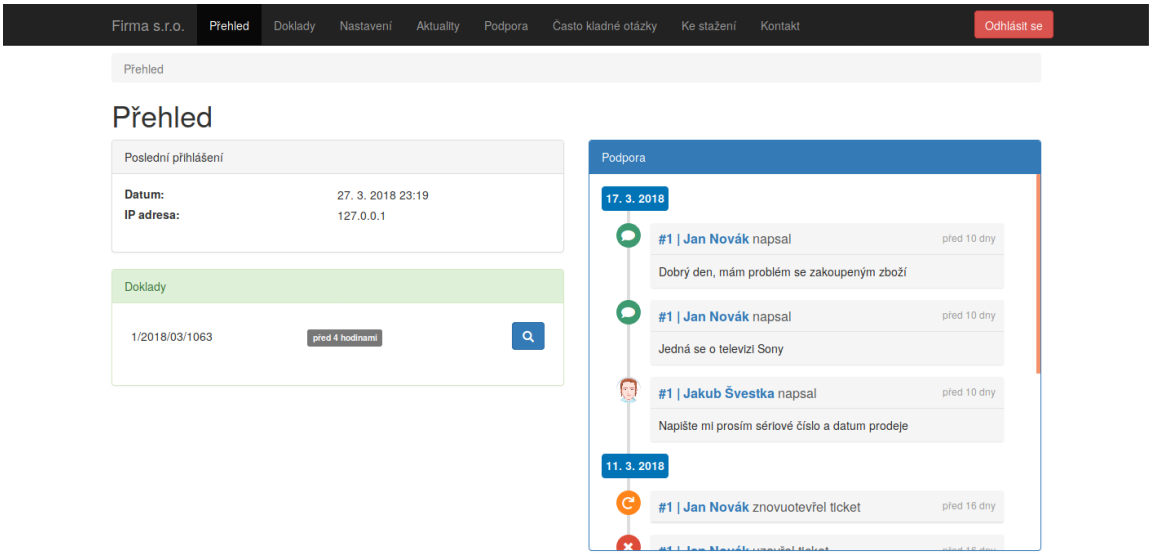
Obrázek C.2: Detail dokladu



Obrázek C.3: Naskladnění zboží



Obrázek C.4: Správa otevřených účtů na pokladně



Obrázek C.5: Portál pro zákazníky – dashboard

Příloha D

Ukázka vygenerovaných dokladů

Firma s.r.o.

Zjednodušený daňový doklad

vygenerováno aplikací FÉR pokladna | doklad online: http://dp.jakubsvestka.eu/g9yd6

č. **1/2018/04/817**

Krásná 8 IČ: 123456789 *telefon:* 123456789 *režim tržby:* běžný

68201 Vyškov DIČ: CZ123456789 *email:* dodavatel@email.tld *provozovna:* 1

web: www.web-dodavatele.tld *pokladna:* 1

vytvořeno: 09. 04. 2018 / 10:39:26

#	Produkt	Počet [MJ]	MJ	Cena [MJ/Kč]	Mezisoučet [Kč]
1	Položka A	10,00	ks	99,90	999,00
2	Položka B *výrobní číslo: A82BD912*	3,00	ks	49,00	147,00
3	Položka C	5,00	ks	0,50	2,50

Rozpis DPH v Kč:

Sazba	Zaokrouhlení	Základ daně	Výše daně
21 %	+0.50	828.10	173.90
15 %	0.00	127.83	19.17

Celkem: **1 149,00** Kč

FIK: 43f0188e-f399-48c2-9b96-a2423d7c5be2-ff

BKP: EB7B1600-A9A5B51B-4DA948A5-2935CE61-134D1FF4

poznámka na dokladu

Obrázek D.1: Zjednodušený daňový doklad

Firma s.r.o.

vygenerováno aplikací FÉR pokladna | doklad online: http://dp.jakubsvestka.eu/g9yd6

Faktura - daňový doklad

č. **2/2018/04/817**

Dodavatel

Firma s.r.o.

Krásná 8
68201 Vyškov

IČ: 123456789
DIČ: CZ123456789

Odběratel

Jan Novák

Dlouhá 8
68301 Rousínov

IČ: 123123123
DIČ: CZ123123123

Typ platby:	**bankovní převod**	Datum vystavení:	**09. 04. 2018 / 10:39:26**
Číslo účtu:	**123456789/0300**	Datum zdan. plnění:	**13. 04. 2018**
Variabilní symbol:	**2817**	**Datum splatnosti:**	**10. 04. 2018**

#	Produkt	Počet MJ	MJ	Cena MJ/Kč	bez DPH Kč	DPH Kč	DPH %	Cena Kč
1	Položka A	10.00	ks	82.56	825.62	173.38	21.00	999.00
2	Položka B *výrobní číslo: A82BD912*	3.00	ks	42.61	127.83	19.17	15.00	147.00
3	Položka C	5.00	ks	0.41	2.07	0.43	21.00	2.50

Rozpis DPH v Kč:

Sazba	Zaokrouhlení	Základ daně	Výše daně
21 %	+0.50	828.10	173.90
15 %	0.00	127.83	19.17

Celkem bez DPH: 956,00 Kč

Celkem: **1 149,00** Kč

režim tržby: běžný
číslo provozovny: 1
pokladna: 1
FIK: 43f0188e-f399-48c2-9b96-a2423d7c5be2-ff
BKP: 80F24588-5BCAEAAA-878E795A-2AF90289-2FE1F52B

poznámka na dokladu

Obrázek D.2: Faktura - daňový doklad

Příloha E

Manuál k testování portálu pro zákazníky

Testování uživatelského rozhraní aplikace

Právě jste byl(a) obeznámen(a) s funkcemi *portálu pro zákazníky* aplikace *Fér pokladna*. Nyní postupujte dle následujících úkolů, jejichž cílem bude provedení několika operací s aplikací. V případě jakýchkoliv dotazů v průběhu plnění úkolů mne neváhejte oslovit. Veškeré postřehy, co se Vám v aplikaci líbilo, nelíbilo či byste vylepšili, napište prosím do dotazníku (poslední otázka), který následuje za testovacími úkoly.

Úkoly

1. Přejděte na stránku aplikace `dp.jakubsvestka.cz` pomocí Vašeho preferovaného webového prohlížeče.
2. **Registrace**
   - Proveďte registraci, kdy v registračním formuláři vyplňte pouze povinné údaje. V případě, že jste podnikatel, proveďte načtení kontaktních údajů na základě `IČ`. Dbejte prosím na správné vyplnění emailu z důvodu aktivace účtu.
   - V emailu Vám došel email s odkazem pro aktivaci účtu. V případě, že ho nemůžete najít, zkuste zkontrolovat složku s nevyžádanou poštou (spam).
   - Aktivujte účet.
   - Přihlaste se pod vytvořeným účtem.
3. **Změna uživatelských údajů**
   - Doplňte ke svému účtu **telefonní číslo**.
4. **Zobrazení dokladu**
   - Proveďte zobrazení dokladu, který byl k Vašemu účtu přiřazen.
   - Proveďte zaslání dokladu na Váš email.
5. **Vytvoření tiketu**

- Proveďte vytvoření tiketu v sekci *Podpora* a přiřaďte k tiketu doklad, který byl k Vašemu účtu přiřazen.

Testovací část je ukončena. Nyní prosím vyplňte následující dotazník.

Dotazník

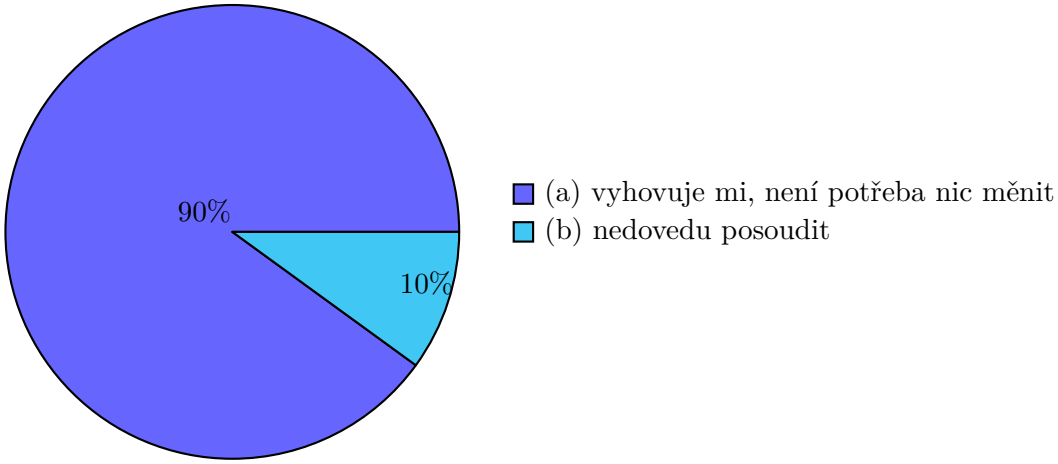
Zde prosím odpovězte na několik otázek odpovědí, která je Vám nejbližší. U každé otázky vyberte **pouze jednu** možnost. Na závěr uveďte jakékoliv postřehy, co se Vám na aplikaci líbilo či nelíbilo.

---

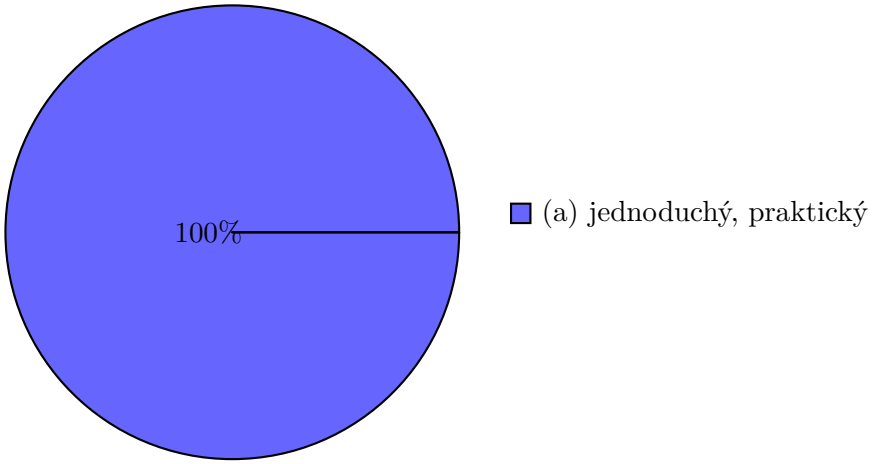
1. **Jak jste spokojen(a) s rozvržením uživatelského rozhraní aplikace?**
   (a) Vyhovuje, není potřeba nic měnit.
   (b) Spokojen/a, ale některé části by mohly být přehlednější.
   (c) Nevyhovuje mi, není dostatečně intuitivní.
   (d) Nedovedu posoudit.
2. **Jak by jste zhodnotil(a) grafický vzhled aplikace?**
   (a) Jednoduchý, praktický.
   (b) Nepřehledný.
   (c) Nedovedu posoudit.
3. **Z jakého zařízení jste s aplikací pracoval(a)?**
   (a) Počítač (notebook).
   (b) Mobilní zařízení (telefon, tablet).
4. **Jak hodnotíte ovládání aplikace?**
   (a) Jednoduché, vždy jsem našel(la) co bylo potřeba.
   (b) Někdy jsem nevěděl(a), jak dále postupovat.
   (c) Zpočátku složité, ale pak jsme se zorientoval(a).
   (d) Nedovedu posoudit.
5. **Líbí se Vám koncept portálu pro zákazníky?**
   (a) Ano.
   (b) Ne, preferuji tištěný doklad.
   (c) Nevím.
6. **Zde uveďte jakékoliv postřehy, co se Vám na aplikaci líbilo či nelíbilo.**

Konec dotazníku. Děkuji za Váš čas, jenž jste věnovali jeho vyplnění.

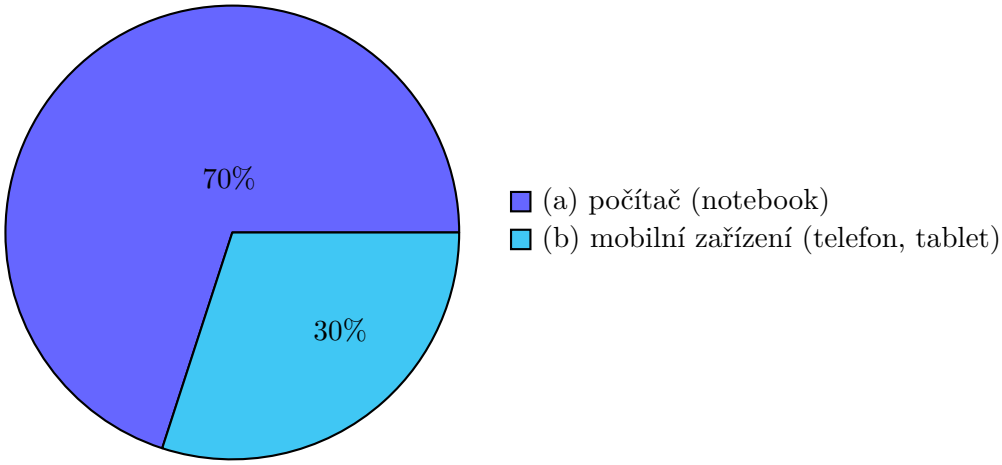
Výsledky dotazníku



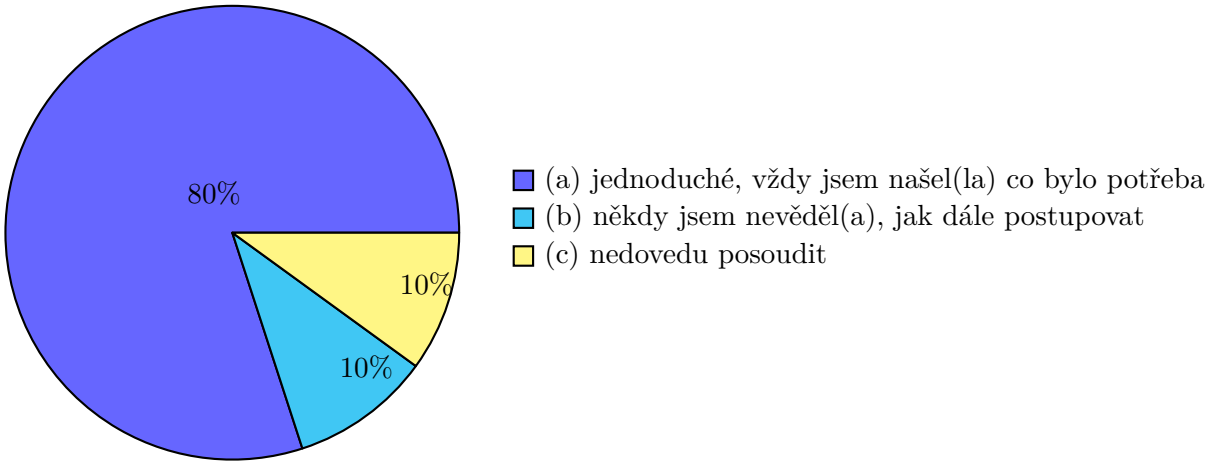
Graf E.1: Jak jste spokojen(a) s rozvržením uživatelského rozhraní aplikace?



Graf E.2: Jak by jste zhodnotil(a) grafický vzhled aplikace?



Graf E.3: Z jakého zařízení jste s aplikací pracoval(a)?



Graf E.4: Jak hodnotíte ovládání aplikace?

Graf E.5: Líbí se Vám koncept portálu pro zákazníky?

Příloha F

Instalační manuál

Potřebný software

- webový server *Apache* (testováno na verzi `2.4`)
  - zapnutý mód `mod_rewirte`
- `PHP` v minimální verzi `7.1` s knihovnami:
  - `php_gmp`
  - `php_json`
  - `php_sqlite3`
  - `php_curl`
  - `php_mbstring`
  - `php_json`
  - `php_soap`
  - `php_zip`
  - `php_xml`
  - `php_tidy`
  - `php_memcached`
  - `php_openssl`
  - `php_mcrypt`
- databázový server *MySQL* (testováno na verzi `5.7`)
- kešovací systém *Memcached*
- balíčkovací systémy: *Bower*, *Composer*, *npm*
- nástroj *Grunt*

Postup instalace

Veškeré příkazy musí být spuštěny v kořenovém adresáři aplikace.

1. instalace závislostí

```
$ composer install
$ bower install
$ npm install
```

2. spuštění úloh – vygenerování minifikovaných `CSS` stylů a `JS` skritpů, kopie obrázků a písem

```
$ grunt production
```

3. nastavení přístupu do databáze
   - vytvoření souboru `app/config/config.local.neon` s obsahem:

```
doctrine:
        user: jmeno
        password: heslo
        dbname: nazev_databaze
```

4. přístupová práva adresářů
   - následující adresáře musí mít právo pro zápis webovým serverem: `temp, log, data`

5. inicializace pokladního systému pomocí interaktivního průvodce

```
$ php www/index.php install

Instalace aplikace FER POKLADNA
===============================

Vytvoreni DB
------------
Test pripojeni k~DB                OK
Databaze                           dp-eet

Vytvorit schema databaze? (yes/no) [yes]:
> yes

Smazani DB                         OK
Vytvoreni DB                       OK

Ucet provozovatele
------------------
Jmeno:
> Jan Novak

Uzivatelske jmeno:
> novak

Heslo (skryte):
>

[OK] Instalace dokoncena
```

Tímto je instalace dokončena. Přihlášení do aplikace lze provést pomocí zvolených přihlašovacích údajů, jež náleží k účtu provozovatele. Mapování `URL` adres na moduly je následující:

- `domena` – portál pro zákazníky
- `domena/cash` – pokladní část
- `domena/admin` – správa pokladny.

Příloha G

Obsah přiloženého CD

CD

- xsvest05_eet_pokladna.zip ..... zdrojový kód aplikace
- instalacni_manual.pdf ..... instalační manuál
- xsvest05_DP_prace.zip ..... zdrojový kód práce pro sazbu v LaTeXu
- xsvest05_DP_prace.pdf ..... práce ve formátu PDF
- videa/ ..... videoprezentace částí aplikace