FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Opensource portál pro tvorbu a správu faktur s webovým rozhraním a API
Student:	Bc. Petr Suchý
Vedoucí:	Ing. Vojtěch Jirkovský
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

1. Analyzujte konkurenční řešení pro tvorbu a správu faktur.
2. Seznamte se se specifickými požadavky cílové skupiny uživatelů.
3. Analyzujte možnosti API různých bankovních ústavů.
4. Na základě předchozích analýz navrhněte API a webové rozhraní portálu.
   a) Portál by měl podporovat export faktur do PDF
   b) Pokuste se navrhnout propojení s API bank pro automatickou kontrolu uhrazení faktury
5. Zvolte vhodnou technologii a řešení implementujte.
6. Řešení otestujte, analyzujte připomínky uživatelů a aplikaci patřičně upravte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 29. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Opensource portál pro tvorbu a správu faktur s webovým rozhraním a API

Bc. Petr Suchý

Katedra softwarového inženýrství
Vedoucí práce: Ing. Vojtěch Jirkovský

29. června 2018

Poděkování

Děkuji inženýru Vojtěchu Jirkovskému za konzultace při řešení diplomové práce, rodině za podporu a Martinu Melkovi za neúnavné zodpovídání mých dotazů.

Prohlášení

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

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

V Praze dne 29. června 2018 .....

Odkaz na tuto práci

Suchý, Petr. *Opensource portál pro tvorbu a správu faktur s webovým rozhraním a API*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato diplomová práce se zabývá tvorbou webové aplikace pro vytváření a následnou správu účetních dokladů neboli faktur. Za cíl si klade analyzovat existující řešení problému a dále požadavky podnikatelů na funkčnost takové aplikace. Na výsledky analýzy naváže návrhem aplikace, která bude dostupná přes aplikační rozhraní a webového rozhraní, které bude sloužit běžným uživatelům k ovládání aplikace. Aplikace klade důraz na přenositelnost a schopnost fungovat v různorodých prostředích. Aplikace bude umožňovat kromě vytváření a správy faktur také jejich export do formátu PDF, odesílání faktur emailem přímo z prostředí aplikace, adresář klientů nebo ceník zboží. Výstupem práce je aplikace ovladatelná přes REST API nabízející funkčnost požadovanou zadáním a dalšími požadavky, které vzešly z analýzy. Druhou částí výstupu je pak jednoduché webové rozhraní pro ovládání aplikace.

Klíčová slova Faktura, Účetní doklad, ARES, PDF, QR platba, PSD2

Abstract

The aim of this diploma thesis is the development of a web application for creating and managing invoices. It's goal is to analyze existing solutions as well as needs and wishes of the entrepreneurs. The results of analysis will be

considered while designing the application, accessible via application interface and also while designing a web interface for common users.
The application emphasizes on portability and ability to run on various environments. The application will provide not just ability to create and manage invoices, but the ability to export them to PDF, send them via email, store user's clients informations or create user's own price list as well.
The result of the thesis is a web application controllable via the REST interface providing all the functionality required by assignment and other functionality which appeared during analysis. Second part of the output is a simple web interface allowing users to control the application.

Keywords Invoice, ARES, PDF, QR payment, PSD2

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Podle údajů české správy sociálního zabezpečení bylo, na konci roku 2017, v České Republice necelý milion [1] osob, které měly podnikání jako hlavní nebo vedlejší činnost. Určitá část těchto podnikajících osob potřebuje vystavovat faktury. Jedním z nich jsem se nedlouho po nástupu na vysokou školu stal i já.

Několik let jsem využíval služeb nejpoužívanějšího [2] českého online portálu na tvorbu a správu faktur **iDoklad**, protože nabízel všechny funkce, které jsem potřeboval a byl zdarma. V polovině roku 2017 se však iDoklad rozhodl většinu funkcí zpoplatnit [3] v rámci jejich tarifů. Byla sice zachována možnost využívat portál zdarma, ale pouze s velmi omezenou sadou funkcí a restrikcemi na počet klientů.

Od té doby jsem několikrát zvažoval vytvoření obdobné aplikace, která by nabídla podobnou sadu funkcí, jako iDoklad před zpoplatnění, avšak zdarma a bez omezení. Jako další výhodu jsem zvažoval zveřejnění zdrojového kódu pod otevřenou licencí tak, aby každý mohl aplikaci upravit pro svoje specifické použití nebo ji nasadit na vlastním webhostingu, kde se nemusí bát zneužití jeho údajů třetí stranou.

iDoklad je dnes velmi robustní nástroj a bylo by naivní předpokládat vytvoření portálu se shodnou funkcionalitou v rámci diplomové práce. V rámci této práce bych tedy rád analyzoval funkce, které jsou důležité především pro jednotlivce, kteří nepotřebují většinu pokročilých funkcí a více ocení službu zdarma, navíc s otevřeným zdrojovým kódem. Na základě této analýzy bych chtěl navrhnout a implementovat webovou aplikaci, která těmto požadavkům vyhoví a umožní těmto uživatelům i nadále zdarma vytvářet a spravovat jejich faktury online.

Kapitola 1

Cíle práce

Prvním cílem mojí práce je seznámit se s již existujícími řešeními na tvorbu a správu faktur, a to jak v online prostředí, tak v podobě aplikací na klientských zařízeních. Provést důkladnou analýzu funkcí, které jednotlivé služby nabízejí a určit jejich výhody a nevýhody.

Druhým cílem je seznámit se s požadavky skutečných uživatelů – určit, které funkce jsou pro ně klíčové, které postradatelné a v neposlední řadě identifikovat funkce, které jim třeba žádná v současnosti dostupná služba nenabízí.

Třetím analytickým cílem mé práce je seznámit se s aplikačním rozhraním různých českých bankovních ústavů, za účelem kontroly příchozích plateb, což by mělo umožnit automatickou kontrolu stavu faktur.

Na základě informací získaných v předchozích částech bude mým dalším cílem určit, které funkce by měl navrhovaný portál nabízet a na základě toho navrhnout aplikační a následně i webové rozhraní, která budou zohledňovat požadavky uživatelů a zároveň budou klást důraz na srozumitelnost, přehlednost a použitelnost celé aplikace. V rámci analýzy bude kladen důraz na body zmíněné v zadání práce – export faktur do formátu PDF a automatická kontrola stavu zaplacení faktury pomocí API bank. Důležité pro návrh i následnou implementaci je též, aby si tuto aplikaci mohl každý uživatel snadno zprovoznit i na svém webu.

Dalším cílem mé práce je zvolit vhodnou technologii a implementovat funkce navržené v předchozím bodě. Během implementace bude dbáno na dodržení dobrých zvyklostí a především standardů tak, aby byla zajištěna stejná funkčnost napříč webovými prohlížeči.

Posledním cílem této práce je otestování implementované aplikace, identifikace problémů, chyb a nedostatků a společně s připomínkami a návrhy od uživatelů aplikaci upravit, případně navrhnout dlouhodobější plán přidávání složitějších funkcí. V neposlední řadě také nastínit možný další vývoj a rozšíření portálu.

Kapitola 2

Rešeršní část

2.1 Podnikatel

Podnikatel je v novým Občanským zákoník definován [4]: „Podnikatelem je podle § 420 fyzická nebo právnická osoba, která samostatně vykonává na vlastní účet a odpovědnost výdělečnou činnost živnostenským nebo obdobným způsobem se záměrem činit tak soustavně za účelem dosažení zisku. Podle této definice se pro to, aby byla osoba považována za podnikatele, nevyžaduje podnikatelské oprávnění. Rozhodujícím kritériem je, jak se osoba v právním styku fakticky chová, tedy skutečná povaha profesionální činnosti podnikatele.“.

Podnikatelem je tedy i každá osoba, která:

- je zapsaná v obchodním rejstříku
- má k podnikání živnostenské nebo jiné oprávnění podle jiného zákona
- uzavírá smlouvy související s vlastní obchodní, výrobní nebo obdobnou činností

Podnikání podle živnostenského zákona Podnikatele, respektive podnikání definuje i živnostenský zákon [4] jako „soustavnou činnost provozovanou samostatně, vlastním jménem, na vlastní odpovědnost, za účelem dosažení zisku a za podmínek stanovených Živnostenským zákonem.“.

2.2 Faktura

Faktura plní funkci účetního dokladu. Z toho plyne, že musí obsahovat několik údajů. Které údaje musí obsahovat se liší podle toho, zda ji vystavil plátce či neplátce DPH. V rámci práce se zaměřuji především na neplátce DPH a tak uvedu ty údaje, které jsou povinné pro neplátce. Jde o podmnožinu informací

obsažených na faktuře vydané plátcem DPH. Jedná se tedy o tyto 4 základní údaje [5] :

- **Označení účastníků** – tedy jméno a příjmení (OSVČ), resp. název firmy, adresu, resp. sídlo firmy, IČO. Platí pro obě strany — dodavatele i odběratele. Vy, jako dodavatel, jste kromě těchto údajů povinen uvést i označení registru, ve kterém jste jako podnikatel zapsaný (obchodní rejstřík, živnostenský rejstřík apod.) a v případě právnické osoby údaj o zápisu včetně oddílu a vložky — toto jsou náležitosti "obchodních listin", které stanovuje Nový občanský zákoník.
- **Slovní a číselné označení dokladu** – např.: „Faktura 2018001“
- **Peněžní suma** – celkově a nebo podrobně (cena za měrnou jednotku a označení množství)
- **Den vyhotovení účetního dokladu** – neboli datum vystavení faktury

Jiné zdroje [6] uvádějí ještě, že by z faktury mělo být zřejmé označení zboží či služby, zda jde o příjem, výdaj či zálohu a v neposlední řadě také text „Podnikatel je zapsán v živnostenském rejstříku.“, pokud má podnikatel zřízenou živnost.

2.3 Směrnice PSD2

Evropská komise na svém zasedání, které proběhlo 25. listopadu 2015, schválila směrnici *2015/2366/EU*, známou též pod označením **PSD2** (Payment Services Directive 2) [7]. Tato směrnice je náhradou za směrnici PSD z roku 2007. Tisková zpráva České bankovní unie [8] uvádí „Tato legislativa mj. umožňuje klientům využívat nové služby nepřímého dání platebního příkazu prostřednictvím poskytovatelů služeb nepřímého dání platebního příkazu (tzv. třetích stran). Banky budou povinny poskytnout (se souhlasem uživatele) těmto třetí stranám informace o platebním účtu klientů.“. Tuto směrnici do české legislativy převádí nový zákon o platebním styku *č. 370/2017 Sb.*, který vstoupil v účinnost 13. 1. 2018 [9]. Směrnice jako taková je tedy již implementována v rámci platného českého zákona. Tisková zpráva ČBU dále uvádí, že významná část prováděcích předpisů EU (standardy, vyhlášky a doporučení), které jsou nutné pro zajištění fungování nových služeb, podléhajících regulaci však k tomuto datu nebyla v platnosti, jelikož nebyly schváleny jejich finální podoby, dostupné jsou zatím pouze návrhy těchto norem. Česká bankovní unie odhaduje, že k prvnímu vymáhání těchto norem dojde až v září roku 2019.

V současnosti již některé společnosti využívají pro přístup k informacím o účtech klienta takzvaný „screen scraping“ [10]. Ten jim umožňuje získávat veškeré informace o účtech klienta i o klientovi samotném. Před implementací směrnice PSD2 byl „screen scraping“ tolerován, ale nová směrnice jej zakazuje,

neboť kromě způsobu přístupu určuje i ke kterým informacím má třetí strana přístup. Nově by banky měly zpřístupnit API, přes které se následně třetí strany mohou dotazovat na stav účtů, historii transakcí nebo iniciovat platbu, to vše samozřejmě se souhlasem klienta.

Pro přístup k API je potřeba, aby se třetí strana zaregistrovala u ČNB a splnila podmínky pro udělení licence, jako je základní jmění, pojištění atd. [11]. Správní řízení pro získání této licence obvykle zabere 5 až 10 měsíců [12]. „Není to jen regulace směrem k bankám, jde zároveň o to, aby přístup k datům klientů nemohla získat ledajaká garážová firma.“ [11].

Tato směrnice by potenciálně velmi pomohla získávání informací o stavu účtů a především o historii transakcí na účtu uživatele, například pro službu automatického párování plateb s fakturami. Vzhledem k tomu, že ještě nejsou dostupné finální specifikace norem, které určují podobu aplikačního rozhraní a způsob jak data třetím stranám technicky zpřístupnit a dále vzhledem k podmínkám na získání licence a na dobu, kterou zabere řízení o jejím získání, bohužel nepovažuji za reálné ji v této práci využít.

Kapitola 3

Analýza

3.1 Požadavky uživatelů

V této sekci jsou popsány požadavky uživatelů, které vzešly z online diskuze s podnikateli[13] a z dotazníku[14].

3.1.1 Možnost exportu do PDF

Základním požadavkem, který je obsažen i v zadání této práce je možnost exportovat faktury ve formátu PDF.

3.1.2 QR platba

Dalším častým požadavkem je integrace QR platby do faktury. QR platba usnadňuje zadávání údajů z faktury, předchází překlepům a nepřesnostem. Dnes již většina mobilních bankovních aplikací umožňuje zadání platby právě QR kódem, jde tedy především o usnadnění pro klienty. Ukázku QR platby můžeme vidět na obrázku 3.1.

3.1.3 Propojení s registrem ARES

Další oblíbenou funkcí je načítání údajů o subjektech z registru ARES. Data jsou dostupná přes API, stačí znát IČO subjektu na který se dotazujeme. Usnadňuje to vyplňování údajů u nových klientů, které nám například předají neúplné údaje nebo není zcela jasný formát adresy. Načtením údajů z registru ARES si můžeme být jisti zadáním kompletních a správných údajů.

3.1.4 Podpis a logo

Mít v záhlaví faktury vlastní logo a pod celkovou částkou svůj vlastní podpis je další z požadavků, který se objevil opakovaně. Faktura je platná i bez loga

Obrázek 3.1: Ukázka QR platby

a podpis zákon také nevyžaduje, avšak myslím si, že faktura vypadá vizuálně lépe, pokud je na ní obojí přítomno.

3.1.5 Párování plateb

Až překvapivě častým požadavkem je automatické párování plateb s fakturami. Pokud tedy například klient zaplatí fakturu, ta se v systému automaticky označí jako zaplacená. V současnosti je tato funkce většinou implementována přes emailová oznámení odesílaná z banky, v budoucnu by to mělo jít přes API bank, ta jak jim to nařizuje směrnice PSD2, viz. sekce 2.3.

3.1.6 Adresář

Uživatelé většinou s klienty komunikují opakovaně a tak si kontakty chtějí ukládat do jakéhosi adresáře. Kromě základních informací potřebných pro vystavení faktury si uživatelé přejí ukládat i další informace jako je emailová adresa nebo telefon.

3.1.7 Statistiky a grafy

Několik uživatelů zmínilo, že oceňují různé přehledy, statistky a grafy. Obsahem statistik jsou pak například tržby nebo obrat za jednotlivá čtvrtletí či měsíce nebo procentuální vyjádření podílu příjmů od konkrétního klienta na celkových příjmech.

3.1.8 Export a odesílání

Mezi požadavky byl zmíněn i export údajů. Nejsem si zcela jist, co tím daný uživatel myslel, ale předpokládám, že měl na mysli buď export uložených kontaktů z adresáře a nebo export faktur do různých formátů (kromě PDF třeba ještě do HTML nebo Excelu). Pokud to bylo myšleno jako export dat pro následující vložení do jiné aplikace na správu faktur, ať už desktopové nebo online, bylo by elegantnějším řešením přistupovat k datům přímo přes API.

Ohledně odesílání emailů s fakturami přímo z aplikace mě zájem celkem překvapil. Požadavek se opakoval několikrát a je zřejmě pro uživatele mnohem důležitější, než jsem předpokládal.

3.1.9 Vlastní texty na faktuře

Uživatelé si dále přejí mít možnost upravovat většinu textů uvedených na faktuře. Jedná se především o informaci o penále z prodlení nebo oblíbená otázka, zda je nutné fakturu tisknou a nešlo by raději trochu ušetřit naše lesy.

3.1.10 Ceník

Několik uživatelů jako požadavek uvedlo existenci ceníku – možnosti uložit si často fakturované položky včetně jejich ceny, aby je pak mohl uživatel snadno zadat při vystavování faktury a nemusel informace o položce pokaždé zadávat znovu.

3.1.11 EET

Trochu překvapivě nikdo z uživatelů nevyjádřil potřebu napojení aplikace na systém EET. Osobně to přisuzuji tomu, že uživatelé, kterých se EET týká pro platby pravděpodobně využívají hardwarové pokladny, které přímo tisknout účtenky a nevyužívají pro vystavování faktur online nástroje.

3.2 Existující řešení

Tvorba a správa faktur online v cloudu není nový nápad a tak existuje spoustu portálů a aplikací, které se tím zabývají. V této části práce se pokusím analyzovat řešení konkurence. Ač je práce cílena na online službu, budu analyzovat webové portály i několik aplikací pro desktop.

3.2.1 Online

Nejprve se zaměřím na existující řešení, která jsou dostupná online, jako webové služby. Velkou výhodu u nich vidím v tom, že jsou nezávislé na platformě, kterou uživatel využívá a že data jsou uložena v cloudu, takže i při chybě hardwaru na straně klienta nedojde ke ztrátě dat. Uložení v cloudu

je však potencionálně i zranitelnost a jistá část uživatelů kvůli tomu nechce využívat služby, které ukládají jejich data online.

Jak již bylo zmíněno v úvodu, celý nápad na tuto práci vznikl původně z používání portálu iDoklad. Vzhledem k jeho robustnosti je jakýmsi vzorem i pro další online služby, které většinou nabízejí větší či menší podmnožinu jeho funkcí. Proto tedy popíšu většinu funkcí detailně právě u iDokladu a u dalších existujících řešení budu zmiňovat již jen odlišnou implementaci nebo přidané, případně chybějící, funkce.

3.2.1.1 iDoklad.cz

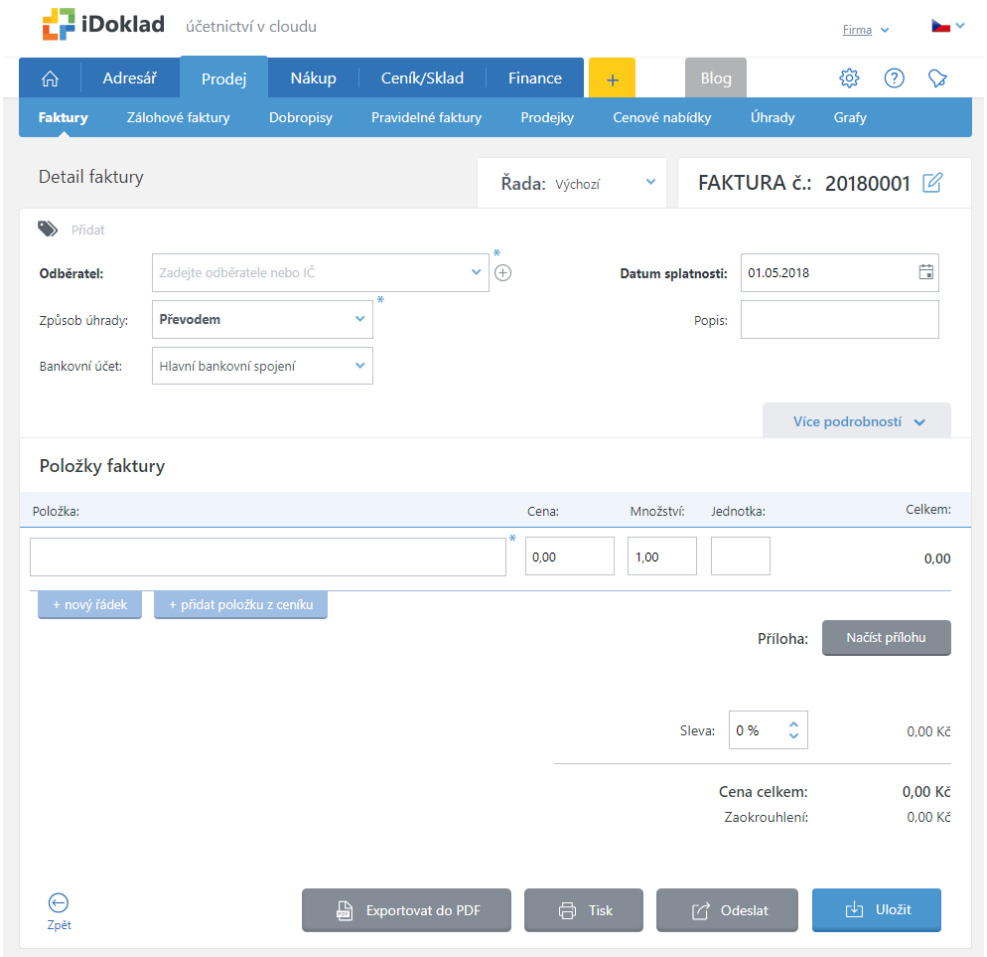
Portál iDoklad od společnosti Solitea jehož vývoj začal v roce 2010 [15] byl spuštěn pro veřejnost v březnu roku 2011 [16]. V současnosti má více než 180 tisíc uživatelů, což z něj dělá nejpoužívanější online portál na tvorbu a správu faktur na českém trhu [16].

Registrace a první přihlášení Ihned po registraci, při prvním přihlášení do systému je uživatel vyzván k vyplnění fakturačních údajů – tedy především IČO a adresy podnikání. Stačí znát pouze IČO, zbytek údajů se již automaticky nastaví podle dat získaných z registru podnikatelů ARES. Dále je možné nahrát vlastní logo a podpis, které se následně u placených tarifů přidají na fakturu.

Statistiky Statistiky a přehledy o fakturách jsou jakousi domovskou stránkou – uživatel je sem přesměrován po přihlášení a i logo v záhlaví uživatele nasměruje na statistiky. Uživatel zde nalezne sumu částky všech vystavených a přijatých faktur za různá období v podobě grafů i textových informací. Dále jsou mu zobrazeni například největší odběratelé, největší dlužníci a také suma peněz z faktur, které jsou již po datu splatnosti, rozdělené do čtyř kategorií – do 1, do 2, do 3 a nad 3 měsíce po splatnosti.

Vystavování faktury Asi nejdůležitější funkce celého systému. Uživatel je nejprve dotázán na protistranu, neboli odběratele či klienta. Údaje lze vyplnit ručně, vybrat z adresáře, je možné odběratele vyhledat pomocí jeho jména nebo nechat načíst jeho údaje z registru podnikatelů ARES – v takovém případě je potřeba znát IČO odběratele. Při vyhledávání podle jména jsou nabízeny subjekty z registru ARES a adresáře, které vykazují nějakou shodu a z nich si uživatel může vybrat. Po zadání je potřeba údaje uložit. Pokud protistrana zatím není v adresáři, pak se do něj uloží.

Nad výběrem odběratele je ještě možné faktuře přiřadit jeden či více štítků – jakýchsi kategorií nebo tagů. Umožňují uživateli faktury lépe třídit. Podle štítku se v nich následně dá i vyhledávat.



Obrázek 3.2: Screenshot rozhraní pro vystavení nové faktury v iDokladu

Dalším údajem, který se vyplňuje je datum splatnosti. Předvyplní se podle toho, jakou splatnost faktur si uživatel nastavil jako výchozí. Implicitní hodnota je 14 dnů ode dne vytvoření faktury.

Následně má uživatel možnost vyplnit svůj vlastní popis faktury. Slouží pouze pro jeho orientaci, do výsledné vygenerované faktury se nijak nepromítne, ale je používán při zobrazení přehledu vydaných faktur.

Opět podle uživatelových preferencí je doplněn způsob platby. Zde je na výběr z 6 možností – převodem, kartou, hotově, dobírkou, zápočtem, zálohou. U placených tarifů si ještě uživatel může zvolit na který z účtů uložených v nastavení má být faktura uhrazena. Pokud uživatel chce je zde ještě možnost „Více podrobností“, která umožňuje detailněji definovat platbu například zadáním variabilního nebo specifického symbolu a čísla objednávky. V tomto

detailnějším nastavení lze také změnit text před položkami faktury nebo nastavit platbu v cizí měně.

Další velmi důležitou součástí faktury je zadávání jednotlivých položek. Každá položka má název, jednotkovou cenu, množství a případně ještě jednotku. Ihned po zadání jednotkové ceny a množství se spočte celková cena položky a patřičně se též aktualizuje suma všech položek. Na začátku je zde pouze jeden řádek na zadání jedné položky, po zadání se automaticky přidá další řádek. Celé položky je možné odebírat případně měnit jejich pořadí. Je možná též přidat položku z ceníku.

V závěru je umožněno k faktuře nahrát přílohu. Tou může být libovolný soubor do velikosti 1 MB. Následuje možnost nastavit na celou fakturu slevu v procentech. Pokud si uživatel v horní části rozhraní zvolil více podrobností, může změnit i text uváděný za položkami faktury.

Na konci stránky jsou pak 4 akce, které může uživatel provést – exportovat do PDF, tisknout, odeslat nebo uložit:

- **Exportovat do PDF** – Pokud jsou všechna povinná pole vyplněna validní hodnotou faktura bude uložena a server následně iniciuje stažení PDF souboru s vygenerovanou fakturou.

- **Tisknout** – Podobně jako předchozí bod, faktura je uložena a otevře se okno s nastavením tisku viz. samostatný odstavec.

- **Odeslat** – Umožňuje uživateli fakturu rovnou odeslat odběrateli emailem. Je zde umožněno změnit předmět i tělo odesílané zprávy a zvolit zda se faktura má přiložit jako PDF dokument či má být přiložen pouze odkaz na stažení.

- **Uložit** – Uloží zadané údaje a vrátí uživatele zpět na přehled vydaných faktur.

Přehled faktur V této sekci uživatel nalezne přehled všech svých vystavených faktur. V záhlaví stránky je možné nastavit filtry tak, aby uživatel viděl například pouze nezaplacené faktury, pouze faktury ze současného kalendářního roku nebo může ve fakturách vyhledávat. Je implementována podpora stránkování a uživatel si může zvolit počet faktur na stránce z 5 nabízených hodnot.

Nad filtry se nachází nabídka nástrojů. Uživatel může vystavit novou fakturu, kopírovat již existující, upravit, vytisknout, odeslat emailem, odeslat upomínku emailem, fakturu uhradit, smazat a nebo udělat sumu přes vybrané faktury. Nástroje lze využít i nad více fakturami najednou – nástroj se vždy aplikuje na všechny faktury u kterých je zaškrtnut checkbox.

Následně je zobrazena jednoduchá tabulka s 6 sloupci – číslo faktury, popis, odběratel, cena, datum vystavení a datum splatnosti. Faktury se dají seřadit

podle každého ze sloupců vzestupně i sestupně. Zobrazené číslo faktury umožňuje po kliknutí přejít na detail zvolené faktury. U každé faktury jsou pak ještě 3 ovládací prvky – checkbox pro hromadné operace, ikonka obálky pro iniciaci odeslání faktury emailem a posledním prvkem je kolečko označující stav faktury – zda je neuhrazená, částečně uhrazená či zcela uhrazená. Přes toto kolečko je možné i přidat k faktuře úhradu a tím měnit její stav. Otevřené dialogové okno pak nabízí přehled úhrad u dané faktury – pokud například odběratel splácel faktury ve více splátkách.

Za určité zklamání považuji, že při vyhledávání se prohledávají pouze pole s číslem faktury, odběratelem a poznámkou k faktuře, ale již se neprohledávají jednotlivé položky faktury nebo například celková cena. Dalším určitým nedostatkem je, že některé nástroje fungují pouze s jednou fakturou. Není tedy například možné kopírovat více faktur najednou. Uživatel je na to však upozorněn až ve chvíli, kdy se daný nástroj pokusí použít a ne už v průběhu vybírání faktur.

Adresář Jak bylo zmíněno, pokud je vystavena faktura novému odběrateli, tento nový odběratel se automaticky přidá do adresáře. Uživatel si do adresáře může ručně přidat i další kontakty, kterým ještě nic nefakturoval.

V záhlaví adresáře je opět sada nástrojů – uživatel může přidat nový kontakt, kopírovat nebo upravit právě označený, vytisknout si seznam označených kontaktů a nebo označené kontakty smazat.

U kontaktů jsou zobrazeny 4 sloupce – název firmy, jméno u kontaktů, kterými jsou fyzické osoby, emailová adresa a telefon. Název firmy slouží k přechodu na detail zvoleného kontaktu. U fyzických osob se jako název firmy používá jejich jméno.

U každého kontaktu je kromě zřejmých údajů jako je jméno resp. název, IČO, DIČ nebo adresa možné evidovat ještě telefonní kontakt, emailovou adresu a webové stránky. Uživatel si také u každého kontaktu může nastavit individuální slevu a dobu splatnosti faktur. Detail kontaktu je rozdělen do tří záložek. První obsahuje již zmíněné údaje, druhá slouží pro uložení informací o kontaktní osobě na straně kontaktu a v poslední záložce si uživatel může nastavit informace o bankovním spojení. V první záložce je opět možné nechat data automaticky načíst z registru ARES. Na závěr si uživatel ke každému kontaktu může přidat soukromou poznámku.

Pod formulářem pro úpravu informací je zobrazen seznam veškerých faktur souvisejících s tímto kontaktem. Zobrazení je velice podobné přehledu faktur.

Ceník Pokud uživatel některé položky vkládá na faktury opakovaně, je mu pro usnadnění umožněno si takovou položku uložit do ceníku. Při tvorbě faktury si pak jen vybere položku z ceníku a nemusí vyplňovat všechny údaje znovu.

U každé položky ceníku je evidován její název, množství a jednotka, cena včetně měny a případně ještě čárový kód dané položky. V rámci ceníku je možné sledovat i stav zásob dané položky, pokud se jedná o položku s omezeným množstvím.

Nastavení Nastavení je rozděleno do několika záložek. První záložkou je „Firma“. Uživatel si zde může upravit údaje, které zadával při prvním přihlášení, vyplnit účetní údaje jako je měna ve které chce fakturovat, zda je či není plátcem DPH, uvést spisovou značku, zvolit zda reprezentuje fyzickou či právnickou osobu, příslušný finanční úřad a hlavní odvětví, ve kterém podniká. Dále má možnost si nahrát svoje logo a podpis, které se následně vkládají na fakturu. Poslední nastavení v této záložce umožňuje určit, kolik desetinných míst má na faktuře být zobrazeno u jednotkové ceny a množství.

Další záložkou v nastavení je „Uživatel“. Tato záložka slouží k změně kontaktních údajů, případně pro změnu hesla.

Následuje záložka „Prodej“, kde si uživatel, pokud je plátcem DPH, může zvolit, která sazba DPH se na jeho zboží či služby vztahuje, nastavit si konstantní symbol používaný na faktuře, dále pak výchozí platební metodu a s tím související zaokrouhlování, kde je na výběr ze tří možností – nezaokrouhlovat, zaokrouhlovat vždy a zaokrouhlovat podle typu platby, pokud jde o hotovostní platbu, zaokrouhlí se na celé koruny, pokud o platbu bezhotovostní, nezaokrouhluje se. Následuje nastavení vzhledu a číslování jednotlivých faktur, kde si uživatel může nastavit, jak se který druh faktury má číslovat, vybrat si jednu ze čtyř šablon pro vzhled faktury a její jazyk. Dále si uživatel může zvolit barvu prvků na faktuře a na závěr ještě upravit texty, které jsou vkládány před a za položky ve faktuře. Pokud uživatel využívá střední nebo nejdražší tarif, je zde možnost nastavit automatické odesílání upomínek emailem.

V záložce „Nákup“ může opět změnit, zda je plátcem daně a podle toho jsou uváděny ceny na přijatých fakturách, výchozí způsob platby a opět číslování přijatých faktur.

Následující záložka „Banka“ umožňuje nastavit uživatelův bankovní účet, který se má na fakturu vyplňovat v případě platby převodem. U středního a nejdražšího tarifu je možnost mít uloženo i více účtů a při vytváření faktury mezi nimi vybírat, uživatelé nejlevnějšího tarifu nebo tarifu zdarma mohou mít uložené pouze jedno číslo účtu. Je zde možné zapnout i automatické párování plateb – viz samostatný odstavec.

Záložka „Pokladna“ slouží pro uživatele, kteří pracují s hotovostí. Je zde možnost vytvořit několik oddělených pokladen, uvést jejich počáteční stav a číslovaní pokladních dokladů.

„E-maily“ jsou záložka, kde uživatel může nastavit zda se faktury odesílané emailem mají přidat jako příloha ve formátu PDF nebo do emailu vložit pouze odkaz na kterém si fakturu můžou zobrazit. Následuje nastavení šablon

emailů odesílaných v různých situacích a posledním nastavením je odesílací server pro emaily. Detailněji viz. samostatný odstavec.

Zbývající tři záložky „Aplikace“, „EET“ a „Import“ popíši jen zběžně. V záložce „Aplikace“ má uživatel možnost editovat seznam štítků, které následně může přiřazovat k fakturám a také si zde může zvolit barvu, kterou má iDoklad používat pro prvky v uživatelském rozhraní. Záložka „EET“ umožňuje nastavení pro podnikatele na které se vztahuje povinnost elektronické evidence tržeb. V rámci této práce se napojení na EET nebude řešit a tak nebudu ani popisovat nastavení, která jsou zde dostupná. Poslední záložkou je „Import“, kde si uživatel může naimportovat buď kontakty z aplikace Microsoft Outlook nebo z Gmailu a dalších služeb. V obou případech je potřeba nahrát soubor s kontakty. Dále si uživatel může naimportovat data ze služby *Superfaktura* a z ekonomického systému *Pohoda*.

Párování plateb iDoklad nabízí automatické párování příchozích plateb s fakturami. Funkce je podporována pouze pro 8 českých a 4 slovenské. Z českých bank jde tedy o:

- ČSOB
- Fio Banka
- Raiffeisen Bank
- Komerční banka
- Air Bank
- Sberbank
- Moneta
- UniCredit bank

Ve všech případech je párování implementováno pomocí zasílání emailů o změně stavu účtu. Uživatel si tedy v nastavení vygeneruje emailovou adresu, na kterou následně nechá z banky odesílat emaily o změnách a iDoklad tyto emaily parsuje a případně mění stav faktury, ke které pohyb na účtu přiřadí. Ostatní banky včetně například Equa bank, jejíž služby v současnosti využívám, neposílají v informačních emailech dostatek informací, aby bylo možné platbu spolehlivě identifikovat.

Tisk Z různých míst v rámci portálu je možné iniciovat tisk, ať už faktury nebo například seznamu kontaktů. Ve všech případech je uživateli vždy nejprve zobrazeno nastavení tisku s náhledem faktury. Uživatel si může následně zvolit, do kterého formátu se mají data k tisku vyexportovat. Na výběr je z 5 formátů pro export – *Adobe PDF, Excel, Rich Text Format, obrázek, webová stránka.*

Po volbě formátu si uživatel může výsledný soubor buďto stáhnout a nebo pokračovat k tisku, což již iniciuje dialog tisku přímo v prohlížeči.

Odesílání emailů Při nastavování šablon odesílaných emailů je možné změnit předmět i samotné tělo zprávy. V obojím je možné používat šablony / zástupné řetězce, kam se při generování emailu doplní údaje podle faktury, která je právě odesílána – například její číslo, celková cena nebo datum splatnosti.

Při odesílání je možné šablonu pro konkrétní fakturu ještě upravit. Zároveň, pokud u kontaktu není uložen email, je uživatel dotázán, na jakou adresu má být email odeslána. Zároveň může odeslat i kopie na vlastní emailovou adresu nebo třeba na email svého účetního.

V odstavci o nastavení bylo zmíněno, že uživatel má na výběr ze tří variant, jak mají být emaily z aplikace odesílány. Jde tedy o:

- **servery iDoklad** – k odeslání se používají přímo poštovní servery iDokladu. Email je tedy odeslán z adresy *spravce@idoklad.cz*.
- **Exchange servery** – pro uživatele využívající Office 365 Exchange nebo vlastní Exchange server. Adresa ze které se email odešle je v režii uživatele.
- **Vlastní SMTP server** – K odesílání emailu se využije SMTP server podle nastavení uživatele. Adresa ze které se emaily budou odesílat je opět na volbě uživatele.

V případě volby vlastního SMTP serveru nebo Exchange serveru je uživatel dotázán na potřebné přihlašovací údaje.

Tarify Jak již bylo zmíněno v úvodu, iDoklad byl několik let dostupný zcela zdarma, ale v červnu roku 2016 byly jeho funkce rozděleny do 4 různých tarifů [3]. Každý z tarifů se liší dostupnými funkcemi, případně limitem aplikovaným na jejich využívání a samozřejmě cenou. Zkrácené porovnání tarifů vycházející z [17] je uvedeno v tabulce 3.1. Z tabulky je jasně vidět, že funkce, které nabízí základní tarif by měly celkem uspokojit jednotlivce. Jistým omezením může být nemožnost přístupu přes API.

3.2.1.2 Fakturoid.cz

Stejně jako iDoklad nabízí i Fakturoid tarif zdarma a tři placené tarify, které jsou o něco málo dražší. Nabídka funkcí je velice podobná včetně shodné terminologie. Mně osobně žádné funkce nechyběly a naopak se mi líbily přídavky například v podobě integrace platebních bran. Je zde znatelně preferováno načítání údajů z registrů před ručním zadáváním. Tarif zdarma je zde také omezen na maximálně 5 kontaktů v adresáři, ale na rozdíl od iDokladu, zde

Tabulka 3.1: Srovnání tarifů iDokladu

Funkce / Tarif	Zdarma	„Základní“	„Oblíbený“	„Prémiový“
Cena (platba měsíčně/ročně)	0	193 / 161	338 / 281	604 / 499
Počet kontaktů v adresáři	5	∞	∞	∞
Fakturace a evidence úhrad	✓	✓	✓	✓
Štítky u faktur		✓	✓	✓
Přílohy k fakturám		✓	✓	✓
Ceník		✓	✓	✓
Mobilní aplikace a QR platba		✓	✓	✓
Štítky u faktur		✓	✓	✓
Vlastní logo a barevné schéma		✓	✓	✓
Více uživstelů jednoho účtu		✓(2)	✓(3)	✓(9)
Banka a párování plateb			✓	✓
Textová podpora	✓	✓	✓	✓
Telefonická podpora		✓	✓	✓(přednostní)
Měsíční limit API dotazů	0	0	2500	7500

je i u tarifu zdarma možný přístup přes API, které je sice i zde limitováno, ale s řádově vyššími limity než na iDokladu.

Na rozdíl od iDokladu je aplikace méně interaktivní, což však považuji spíše za pozitivum. iDoklad má spoustu ovládacích prvků, jako třeba nabídku akcí s danou fakturou, svázaných s najetím myši nad fakturu v přehledu, což dle mého názoru ubírá přehlednosti, jelikož na uživatele často vyskakují menu, která ani otevřít nechtěl. Fakturoid má tedy z mého pohledu více statický, ale přehlednější a čistší design aplikace. Podobně je tomu i s fakturou. Oproti iDokladu je velice strohá a designově prostá. Placené tarify slibují možnost volby vzhledu faktur, párování plateb nebo třeba ceník, podobně jako u iDokladu.

Funkce, která mě u Fakturoidu trochu zklamala byla přidávání vlastního loga a podpisu. Udělal jsem nejprve fakturu, pak jsem do systému přidal logo a podpis a když jsem chtěl fakturu vygenerovat znovu, tak ač se v náhledu logo i podpis zobrazovaly, tak ve výsledné faktuře nebyly. Při vytvoření další faktury už vše fungovalo podle očekávání. Trochu zvláštní je i samotné přidávání vlastního loga a podpisu do aplikace. Ta má zřejmě pro logo a podpis na faktuře vyhrazené obdélníkové rámečky s poměrem stran zhruba 3:1, ale pokud má uživatel například čtvercové logo nebo kratší podpis, je již v náhledu zobrazen na levém okraji dlouhého bílého obdélníku, což působí poněkud nezvykle. Mou poslední výtkou je subjektivně nižší přehlednost u stránky s přehledem faktur.

Celkově však Fakturoid působí jako velice zdařilá obdoba iDokladu, která pro mě osobně působí příjemněji a přehledněji než iDoklad samotný.

3.2.1.3 SuperFaktura.cz

SuperFaktura na rozdíl od předchozích existujících řešení nemá tarif zdarma, pouze tři placené tarify, které si člověk před koupí může zdarma vyzkoušet.

Cenově vychází podobně jako iDoklad. Funkce jsou opět podobné, chybělo mi pouze načítání údajů z registru v nastavení uživatele. Jako zajímavý přídavek mě překvapila možnost odesílání faktur poštou v papírové podobě a odeslání připomínky na nezaplacenou fakturu pomocí SMS zprávy přímo z aplikace. Vygenerovaná faktura je o něco přehlednější než například u Fakturoidu (viz 3.2.1.2), ale sazba obrázků by mohla být zvládnuta lépe.

Za výhody považuji již zmíněné SMS a klasickou poštu, dále oceňuji robustní podporu API, které je propagováno znatelně více než třeba u iDokladu. K API jsou zde návody, dokumentace, hotové knihovny pro přístup z různých programovacích jazyků a adaptéry pro integraci API Superfaktury do různých účetních systémů a e-shopů. Příjemná drobnost je možnost stažení více faktur najednou jako ZIP archiv.

API je bohužel dostupné pouze u nejdražšího tarifu, není však omezen počet dotazů na něj. Logo uživatele je na faktuře umístěno poměrně blízko linkám, které mají vizuálně dělit fakturu. QR kód pro platbu je umístěn taky velmi blízko ostatním prvkům a minimálně moje bankovní aplikace hlásila chybu při pokusu o jeho načtení. Při nastavování údajů o uživateli jsem trochu postrádal možnost načíst informace z registru ARES, při vystavování faktury novému subjektu již načítání údajů z registru funguje.

3.2.1.4 mPohoda.cz

Zde se nejedná o zcela samostatnou aplikaci, ale spíše o webový doplněk desktopové účetní aplikace Pohoda, což jde poznat i podle toho, že některé funkce jsou uživateli zpřístupněny pouze pokud vlastní licenci na desktopovou Pohodu. Nicméně mPohoda je zdarma, nenabízí žádné placené tarify a neaplikuje ani omezení na počet subjektů v adresáři. Při prvním spuštění je potřeba vyplnit poměrně velké množství nastavení než je uživateli umožněno vytvořit první fakturu. Celá aplikace je vizuálně velmi jednoduchá. Dominuje jí až zbytečně velká hlavička, kde se zobrazuje vždy jen název sekce, kde se uživatel právě nachází. Aplikace mi celkově přišla hůře přehledná, zvláště pokud uživatel vstoupí do sekce, která je prázdná, v takové situaci na mě aplikace vždy působila dojmem, že se něco nepodařilo načíst – dokud jsem si vždy nepřečetl tu jednu zobrazenou řádku textu, kde se psalo, že v dané sekci se nepodařilo nic najít.

Za pozitivní považuji, že je k dispozici zdarma a neaplikuje žádný limit na počet odběratelů. Vygenerovaná faktura je dobře vizuálně rozdělená pomocí čar, ale texty jsou často velmi blízko těmto čárám. Načítání údajů z ARES je zde podporováno jak v nastavení uživatele tak při zadávání nového subjektu. Určitou výhodou může být i nabídka exportu faktury do XML. Tím však pozitiva končí. Oproti ostatním existujícím řešením trochu zaostává nabídkou funkcí a ty, které má, jsou často nedotažené.

Za první velkou nevýhodu považuji absenci API, za další pak úplnou absenci QR platby na faktuře. Nejde zde provádět žádné hromadné operace

s fakturami a co mi přišlo obzvlášť nepříjemné je nemožnost fakturu stáhnout přímo z přehledu vytvořených faktur. Uživatel musí vždy přejít do detailu faktury a až zde jsou tlačítka pro stažení. Nedotažené mi přišlo i odesílání faktur emailem. Nejenže nejde do šablony vkládat zástupné řetězce, které by se nahradili údaji z faktury, ale pokud chcete odeslat fakturu z detailu faktury a u subjektu není v adresáři vyplněna emailová adresa, není možné fakturu odeslat. Uživatel je pouze vyzván aby emailovou adresu doplnil ke kontaktu v adresáři, následně nechal v detailu faktury znovu načíst údaje o subjektu a fakturu uložil. Až pak je mu umožněno odeslat ji emailem, nemá však již možnost zasáhnout do podoby odeslaného emailu.

3.2.1.5 Trivi

Společnost Trivi se primárně zabývá kompletním zpracováním účetnictví pro OSVČ a firmy a systém na správu faktur je spíše okrajovou záležitostí, přesto rozhodně stojí za zmínku. Je provozován zdarma pro neplátce DPH a fyzické osoby bez IČO a není nijak omezen množstvím kontaktů v adresáři. Registrace do systému je poměrně zdlouhavá a po jejím dokončení je potřeba čekat, než vám systém účet opravdu vytvoří a dovolí se poprvé přihlásit. Na Trivi je hodně poznat, že je opravdu robustním systémem a tak je všude nepřeberné množství nastavení, přepínačů, seznamů a políček, ve kterých se uživatel snadno ztratí. Bohužel zde není nijak zvýrazněno, která políčka jsou například nutná a která dobrovolná. Velikostí i popiskem působí všechna políčka stejně důležitě a uživatel se v nich snadno ztratí. Z existujících řešení mi přijde asi nejméně přehledný, je poznat, že jde o součást systému, který je schopný obsluhovat i nadnárodní korporace, ale pro uživatele, který potřebuje jen vystavit několik málo faktur ročně je až zbytečně složitý.

Za výhodu, kromě neomezeného používání, považuji i přítomnost API, které je podobně jako celá aplikace velmi robustní a lze přes něj udělat prakticky cokoliv. API má i velmi dobře zpracovanou a přehlednou dokumentaci. V aplikaci je implementována většina funkcí, které nabízí ostatní existující řešení, jen je občas trochu těžší je najít a správně použít.

V aplikaci se mi nepodařilo najít ceník, bylo tedy pokaždé nutné jednotlivé položky faktury zadat znovu. Nastavení uživatele mi přišlo velmi nešťastně uděláno formou rozbalovacích sekcí. V každé části nastavení je několik sekcí, kdy první je při příchodu automaticky rozbalena. Není zde však žádné upozornění na další sekce, které jsou většinou pod okrajem okna a uživatel, pokud se k těmto sekcím nastavení chce dostat, musí buď zavřít sekce nad nimi nebo k nim doscrollovat. Při vytváření nové faktury je uživatel nucen zvolit, do jaké „účetní kategorie“ vytvářená faktura spadá, k čemuž zde není žádná nápověda. Ostatní systémy žádnou takovou informaci nevyžadují. Poslední výtkou je, že stejně jako v mFaktura nejde ani v Trivi vytisknout nebo odeslat fakturu z přehledu faktur, ale až z detailu faktury.

3.2.1.6 iÚčto.cz

iÚčto.cz volí mírně odlišnou strategii jak omezit tarif zdarma a to 100 evidovaných dokladů, což pro opravdu drobné podnikatele může bez problémů vystačit na několik let. Aplikace mi přišla rozumně přehledná a i nabídka funkcí je víceméně shodná s ostatními řešeními. Placené tarify jsou zde jako obvykle tři a cenově vycházejí mírně dražší než iDoklad.

Za výhodu považuji funkce bez omezení i u tarifu zdarma – pro uživatele, který opravdu vystaví jen několik málo faktur ročně to může být řešení na mnoho let, než dosáhne na limit 100 evidovaných faktur. Další výhodou je přístup přes API u všech tarifů, nepodařilo se mi zjistit, jestli je nějak omezen počet dotazů na API. Vzhled vygenerované faktury mi přijde vyhovující, přesto, že zde nejsou žádné dělící čáry texty jsou od sebe většinou dostatečně odděleny a člověk se při čtení neztratí.

Za nevýhodu považuji občas nepřehledné nastavení – například párování plateb s fakturami jsem v něm najít nedokázal, ale podle seznamu funkcí na webu služby to aplikace umí. Dalším spíše drobným nedostatkem mi přišla volba ikon pro akce s fakturou, konkrétně v přehledu faktur jsou u faktury zobrazeny tři shodné červené křížky vedle sebe, kde každý provede jinou operaci a pouze první (smazat) mi přijde relevantní k červenému křížku. Zbývající dva křížky umožňují zaúčtovat přijatou platbu k faktuře a k takové akci se podle mě červený křížek nehodí. Jako poslední jsem si všiml, že pokud není nastaven text pro zápatí faktury, tak se upozornění na jeho neexistenci ve znění „*Text není nastaven! Doplňte prosím v Nastavení → Profil firmy*“ dostane i na výslednou vygenerovanou fakturu.

3.2.1.7 Vyfakturuj.cz

Služba Vyfakturuj.cz je u tarifu zdarma omezena, podobně jako většina existujících řešení, na 5 kontaktů v adresáři . Dále nabízí 4 placené tarify, které jsou cenově velmi podobné iDokladu až na první, který je za necelou stokorunu. Zajímavou funkcí u placených tarifů je, podobně jako u SuperFaktury, možnost odeslání faktury v papírové podobě poštou.

Vygenerovaná faktura je designově jednodušší, ale přijde mi dobře přehledná. Jedinou výtku bych měl k umístění některých textů – například popisek „Neplátce DPH“ bych očekával spíše u informací o subjektu než u čísla faktury. Zbytek systému mi přijde celkem intuitivní a dobře přehledný.

Trochu mě zklamalo, kolik funkcí je dostupných pouze v placených tarifech. Na rozdíl od ostatních řešení zde v tarifu zdarma není dostupný ceník, přístup přes API ani automatické párování plateb s fakturami. Pokud uživatel tyto funkce potřebuje, je odkázán na jeden ze dvou nejdražších tarifů.

3.2.1.8 Vyfakturuj.to

Ač se názvem velmi podobá předchozímu řešení, jedná se o zcela jinou služby. Oproti ostatním je dostupná poměrně krátkou dobu – teprve od začátku roku 2018, což je poznat i při používání aplikace. Je provozována zdarma, bez omezení, ovšem nabídka funkcí je oproti ostatním existujícím řešením poměrně strohá. Nejsou zde ani žádné placené tarify, nabízející nějaké rozšíření sady dostupných funkcí.

V podstatě jedinou výhodou Vyfakturuj.cz je dostupnost zdarma bez omezení. Tím ale výhody bohužel končí a oproti konkurenci systém značně zaostává.

Výčet nedostatků této služby je bohužel celkem dlouhý. Za nejvýraznější nevýhodu považuji nutnost téměř vše vyplňovat ručně. Aplikace nedoplní číslo faktury, našeptávání položek z ceníku funguje pouze v případě, že uživatel zadá přesný název položky a načítání údajů z registru ARES není dostupné ani při vytváření nového kontaktu, ani v nastavení údajů o uživateli. Vygenerovaná faktura mi přijde hůře přehledná, nejsou zde vizuální prvky, které by na první pohled a jasně oddělovaly sekci s položkami od sekce se součtem a podpisem. Jednotlivé položky na faktuře také nejsou nijak odděleny, což uživateli zhoršuje orientaci v řádcích. Použité logo společnosti mi přijde až zbytečně velké a umístěné až příliš blízko ostatním textům na faktuře. V nastavení je poněkud neintuitivní, že uživatel musí každou část vyplnit a uložit zvlášť, jinak o vyplněné údaje přijde. Na stránce s nastavením jsou sice 3 tlačítka pro uložení, každé z nich však uloží pouze sekci, ke které je přiřazeno a data z ostatních sekcí se bohužel ztratí. Na závěr již jen uvedu, že na vygenerované faktuře není QR platba, aplikace vůbec nenabízí přístup přes API a není zde možnost odesílat faktury emailem.

3.2.1.9 FAKTURYON

Služba Fakturyon se nijak netají tím, že cíli spíše na menší podnikatele. Nenabízí žádné služby zdarma, pouze možnost vyzkoušet si nabízený tarif na 30 dnů zdarma. Placený tarif je zde pouze jeden za 50 Kč měsíčně při platbě alespoň na rok dopředu. Ač je služba poměrně levná, nabídkou funkcí obstojně konkuruje i mnohem dražším řešením. Službu bych se nebál doporučit jako levnou alternativu iDokladu, která nabízí většinu jeho funkcí.

Aplikace nabízí v podstatě stejnou sadu funkcí jako ostatní existující řešení. Najdeme zde tedy načítání údajů z registru ARES, ceník, QR platbu, automatické párování plateb, vlastní podpis i logo na faktuře, API atd. Aplikace je celkově dobře přehledná a intuitivní. Stejně tak i vygenerovaná faktura je velice dobře přehledná, troufnu si říct, že možná nejlépe ze všech analyzovaných řešení.

Fakturyon má z mého pohledu jen několik málo nedostatků. Jedním z nich je, že subjekty, kterým byla vystavena faktura se automaticky neukládají do

adresáře. Pokud tedy chce uživatel používat subjekt opakovaně, musí si jej ručně vložit do adresáře. Další, ale opravdu drobná výtka míří na vygenerovanou fakturu, kde mi přijde, že je text vysázen až příliš blízko k okraji stránky. Trochu bych zapřemýšlel i nad rozmístěním prvků na stránce – například při vytváření nové faktury musí uživatel i na větším monitoru scrollovat dolů, aby mohl zadat položky, ale tlačítko na uložení faktury je v pravém horním rohu a tak musí vždy scrollovat zpět nahoru, aby fakturu mohl uložit. A posledním nedostatkem je API, které je sice podle informací na stránce dostupné, ale neexistuje k němu zatím žádná dokumentace, takže je poměrně těžké, spíše nemožné jej v aktuální situaci použít.

3.2.1.10 sÚčto.cz

Další existující službou je sÚčto.cz, které je opět názvem podobné jiné službě – iÚčto, ale jde o dvě různé služby. Obě však aplikují podobné pravidlo pro omezení tarifu zdarma – počet dokladů. Zatímco iÚčto má limit nastaven na 100 dokladů, sŮčto jich umožňuje uložit jen 60. Je nabízen pouze jeden placený tarif za 300 Kč měsíčně bez DPH. Systém je celkem intuitivní, ale občas je vyžadováno až příliš mnoho nastavení, která ostatní řešení buď nevyžadují, nebo jej dokáží odvodit automaticky.

Za jistou výhodu považuji, že i v rámci tarifu zdarma je umožněno přístup k jednomu účtu dvěma různým uživatelům. Uživatel tedy může účetnictví zpřístupnit i svojí účetní bez nutnosti ji sdělovat svoje přihlašovací údaje. Jinak aplikace nabízí podobnou sadu funkcí jako ostatní existující řešení jako je načítání údajů z registru ARES, vlastní logo a podpis či QR platbu na faktuře.

U aplikace mě poněkud zklamalo, že je často nutné zadávat ručně i údaje, které lze dopočítat z jiných zadaných údajů – například kód IBAN, který jde zjistit ze zadaného čísla účtu a kódu banky. Bez zadání IBAN pak nefunguje QR platba. Při tvorbě faktury není dostupný ceník, všechny položky je tedy potřeba zadávat ručně a trochu překvapivé je i to, že aplikace očekává zadání ceny položky bez DPH. Vygenerovaná faktura jako celek je rozumně přehledná, texty mi však opět přijdou vysázené velmi blízko okrajům a v tabulce se seznamem položek faktury nejsou nijak odlišeny řádky, což zhoršuje její čitelnost. Poslední výtkou jsou pak trochu otravné notifikace. Ačkoliv jsem při registraci a následně v nastavení vypnul zasílání veškerých emailů, stejně mi do emailové schránky dorazila zpráva o tom, že jsem vytvořil novou firmu a o několik minut později i že jsem vytvořil první fakturu.

3.2.1.11 fakturaonline.cz

Trochu odlišnou službu nabízí fakturaonline. Nenabízí žádný tarif zdarma, pouze možnost si na týden vyzkoušet dražší z placených tarifů. Placené tarify jsou dva – levnější za 20 kč a dražší za 40 korun měsíčně. Levnější z tarifů

nabízí pouze jednorázové vystavení faktury, není možné se k ní později vrátit a stáhnout ji znovu nebo upravit, lze ji pouze uložit v PDF nebo odeslat emailem a pak jsou zadané informace zapomenuty. Dražší tarif nabízí archiv vystavených faktur, možnost jejich úpravy a pamatuje si i další informace jako je číslo faktury, bankovní spojení, logo a podpis atd.

Za určitou výhodu lze považovat, že veškeré nastavení probíhá na jedné stránce, uživatel se nemusí nikam přepínat a následně se vracet. Ač je systém hodně minimalistický, nabízí slušnou paletu funkcí. Nechybí tedy ani načítání údajů z registru ARES, QR platba na faktuře, vlastní logo a podpis nebo možnost vybrat si vzhled faktury.

Za nevýhodu, hlavně u levnějšího tarifu, považuji zapomínání faktur. Pokud například uživatel udělá na faktuře překlep, na který ho následně klient upozorní, musí celou fakturu vytvořit znovu. Ač je zde možnost ukládat kontakty, jde uložit v podstatě pouze základní informace načtené z registru ARES, není tedy možné uložit si u subjektu telefonní číslo, email nebo další kontaktní údaje. Dražší tarif si pamatuje informace zadané do předchozích faktur. Jednou z nich jsou i položky, které jsou našeptávány při jejich přidávání na novou fakturu. Našeptáván je však pouze název položky a při jejím zvolení není doplněna její cena. Seznam nabízených položek navíc není úplný. Vygenerovaná faktura je rozumně přehledná, sekce jsou poměrně jasně odděleny. V rámci tabulky s položkami opět není nijak podpořeno řádkování a tak se uživatel v položkách faktury snadno ztratí, zvlášť pokud jich je více. Logičtější by mi přišlo i prohodit pozici odběratele a dodavatele tak, aby se logo zadané dodavatelem nezobrazovalo u odběratele. Na vygenerované faktuře se mi stále zobrazoval nápis „plátce DPH“, přesto, že plátcem nejsem. Nápisu se mi nepodařilo zbavit ani podle instrukcí uvedených ve FAQ. Posledním nedostatkem, který si však nedokážu vysvětlit, je, že po celou dobu pobytu na této stránce běží procesor mého počítače na 100 %, po zavření stránky se využití procesoru vrátí na obvyklých cca 10 %.

3.2.1.12 faktura-pdf.cz

Poslední analyzovanou online službou je faktura-pdf. Analýza je zde poněkud ztížena faktem, že aplikace nenabízí žádnou možnost si její služby vyzkoušet. Popisované informace tedy budou pouze z toho, co lze najít na jejím webu. Nabízí dva tarify, levnější za 30 korun měsíčně a dražší za 100. Celý design webu působí poměrně zastarale, podobá se stránkám z přelomu tisíciletí.

Výhody služby hledám těžko. Podle webu nenabízí oproti ostatním existujícím řešením nic nového a bez možnosti si službu vyzkoušet nedokážu další výhody posoudit. Podle informací by aplikace měla podporovat načítání adresy z ARES. Vlastní logo na faktuře, ale podpis není zmíněn, předpokládám tedy, že není možné ho na fakturu přidat. Dražší tarif by měl nabízet i „xml komunikaci“, předpokládám tedy nějakou formu aplikačního rozhraní.

Nevýhod je oproti tomu několik. Jako asi nejvýraznější mi přijde právě nemožnost si aplikaci vyzkoušet. Design stránky působí celkem zastarale a tak bych se jako uživatel bál rovnou zaplatit za službu, u které nevím, co od ní očekávat. Další významnou nevýhodou je nutnost platit dražší tarif, aby uživatel mohl upravovat již vystavenou fakturu. Celkem nepříjemné je i to, že faktury jsou v systému uloženy pouze rok od posledního zaplaceného měsíce jednoho z tarifů, pak jsou nenávratně smazány i pokud uživatel začne znovu platit. Kromě absence podpisu na faktuře není zřejmě přítomna ani QR platba.

3.2.2 Offline

V druhé části analýzy existujících řešení se zaměřím i na několik desktopových aplikací. Určitá část uživatelů pro tvorbu faktur používá například Microsoft Excel, ale já se zaměřím pouze na aplikace určené přímo na práci s fakturami. Nabídka funkcí u těchto aplikací bude zřejmě mírně odlišná, pokusím se je však analyzovat z podobného pohledu jako online existující řešení. Na rozdíl od online řešení lze očekávat, že u desktopových aplikací bude odlišný model platby za službu.

3.2.2.1 ProFact

Desktopová aplikace ProFact je dostupná ve třech variantách. První je zdarma ale přesto nabízí většinu funkcí, které online řešení nabízejí většinou za peníze. Jedinou funkcí, kterou by uživatel asi mohl postrádat je ceník. Zbývající varianty za 990 a 3790 korun již poskytují komplexní evidenci, která by mohla stačit i střední firmě.

Zdarma tedy uživatel dostane většinu funkcí, za kterou u online řešení musí platit. Pokud je uživatel připojen k internetu, funguje zde načítání informací z registru ARES, vygenerovaná faktura obsahuje QR platbu, uživatel si může do programu vložit logo a podpis, které jsou následně používány na vygenerovaných fakturách. Stejně tak je nabízeno i odesílání faktur emailem, které probíhá přímo z aplikace ProFact.

Nevýhod jsem zaznamenal pouze několik. Za určitou nepříjemnost považuji velikost textů v aplikaci. Veškeré texty, políčka a tlačítka jsou poměrně malá a bohužel se mi v nastavení nepovedlo najít způsob, jak jejich velikost změnit. Pro uživatele s horším zrakem to může být poměrně problém i proto, že v desktopové aplikaci nejde písmo zvětšit tak snadno jako ve webovém prohlížeči. Aplikace měla problém při vložení loga ve formátu PNG s průhledným pozadím. Místo loga se vložil pouze černý čtverec, který se následně dostal i na vygenerovanou fakturu.

3.2.2.2 AdmWin

AdmWin je nabízen v několika různých variantách a uživatel si vždy stahuje jen program, který obsahuje určitou sadu funkcí. Nejde tedy o jeden ucelený software, jehož nabídka funkcí by se upravila podle zakoupené licence. Při změně je potřeba si ze stránek společnosti stáhnout novou verzi. Základní varianta umožňující tvorbu faktur a jejich export do PDF je zdarma. Celá aplikace je hodně o oknech, program otevírá nové okno v podstatě na jakoukoliv operaci a tak se snadno stane, že je otevřeno třeba 5 oken přes sebe. Vzhledově i použitými ikonami mi aplikace připomíná Windows XP.

Oproti online řešením je u aplikace ProFact nabídka funkcí poměrně strohá. Z obvyklé sady je tu pouze ceník a QR platba. Funkce jako je načítání údajů z registru ARES, vlastní logo nebo podpis zde chybí. Aplikace mi celkově nepřišla moc intuitivní, nabízí nepřeberné množství kolonek a tlačítek ve kterých se uživatel snadno ztratí a často není jasný ani jejich význam. Navíc často jdou různé části vyplňovat až po uložení předchozích částí, například položky faktury lze zadat až po uložení odběratele. Vygenerovaná faktura nepůsobí příliš přehledně, jednotlivé sekce jsou hraničeny tenkým rámečkem, ale každá sekce je jinak široká a vysoká což přehlednosti nepřidá. Posledním nedostatkem, který jsem zaznamenal je že na vygenerovanou fakturu se zobrazila QR platba přesto, že jsem nezadával informace o účtu. Údaje v ní byly v podstatě samé nuly, elegantnější řešení by bylo buď QR platbu na fakturu vůbec neumisťovat, pokud nejsou známy potřebné údaje, nebo alespoň ponechat neznámé údaje prázdné, tak jak to udělala aplikace ProFact.

3.2.2.3 Fakturky

Posledním zástupcem desktopových aplikací na správu faktur jsou Fakturky. Design aplikace je podobný aplikacím na Windows 95. Nabízí možnost vyzkoušení zdarma, kde jsou ovšem i základní funkce hodně omezeny a nebo za 710 korun plnou verzi. Aplikace opět funguje na bázi oken, kterých opět může být najednou otevřeno poměrně hodně.

Ve verzi zdarma k vyzkoušení nelze na fakturu vložit vlastní logo ani podpis, ale především v ní není povolen ani export faktur do PDF, což je pro tuto práci stěžejní vlastnost. Jedná se o první aplikaci, která tuto funkci jakkoliv omezuje. Fakturky nepodporují načítání údajů z registru ARES, u faktury nejde dodatečně vložit QR platba a okna aplikace se neaktualizují. Má li uživatel otevřený přehled faktur a vytvoří novou fakturu, musí znovu otevřít seznam faktur, aby ji v něm viděl. Celkově mi aplikace přišla nepřehledná a na dnešní standardy uživatelských rozhraní zastaralá.

3.3 Shrnutí kapitoly

V rámci dotazníku i diskuze na Facebooku bylo uživateli zmíněno ještě několik dalších funkcí, které využívají – například „nabídkový systém nebo export a tisk štítků pro přepravní společnosti“. Šlo však pouze o jednotlivce a tyto funkce tedy pravděpodobně nebudou masově využívané. Na druhou stranu je pravda, že dotazník bohužel zodpovědělo pouze 45 respondentů, což není příliš velký vzorek a navíc je potřeba určitou váhu přikládat i tomu, že dotazník byl šířen pouze online a to většinou po sociálních sítích. To je myslím poznat i u otázky na věk respondentů, kde všichni uvedli, že jejich věk je do 40 let.

Další určitou nepříjemností je, že směrnice PSD2, která by měla značně zjednodušit přístup k datům o platbách, sice již vstoupila v platnost, ale na její reálné nasazení mají banky čas zhruba do poloviny roku 2019. Některé banky, jako například Equa Bank, již API nabízejí [18], avšak pro přístup k němu je potřeba platná licence od ČNB, jejíž získání je zřejmě nad rámec této práce, ja bylo zmíněno v 2.3.

Kapitola 4

Návrh

Z analýzy, především tedy z dotazníku vyplynulo, že funkce jako je export do PDF, načítání údajů z registru ARES nebo vlastní logo a podpis na fakturách jsou hojně využívány a tak by je navrhovaná aplikace určitě měla podporovat. Naopak poměrně malá část uživatelů, využívajících online systémy pro správu faktur, využívá i EET.

Podobně malé procento uživatelů využívá online systémy pro evidenci nákladů, tedy přijatých faktur a účtenek a také možnost spravovat skladové zásoby není příliš využívaná. Přisuzuji to tomu, že podobně jako u EET, kde jsou hotová řešení zaměřená na konkretní problematiku je i u skladu vhodnější použít nějaký specializovaný skladový systém než k tomuto účelu využívat online nástroj na správu faktur.

Přesto jsem se rozhodl některé z funkcí, které nejsou příliš využívané do návrhu systému zahrnout. Jde především o přístup přes API a pak také ceník. V rámci této práce se nebudu zabývat ani rozšířením systému pro plátce DPH, přesto, že čtvrtina uživatelů v dotazníku se vyjádřila, že jsou plátci DPH.

Systém jsem se rozhodl implementovat v jazyce PHP, který je v online prostředí velmi rozšířen a především je součástí základní nabídky u většiny českých webhostingových společností, na rozdíl od ostatních technologií. Většinu dat aplikace bych chtěl ukládat do databáze, pouze konfiguračních proměnné budou ukládány přímo v souborech na webhostingu.

4.1 Úložiště

V rámci aplikace rozlišuji dva druhy dat, které je potřeba perzistentně uložit. Jedná se o konfigurační soubory a pak o samotná data uživatelů a další aplikační data, která je možné uložit do databáze.

4.1.1 Konfigurační soubory

Základní konfigurační konstanty aplikace, jako jsou přihlašovací údaje k databázi nebo emailové schránce budou ukládány do konfiguračního souboru přímo v kořenové složce webu. Jako vhodný formát se jeví **ini**, které je podle mě pro dobře čitelné i pro člověka a zároveň je jeho syntaxe jednoduše pochopitelná i pro nezkušeného uživatele.

```
[database]
host = "db.example.org"
```

Další výhodou je, že PHP pro jeho načítání poskytuje funkci. Nastavení je tedy ukládáno do souboru **config.ini** umístěného v kořenovém složce.

Bohužel, některé webhostingy ve výchozím nastavení dovolují soubory s příponou `.ini` bez problémů číst a je potřeba toto chování změnit pomocí nastavení v souboru `.htaccess` (viz 5.1.1). Pokud uživatel z nějakého důvodu nechce nebo nemůže do tohoto souboru zasahovat, bude mu umožněno uložit základní konfiguraci aplikace i dalším způsobem a to do souboru **config.php**, který by neměl jít přečíst na žádném webhostingu. Syntaxe je zde o něco složitější a pro uživatele i méně přehledná. Nastavení probíhá pomocí přiřazování hodnot do vícerozměrného pole například takto:

```
$config["database"]["host"] = "db.example.org";
```

Kde první index pole je shodný se jménem sekce v `ini` souboru a druhý index odpovídá názvu konkrétní proměnné v rámci této sekce.

Konfigurační data jsou načítána z obou souborů najednou, avšak pokud je v každém souboru nastavena jiná hodnota konfigurační proměnné, je **preferována hodnota ze souboru .ini**. Teoreticky je možné obě metody kombinovat, v rámci přehlednosti bych to však nedoporučoval. V rámci aplikace se ke konfiguračním proměnným přistupuje pomocí třídy `Configuration`, která implementuje rozhraní `ArrayAcces` [19], které umožňuje ke konfiguračním proměnným přistupovat jako k prvkům pole.

4.1.2 Data aplikace

Veškerá data aplikace jsou uložena do databáze. Přístup k databázi je řešen pomocí databázového rozšíření jazyka PHP – **PDO** (PHP Data Objects) [20]. Toto rozšíření usnadňuje integraci databáze do aplikace, jelikož nabízí jednotné rozhraní pro komunikaci s databází. Jaká databáze bude ve finále použita již nemá vliv na zdrojový kód aplikace, stačí aby pro tuto databázi byl dostupný PDO ovladač. Ty jsou podle [21] dostupné pro většinu běžně používaných databází, jako je MySQL, MS SQL Server, PostgreSQL nebo Oracle.

Jelikož se v aplikaci poměrně často pracuje s uživatelským vstupem, jsou při veškeré komunikaci s databází používané „připravené dotazy“ (Prepared

statements), které můžou mít pozitivní vliv na rychlost komunikace s databází, ale především předcházejí útokům známým jako **SQL Injection** [22].

Tabulky v databázi jsou vzájemně provázány pomocí cizích klíčů. To vytváří jisté požadavky na konzistenci databáze, tak aby nevznikaly osiřelé záznamy jako jsou například položky faktury, která již neexistuje. Tato integritní omezení však nejsou hlídána na úrovni databáze, ale na aplikační úrovni.

4.1.3 Entity

Navrhovaná aplikace má dvě základní entity – fakturu a uživatele, neboli **Invoice** a **User**. Na ně jsou pak navázány další entity jako například subjekty, bankovní účty, ceník, atd. Každá z těchto entit je uložena do samostatné tabulky v databázi (viz databázový model 4.1).

4.1.3.1 User

Entita **User** reprezentuje jednoho konkrétního uživatele aplikace. Stejně jako všechny ostatní entity je identifikován pomocí unikátního `id`, které slouží zároveň jako primární klíč v databázové tabulce.

Dalším atributem je `email`, reprezentující emailovou adresu uživatele. I ta by měla být unikátní, jelikož při registraci uživatele se kontroluje, zda shodná adresa již není přítomna v databázi. V takovém případě je registrace zamítnuta.

Následují atributy `name` a `surname`, které zjevně reprezentují jméno a příjmení uživatele. Pokud je uživatelem společnost, předpokládá se, že celý její název bude uložen v atributu `name` a atribut `surname` zůstane prázdný.

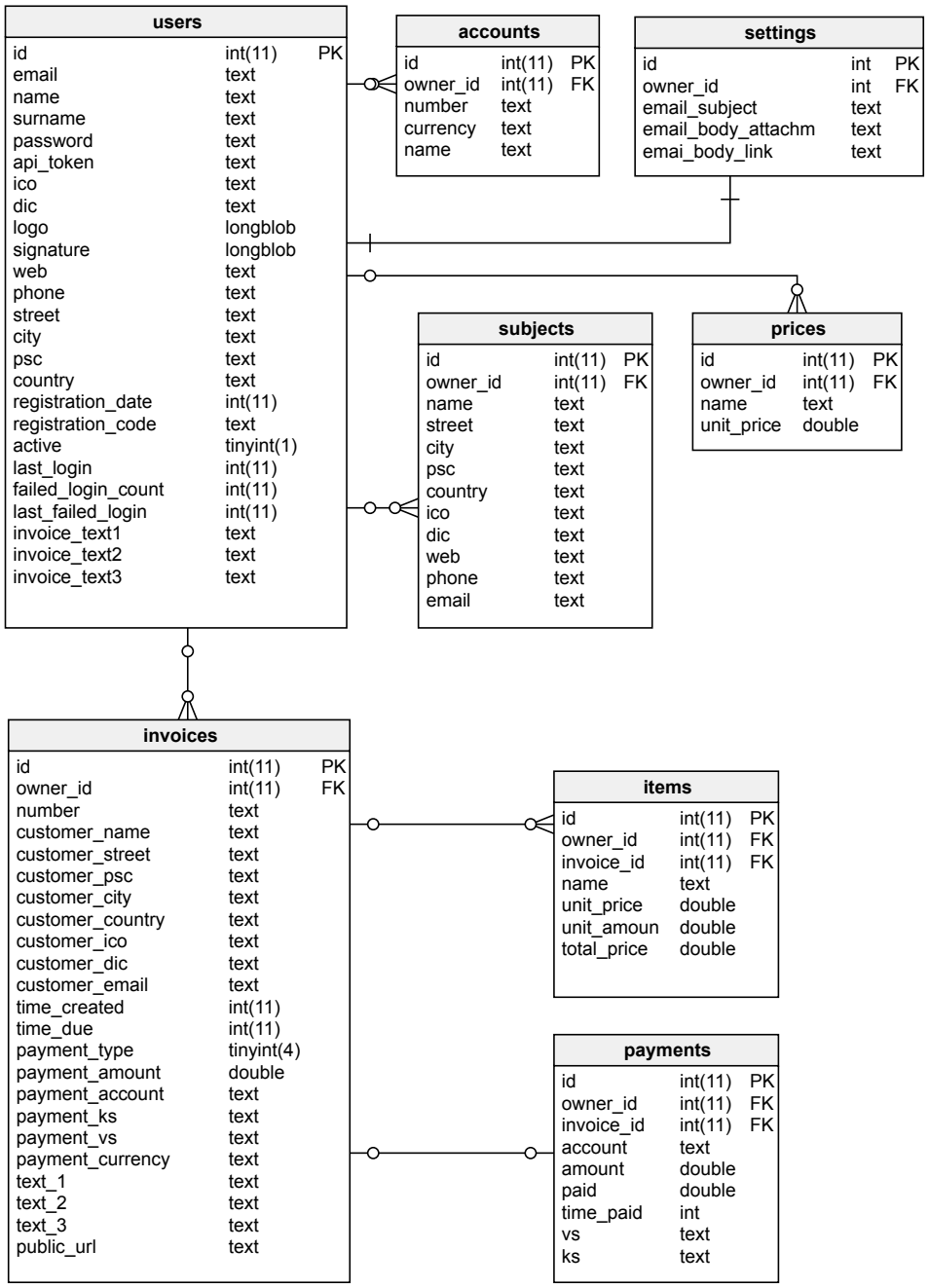
Atributy `password` a `api_token` slouží pro autentizaci uživatele. Ve sloupci `password` je uložen hash hesla, které uživatel při registraci zadal. Veškerá autentizace v aplikace probíhá pomocí *api_tokenu*, heslo je tedy používáno pouze pro jeho získání. *Api_token* je uživateli vygenerován při registraci a je uložen ve sloupci `api_token`. Na žádost uživatele mu je vygenerován nový *api_token*.

`ico` a `dic`, jak už název napovídá, slouží k uložení identifikačního čísla osoby (IČO) a daňového identifikačního čísla (DIČ). IČO má přidělené každý podnikatel, DIČ je přiděleno, pokud se podnikatel dobrovolně přihlásí jakožto plátce daně nebo pokud ho k tomu donutí splnění zákonem daných podmínek. IČO tedy bude vyplněné vždy, DIČ být vyplněno nemusí.

Následují `logo` a `signature`. Jejich jediným účelem je uchovávat obrázek s logem a podpisem klienta. Poslouží při generování faktury do PDF.

Jako další kontaktní atributy kromě `email`slouží také `web` a `phone`. Opět celkem očekávatelně je do nich uložen odkaz na webovou stránku uživatele respektive jeho telefonní číslo.

Atributy `street`,`citypsc` a `country` reprezentují poštovní adresu uživatele. Do atributu `street` je kromě jména ulice ukládáno číslo popisné, případně



Obrázek 4.1: Databázový model aplikace

i číslo orientační. `textbf{psc}` je textového typu z toho důvodu, že někteří uživatelé preferují zadávání poštovního směrovacího čísla, jako trojici oddělenou mezerou od zbývající dvojice, před zadáním jako pětice čísel. Formát zadaného PSČ se snažím uchovat, proto není možno zvolit číselnou podobu.

Mezi „registrační“ atributy entity `User` patří `registration_date`, dále pak `registration_code` a `active`. Při registraci je zaznamenán její čas do sloupce `registration_date`. Jedná se o unixové časové razítko (definováno jako počet vteřin od 1.1.1970), proto je zvolen číselný typ. Dále je uživateli během registrace vygenerován náhodný registrační kód, který je uložen do `registration_code` a zároveň odeslán uživateli na zadaný email. Dokud uživatel neověří svou emailovou adresu, je ve sloupci `active` uložena 0, po ověření 1.

Atribut `last_login` slouží pouze k uložení časového razítka posledního úspěšného přihlášení uživatele. Do atributu `failed_login_count` je ukládán počet nezdařených pokusů o přihlášení a v `last_failed_login` je uloženo časové razítko posledního neplatného pokusu o přihlášení. Obojí je použito při zamykání účtu.

Poslední trojice atributů – `invoice_text1-3` udržuje výchozí hodnoty pro texty vkládané na nově vytvořenou fakturu. `invoice_text1` reprezentuje text před položkami faktury, `invoice_text2` pak text za položkami faktury a nakonec `invoice_text3` text v zápatí exportované faktury.

4.1.3.2 Invoice

Další významnou entitou navrhované aplikace je faktura neboli **Invoice**, sloužící k uložení informace o konkrétní faktuře. Faktura je opět identifikovaná pomocí unikátního `id`. Dále je zde uložen atribut `owner_id`, což je cizí klíč do tabulky `users`. Ač je preferováno čistě číselné označení faktury, je uživateli umožněno do jejího čísla (atribut `number`), vkládat i písmena. Proto je opět zvolen textový datový typ. Všechny atributy začínající `customer_` mají shodná jména i funkčnost jako obdobné atributy v entitě **User**. Mohlo by se zdát lepší uložit údaje o zákazníkovi do samostatné tabulky a případně je použít ve více fakturách. U každé faktury je však potřeba uchovávat údaje platné v době jejího vystavení. Pokud se tedy například zákazník později přestěhuje a nové faktury budou vystavovány na novou adresu, staré faktury musí mít uloženou stále starou adresu zákazníka. Zákazníkova emailová adresa (uložená v atributu `customer_email`) je využívána při odesílání faktury emailem přímo z aplikace. U faktury je dále do sloupců `time_created` a `time_due` uloženo časové razítko vytvoření faktury resp. její splatnosti. Ve výchozím stavu je mezi těmito razítky přesně 14 dnů (1 209 600 sekund). Atributy začínající `payment_` v sobě nesou informace potřebné k zaplacení faktury. Prvním z nich je typ platby. Zde je zatím podporován bankovní převod a platba hotovostí. V aplikaci jsou definované konstanty `PAYMENT_TYPE_TRANSFER` a `PAYMENT_TYPE_CASH` reprezentující jednotlivé možnosti. `amount` udržuje infor-

maci o celkové ceně faktury, `account`, `ks` a `vs}` pak v případě, že je zvolena platba převodem, udržují číslo účtu, na které má být faktura zaplacena a dále konstantní a variabilní symbol platby. Posledním atributem týkajícím se platby je `currency`, neboli měna, ve které je faktura vystavena. Atributy `text_1-3` slouží k uložení již zmiňovaných textů na faktuře. Každá faktura může mít jiné texty, proto je potřeba je ke každé faktuře ukládat zvlášť. Poslední atribut – `public_url` v sobě nese informaci o unikátní URL, na které je faktura dostupná bez nutnosti přihlášení. Pro každou fakturu je tento atribut vygenerován při vytváření a nelze jej změnit.

4.1.3.3 Account

Tato entita reprezentuje bankovní účet uživatele. Jako všechny entity v aplikaci i bankovní účet je identifikován unikátním`id` a obsahuje cizí klíč z tabulky `users` – `owner_id`. Dále je u účtu evidováno jeho číslo (`number}`, měna účtu (`currency`) a název, který mu přiřadil uživatel pro snadnější orientaci – `name`.

4.1.3.4 Price

Další entitou je ceník, respektive jedna cena. Cena je opět identifikována pomocí `id` a obsahuje cizí klíč `owner_id`. U každé ceny je uložen název, který ji uživatel přiřadil (atribut `name`) a cena za množstevní jednotku – atribut `unit_price`.

4.1.3.5 Subject

Entita **Subjekt** reprezentuje zákazníka. Slouží k uložení informací o zákazníkovi tak, aby je nebylo nutné při vytváření faktury pokaždé znovu zadávat. Subjekt je identifikován pomocí `id` a obsahuje cizí klíč `owner_id`. Jména polí i jejich vlastnosti a použití jsou shodné jako u entity **User** 4.1.3.1.

4.1.3.6 Item

Entita **Item** reprezentuje jednu položku faktury. Identifikována `id`, s cizím klíčem `owner_id` a druhým cizím klíčem – `invoice_id`, tedy `id` faktury, ke které patří. Každá položka faktury má své jméno (`name`), jednotkovou cenu (`unit_price`), cenu za určité množství (`unit_amount`) a celkovou cenu (`total_price`), což je násobek jednotkové ceny a množství. Pro snazší manipulaci je tento násobek uložen jako samostatný sloupec.

4.1.3.7 Payment

Poslední entitou aplikace je platba. Opět identifikována pomocí unikátního `id`, obsahuje dva cizí klíče – do tabulky `invoices` jde o klíč `invoice_id` a

do tabulky `users` pak o klíč `owner_id`. V atributu `account` je uloženo číslo účtu, na které má platba přijít, v atributu `amount` pak částka, na kterou je daná faktura vystavena. V atributu `paid` je zaznamenána dosavadní uhrazená částka a v atributu `time_paid` pak časové razítko poslední příchozí platby. Poslední dva atributy obsahují další informace k očekávané platbě – její variabilní symbol (`vs`) a konstantní symbol (`ks`). Platba je vytvářena i pro faktury placené v hotovosti, slouží jako potvrzení uhrazení těchto faktur. Pro hotovostní platby jsou důležité pouze atributy `amount`, `paid` a `time_paid`.

4.1.4 Tabulky nereprezentující entitu

Většina tabulek v databázi reprezentuje konkrétní entitu používanou v aplikaci. Výjimku tvoří tabulka **settings**, do které jsou přesunuta některá nastavení uživatele.

4.1.4.1 Settings

V tabulce **settings** jsou momentálně uložena především nastavení pro odesílání faktur emailem klientovi. Je zde tedy položka pro výchozí předmět odesílaného emailu (`email_subject`), dále pak text emailu, pokud je faktura přiložena jako příloha (`email_body_attachment`) a poslední položkou je výchozí text emailu, kde je uveden odkaz na veřejnou URL faktury (`email_body_link`). Stejně jako u entit, obsahuje každý záznam v tabulce ještě unikátní `id` a cizí klíč do tabulky `users` – `owner_id`.

4.2 API

Jádro celé aplikace je dostupné přes aplikační rozhraní, které je navržené tak, aby co nejlépe splňovalo principy REST [23]. Tomu odpovídá navržení URL používaných v aplikaci, definování akcí pomocí HTTP metod, využívání HATEOAS [24] pro navigaci v entitách, správné užití návratových kódů nebo bezstavovost aplikace. Veškerá komunikace probíhá ve formátu JSON.

4.2.1 Funkce

V této sekci je popsáno několik základních funkcí aplikace.

4.2.1.1 Registrace uživatele

Základní funkcí všech aplikací, které nějak pracují s uživatelskými účty, je registrace nového uživatele. V navrhované aplikaci je jako jednoznačný identifikátor uživatelského účtu používána emailová adresa. Společně s heslem se tedy jedná o povinné položky při registraci. Dále může uživatel uvést další údaje jako je jméno, příjmení, IČO, DIČ, webovou adresu, telefon a položky adresy – ulici, číslo popisné a orientační, dále pak město a poštovní směrovací

číslo a nakonec stát. Všechny údaje vyjma emailové adresy jde později doplnit nebo změnit.

Aplikace při přijetí registračního požadavku zkontroluje přítomnost povinných polí a následně vytvoří nový uživatelský účet. Na emailovou adresu, kterou uživatel uvedl při registraci, je odeslán email s odkazem, který slouží k ověření emailové adresy a po jeho navštívení je uživatelský účet aktivován.

Dokud není uživatelský účet aktivován, není možné se k účtu přihlásit a využívat aplikaci.

4.2.1.2 Přihlášení a autentizace uživatele

V rámci aplikace se lze přihlásit pouze k uživatelskému účtu, u nějž byla ověřena emailová adresa a je tedy aktivován. Při pokusu o přihlášení k neaktivovaném účtu je uživateli zamítnut přístup a je informován o nutnosti účet nejprve aktivovat.

Po třech neúspěšných pokusech o přihlášení ke konkrétnímu uživatelskému účtu, je tento účet zablokován. Nejprve na dobu jedné hodiny. Následně jsou uživateli umožněny další tři pokusy o přihlášení. Pokud ani jeden z těchto dalších tří pokusů o přihlášení není úspěšný, je účet znovu zablokován, tentokrát již na dvě hodiny. Po odblokování účtu má uživatel vždy 3 pokusy o přihlášení. Doba zablokování se prodlužuje vždy na dvojnásobek předchozí doby zablokování. Po úspěšném přihlášení se počet neúspěšných vynuluje včetně případné doby blokování.

Při úspěšném přihlášení je uživateli sdělen jeho *api_token*, který využívá k autentizaci do všech ostatních sekcí aplikace. Email a heslo jsou tedy využívány pouze pro získání *api_token*. Ve zbytku aplikace se uživatel jeho pomocí autentizuje a email a heslo již nejsou dále využívány.

4.2.1.3 Načítání údajů z registru ARES

„Administrativní registr ekonomických subjektů je informační systém, který umožňuje vyhledávání nad ekonomickými subjekty registrovanými v České republice. Zprostředkovává zobrazení údajů vedených v jednotlivých registrech státní správy, ze kterých čerpá data (tzv. zdrojové registry).“ [25]. Vyhledávání v tomto registru je možné buď pomocí formulářů na webovém rozhraní nebo pomocí HTTP dotazu. V tomto případě je potřeba jako parametr dotazu uvést IČO požadovaného subjektu. Odpověď registru je pak ve formátu XML a pro existující IČO lze z odpovědi získat údaje o jméně subjektu, jeho adrese a také jeho DIČ.

Načítání údajů z registru ARES je v aplikaci využíváno na více místech a proto je vytvořena funkce `loadInfoFromARES($ico)`, jejímž výstupem je pole obsahující údaje o subjektu s daným identifikačním číslem osoby.

4.2.1.4 Ukládání údajů o subjektech

Při vytvoření nové faktury je zkontrolováno, zda je již zákazník s tímto jménem v adresáři uživatele. V tomto adresáři jsou všichni zákazníci (subjekty), kterým uživatel někdy vystavil fakturu. Dále se v tomto adresáři nacházejí údaje subjektů, které si sem uživatel sám přidal.

U každého subjektu je evidováno jeho jméno, adresa, IČO, DIČ, webová adresa, telefonní číslo a emailová adresa. Vložení subjektu uživatelem je možné buď přímo odesláním jeho údajů v rámci HTTP dotazu a nebo je možné nechat nový subjekt vytvořit na základě údajů získaných z registru ARES (viz 4.2.1.3).

4.2.1.5 Tvorba faktury

Vytváří faktur je jedna ze základních funkcí navrhované aplikace. Pro její vytvoření je nutné znát v první řadě údaje o klientovi. Ty je možné zadat 3 způsoby – načíst údaje o klientovi z registru ARES (viz 4.2.1.3), použít jeden ze subjektů v adresáři uživatele (viz 4.2.1.4) a nebo údaje o klientovi zadat přímo. U každého klienta je potřeba uvést jeho jméno a adresu, bez těchto údajů nelze fakturu vytvořit.

Další důležitou položkou faktury je její číslo. Čísla faktur by měla tvořit souvislou řadu. Číslo faktury tedy může poskytnout uživatel, pokud si z nějakého důvodu přeje použít číslo mimo řadu a nebo číslo nové faktury vygeneruje aplikace. Pokud číslo nové faktury generuje aplikace, vznikne jako číslo o jedničku větší než je nejvyšší číslo faktury vystavené v aktuálním roce, případně číslo jedna pro první fakturu v roce (pokud dosavadní nejvyšší číslo vystavené faktury bylo 20180005, nová faktura dostane přiděleno číslo 20180006).

Dále je potřeba uvést údaje potřebné pro zaplacení faktury, především pak způsob úhrady a měnu, ve které je faktura vystavena. Pokud tyto údaje nejsou poskytnuty, implicitně se předpokládá úhrada hotovostí a jako měna faktury pak česká koruna. V případě platby převodem je pak nutné zadat ještě číslo účtu, na který má být provedena platba. Jako variabilní symbol se v takovém případě použije číslo faktury.

Posledním údajem potřebným pro vytvoření nové faktury je trojice textů, které jsou na ni uvedeny. Jedná se o texty před jejími položkami, za seznamem položek a v zápatí faktury.

Přidávání jednotlivých položek na fakturu pak probíhá v rámci správy faktury (viz 4.2.1.6).

4.2.1.6 Správa faktury

V předchozí části je popsán mechanismus vytvoření faktury. Fakturu je dále možné upravit nebo smazat. Základní úpravou je přidání nebo odebrání položek faktury. Položky se přidávají a odebírají po jedné a po každém přidání

či odebrání je znovu přepočítána celková cena faktury a tato informace je aktualizována na všech místech, kde je uložena.

Jednotlivé položky je též možné upravit a to jak jejich název, tak cenu a množství. Po uložení změn u položky je též potřeba aktualizovat informace na dalších místech.

Dále jde na faktuře upravit v podstatě veškeré údaje – od údajů o klientovi, přes údaje o platbě a číslo faktury až po trojici textů, které se na faktuře nacházejí. Jedinou neměnnou informací pro fakturu je její `id`, dále pak `id` jejího vystavovatele uložené v `owner_id` a její kód pro tvorbu veřejné URL v `public_url`.

4.2.1.7 QR platba

Pro usnadnění platby faktury je vhodné na fakturu umístit QR kód obsahující informace o platbě ve standardizovaném formátu. V České republice se používá formát definovaný Českou bankovní asociací [26]. Pro vytvoření validního QR kódu popisujícího platbu je potřeba nejprve vytvořit validní SPAYD řetězec. Formát tohoto řetězce je popsán v již zmíněném standardu. Do QR platby je možné uložit veškeré údaje o platbě jako je číslo účtu, částka, měna, veškeré symboly, zpráva pro protistranu atd. Vygenerovaný řetězec je pak potřeba pouze nechat převést do podoby QR kódu. K tomu je v aplikaci využita opensource knihovna **PHP QR Code** [27].

4.2.1.8 Export faktur do PDF

Jedním z bodů zadání je schopnost aplikace exportovat faktury do formátu PDF. Existuje mnoho způsobů, jak vygenerovat PDF soubor. Asi nejlepším by bylo PDF generovat z šablony pomocí nějaké varianty programu **TeX**. Bohužel není zaručeno, že na webhostingu, na kterém aplikace poběží, bude nějaká varianta **TeX**u dostupná. Proto jsem se rozhodl zvolit řešení využívající pro generování PDF souborů PHP knihovnu. Těchto knihoven existuje několik. Původně jsem chtěl použít knihovnu **Dompdf** [28], avšak po krátkém testování jsem se kvůli lepší kompatibilitě s CSS, výrazně vyšší rychlosti a stále aktivnímu vývoji rozhodl nakonec použít knihovnu **mPDF** [29].

Knihovna na vstupu očekává HTML a jejím výstupem je PDF soubor. Dále je možné ovlivnit vlastnosti vygenerovaného PDF – změny orientace stran, nastavení záhlaví a zápatí stran, rozměry dokumentu atd.

Pro vytvoření HTML, které je následně použito jako vstup pro generování PDF, jsem definoval rozhraní `InvoiceTemplateInterface` (viz 4.2.2.2) obsahující mimo jiné metodu `renderHtml(Invoice $invoice)`, která na vstupu očekává instanci třídy **Invoice** a jejím výstupem je HTML string. V rámci aplikace je navržena také výchozí šablona, která implementuje zmíněné rozhraní.

Ač se mi povedlo na internetu najít několik šablon faktury, bohužel žádná nebyla kompatibilní s knihovnami na převod z HTML do PDF. S největší pravděpodobní je tato nekompatibilita způsobena nesplnitelnými nároky na šířku dokumentu a použitými CSS vlastnostmi. PDF soubor si můžeme představit jako dokument o šířce 675 pixelů, do kterého je za pomoci CSS potřeba vysázet veškerý obsah.

Celou šablonu jsem tedy napsal od základu sám. Chápu však, že každý uživatel může mít jiné preference a tak je použito rozhraní tak, aby v budoucnu byl umožněn vývoj dalších šablon faktur.

4.2.1.9 Párování plateb

Bodem zadání je též prozkoumat API bank pro automatické párování plateb. Bohužel jak je popsáno v 2.3, rozhraní ještě nemusejí mít finální podobu a pro přístup k nim je potřeba získat licenci od ČNB.

V aplikaci jsem byl tedy nucen uchýlit se k řešení založeném na čtení notifikačních emailů, které banka na žádost klienta posílá například při příchozí platbě na bankovní účet. Bohužel, jak je vidět v ukázce emailu 4.1, ne všechny banky zasílají email se všemi potřebnými údaji – chybí například číslo účtu na které byla platba přijata nebo variabilní symbol platby. Jelikož jsou platby v aplikaci identifikováno právě touto dvojicí údajů, bez jejich znalosti není možné platbu jednoznačně spárovat s fakturou.

Pokud však banka v emailu uvádí veškeré potřebné informace (jako například ČSOB v ukázce emailu 4.2), lze na základě takového emailu přiřadit platbu ke konkrétní faktuře. Podařilo se mi získat notifikační email z pěti českých bank, kde dvě posílají email s nedostatečnými údaji – Equa bank a mBank a zbývající tři posílají všechny potřebné informace – ČSOB, Komerční banka, Air Bank. Pro tyto tři banky podporuje aplikace párování plateb. Podle [30] by měly potřebné informace v notifikačním emailu posílat ještě Fio banka, Raiffeisen bank, Sberbank, Moneta money bank a UniCredit Bank. Bohužel se mi nepodařilo získat notifikační emaily z těchto bank. Z velkých českých bank nedostatečná data posílá také například Česká spořitelna.

Ukázka emailu 4.1 (Equa bank)

```
Vážená klientko / Vážený kliente,

na Váš účet 1025320267 byla připsána částka ve výši 4000.00 CZK.
Disponibilní zůstatek dne 19.06.2018 09:00 je 7505.25 CZK.

Pro více informací navštivte své internetové bankovnictví.

Vaše Equa bank
```

Ukázka emailu 4.2 (ČSOB)

```
Vážená klientko, vážený kliente,
```

```
dne 4.4.2018 byla na účtu 218119231 zaúčtovaná transakce TPS:
částka +451,00 CZK
z účtu 19022381/0100
detaily platby:
OBEC SMĚDČICE
KS 0132
VS 0000000007
SS 0000000000
VS protistrany 0000000007
SS protistrany 0000000000
zpráva pro příjemce:
ODMENA CLEN P.SUCHY 03/2018

Zůstatek na účtu po zaúčtování transakce: +451,00 CZK

Děkujeme za využití služeb ČSOB Info 24,

Vaše ČSOB
```

Pokud tedy uživatel využívá služeb jedné z podporovaných bank a nastaví si odesílání notifikačních emailů na adresu spravovanou aplikací, platby budou spárovány. Skript **paymentChecker.php** nejprve přečte obsah schránky a pak v cyklu pro každou zprávu ve schránce, podle adresy, ze které je zpráva odeslána odvodí o kterou banku se jedná a podle toho se pokusí z emailu vyextrahovat potřebná data a zaznamenat změny do databáze. Emaily jsou následně přesunuty do jiného adresáře tak, aby bylo jasně odděleno, které jsou již zpracovány a které na zpracování teprve čekají.

4.2.1.10 Statistiky

Poslední funkcí, která stojí za zmínku je generování statistických přehledů. Aplikace nabízí měsíční přehled příjmů uživatele vygenerovaný z vystavených faktur, dále například tři největší zákazníky, co se příjmů týče.

4.2.2 Třídy

V aplikaci je používáno několik tříd, které reprezentují jednotlivé entity používané v aplikaci nebo jinak usnadňují práci s nimi. Třídy využívají vlastní jmenný prostor `\Openfaktura\API\` .

4.2.2.1 Výjimky

V aplikaci je několik funkcí, ve kterých mohou nastat neočekávané chyby. Tyto funkce v takovém případě vyhodí jednu z výjimek. Výjimky jsou specifické

pouze svým jménem, které usnadňuje na základě zachycení konkrétního typu výjimky určit, co zapříčinilo chybu.

V aplikaci jsou tedy definovány následující výjimky. Jejich jména poměrně dobře vypovídají o situaci, ve které je daná výjimka vyhozena.

- ARESUnreachableException
- DatabaseConnectionErrorException
- InvalidICONumberException
- InvalidIdException
- InvoiceNumberNotNumericException
- UserAlreadyExistsException

4.2.2.2 Šablony pro export faktur

Jak již bylo zmíněno v 4.2.1.8, v aplikaci je definováno rozhraní **InvoiceTemplateInterface**, které obsahuje 3 metody:

```
interface InvoiceTemplateInterface
{
  public static function getInfo();
  public static function getName();
  public static function renderHtml(\Openfaktura\API\Invoice
  ↪  $invoice);
}
```

První dvě metody slouží pouze pro získání informací o použité šabloně. Tyto funkce budou využity ve chvíli, kdy bude v aplikaci dostupných několik šablon pro export a uživatel si bude moci sám zvolit, kterou chce použít. Funkčnost třetí metody už byla také dříve popsána v sekci 4.2.1.8. Třída **DefaultInvoiceTemplate** je pak referenční implementací tohoto rozhraní.

4.2.2.3 BankAccount

Tato třída reprezentuje bankovní účet, nabízí standardně gettery a settery na většinu jejích proměnných a dále poskytuje 4 metody:

```
function countIBAN($firstPart, $secondPart, $bankCode,
↪  $withSpaces)
function countIBANFromString($accountString, $withSpaces)
function generateQRPayment($amount, $date, $VS, $KS, $SS,
↪  $message)
function generateQRPaymentURL($amount, $date, $VS, $KS, $SS,
↪  $message)
```

První dvě metody slouží k výpočtu kódu IBAN, který je následně využíván v QR platbě nebo na exportované faktuře, třetí metoda vytvoří obrázek QR kódu obsahujícího zadané údaje a poslední metoda vrací odkaz na kterém lze získat obrázek tohoto QR kódu. Odkaz na obrázek je používán při exportu faktury do PDF nebo HTML.

4.2.2.4 Configuration

Tato třída byla zmíněna již v sekci 4.1.1. Slouží k přístupu ke konfiguračním proměnným v rámci aplikace. Implementuje rozhraní `ArrayAccess` [19] , díky čemuž je pak možno k jednotlivým proměnným přistupovat jako k pojmenovaným prvkům vícerozměrného pole.

4.2.2.5 CurrentUser

Jde o třídu, která v sobě nese informace o právě přihlášeném uživateli. Poskytuje v podstatě jen sady getterů a setterů. Čtyři z nich jsou pro fungování aplikaci zásadní a proto stojí za zmínku:

```
function isAuthenticated()
function isActive()
function getId()
function getApiToken()
```

Návratová hodnota první metody slouží jako ukazatel, zda je uživatel přihlášen či nikoliv. Pokud přihlášen není, přichází na řadu druhá metoda, která určuje, zda má uživatel vůbec řádně aktivovaný účet. Podle toho jsou vráceny různé odpovědi na pokusy o přihlášení. Následující dvě metody by neměly být volány bez předchozího ověření, že je uživatel přihlášen. `getId()` vrací unikátní `id` právě přihlášeného uživatele a metoda `getApiToken()` pak celkem předvídatelně vrací jeho token pro přístup k API.

4.2.2.6 Invoice

Invoice je asi nejrozsáhlejší třída celé aplikace. Reprezentuje konkretní fakturu a poskytuje metody pro přístup k jejím vlastnostem a několik dalších metod, které mají usnadnit práci s fakturou.

```
public function loadFromDatabase($id, PDO $database)
public function updateWithArray($updateArray)
public function updateToDatabase(PDO $database)
public function insertToDatabase(PDO $database)
```

Funkce `loadFromDatabase()` slouží k načtení informací o jedné konkrétní faktuře z databáze. Jejími vstupními parametry je **id** faktury, kterou chceme načíst a druhým parametrem je PDO spojení s databází.

Funkce `updateWithArray()` na vstupu očekává jediný parametr a to asociativní pole, podle něhož se náležitě upraví informace, které v sobě tato instance třídy právě nese.

Pokud chceme změnu dat provedenou pomocí předchozí metody napevno zanést i do databáze, je nám k dispozici metoda `updateToDatabase()`, která aktualizuje záznam faktury v databázi tak, aby hodnoty souhlasily s těmi, které má právě uloženy v sobě.

Poslední metodou je `insertToDatabase()` , která má sloužit především pro usnadnění zapsání dat o nově vytvořené faktuře do databáze.

4.2.2.7 Item

Třída `Item` reprezentuje jednu položku faktury. Jedná se tedy jen o jakousi obálku usnadňující přístup k jejím proměnným. Umožňuje získat a nastavit jméno položky, její jednotkovou cenu a množství.

4.2.2.8 Subject

`Subject` je třída, která slouží pro reprezentaci veškerých subjektů, ať už jde o subjekty z adresáře uživatele či přímo o samotné uživatele. Používá se v rámci třídy `Invoice` pro identifikaci dodavatele a klienta na faktuře. Podobně jako zmíněná třída `Invoice` poskytuje kromě standardních getterů a setterů i několik metod pro snadnější manipulaci:

```
public function loadUser($id, PDO $db)
public function loadFromARES($ico)
public function loadSubject(PDO $database,$subjectId, $ownerId)
public function getLogoURL()
public function getSignatureURL()
```

První metoda – `loadUser()` slouží k načtení údajů o uživateli. Očekává `id` uživatele, kterého chceme načíst a `PDO` spojení s databází.

Další metodou je `loadFromARES()`, která má podobnou funkčnost, pouze tentokrát nezískává údaje o subjektu z lokální databáze, ale z registru ARES (viz sekce 4.2.1.3). Na vstupu očekává IČO subjektu, který chceme načíst.

Třetí metodou na načítání subjektu je `loadSubject`. Jak již název napovídá, slouží k načtení subjektu z adresáře uživatele. Aby toho mohla dosáhnout, potřebuje však znát `id` subjektu a jeho vlastníka a pak také `PDO` spojení s databází.

Zbývající dvě metody pouze generují URL vedoucí na logo respektive podpis uživatele. Zohledňují při tom konfigurační proměnné. Vygenerované URL se používají při exportu faktur do HTML a PDF.

4.2.2.9 JsonResponse

Tato třída nereprezentuje žádnou entitu aplikace, pouze shromažďuje funkce pro snadnější odesílání odpovědi uživateli. Kromě getterů a setterů na tělo odpovědi a návratový kód stojí za zmínku tyto tři metody:

```
public function setHeader($name, $value)
public function setupStandardResponseByCode($responseCode)
public function send()
```

Funkce `setHeader()` slouží k nastavení konkrétní hlavičky v odpovědi. Jak jména vstupní proměnných napovídají v proměnné `name` očekává metoda jméno hlavičky a následně v proměnné `value` pak její požadovanou hodnotu. Pokud je dvakrát nastavena hlavička se stejným jménem, hodnota se přepíše.

Pro usnadnění odesílání často používaných odpovědí je přítomna metoda `setupStandardResponseByCode()`, která pro časté návratové kódy sama nastaví tělo odpovědi. Očekává nepovinný parametr `responseCode`. Pokud je zadán, nastaví se celá třída podle něj, pokud zadán není, nastaví se celá třída podle návratového kódu, který má právě uložena.

Nejdůležitější metodou této třídy je `send()`. Ta nastaví návratový kód odpovědi, správně zakóduje tělo do formátu JSON, nastaví všechny hlavičky a odešle odpověď uživateli.

4.2.3 Zdroje

V RESTu jsou data reprezentována pomocí zdrojů (anglicky resources). Podle doporučení [31] by každý zdroj měl být dostupný pod unikátní URI. Doporučuje se pro jména zdrojů používat podstatná jména, dále používat v URI pouze malá písmena a pomlčky. Kolekce zdrojů by měla být pojmenována množným číslem. Dalším doporučením je naznačit hierarchii zdrojů pomocí lomítka a naopak na konci adresy lomítko nepoužívat. Aplikace je navržena tak, aby uměla pracovat jak s adresami zakončenými lomítkem i bez lomítka.

K iniciaci akce se zdrojem jsou používány HTTP metody. Výjimku tvoří takzvané controllery, které umožňují i v adrese použít sloveso. Výsledná URI zdroje pak může vypadat například takto:

```
https://api.openfaktura.cz/users/{user-id}/invoices/{invoice-id}/
↪  items/{item-id}
```

Po doplnění **id** jednotlivých zdrojů pak například:

```
https://api.openfaktura.cz/users/20/invoices/12/items/3
```

V navrhované aplikace nalezneme 3 základní zdroje z čehož dva jsou kolekce entit a třetí slouží pro autentizaci uživatele.

1. **Auth** dostupný na `https://api.openfaktura.cz/auth` slouží k ověření emailu uživatele a aktivování nově vytvořeného účtu, dále pak umožňuje přihlášení uživatele pomocí kombinace emailové adresy a hesla za účelem získání *api_token* a poslední funkcí, kterou nabízí je možnost vytvořit nový *api_token*.

2. **Invoices** dostupný z `https://api.openfaktura.cz/invoices` nabízí přehled veškerých faktur uživatele, dále pak detail jednotlivých faktur s možností exportu do formátu PDF nebo HTML, přehled položek faktury, úpravy údajů faktury, mazání faktur atd.

3. **Users** dostupný na `https://api.openfaktura.cz/users` je nejobsáhlejším zdrojem celé aplikace. Základní funkčností je registrace a správa uživatelů, následuje získání a správa kolekcí, které má uživatel uloženy – sem patří například bankovní účty, subjekty nebo položky uživatelova ceníku. Dále obsahuje subkolekci `invoices` sloužící jako alias pro předchozí zdroj. Posledním zdrojem, který **Users** nabízí jsou statistiky.

4.3 Frontend

Rád bych zde na úvod uvedl, že své grafické cítění považuji za podprůměrné a proto i většina navrženého řešení není příliš vizuálně atraktivní. Budu rád pokud grafickou stránku aplikace v budoucnu někdo upraví. Funkčnost by však splňovat měla.

Frontend je navržen jako zcela nezávislá vrstva umožňující ovládání aplikace pomocí navrženého aplikačního rozhraní. S jádrem aplikace nic nesdílí a tak mohou být nasazeny nezávisle na sobě na různých doménách. Webové rozhraní jsem se snažil navrhnout co nejvíce nezávislé na serveru, na kterém běží. Ten tedy pouze na začátku pošle kostru stránky klientovi doplněnou o základní konfiguraci – především URL na které se nachází API a od této chvíle je již vše v režii JavaScriptu na straně klienta.

4.3.1 Použité knihovny

Jak již bylo zmíněno, grafický návrh a hlavně pak kaskádové styly nejsou moje nejsilnější stránka a tak jsem se rozhodl celé GUI vytvořit za pomoci externích knihoven, které by měly pomoci zjednodušit implementaci grafického rozhraní a také usnadnit práci s API.

4.3.1.1 Bootstrap

Bootstrap je v současnosti velice populární knihovna pro tvorbu responsivních webových rozhraní. Během studia jsem jej použil na několika různých školních projektech a tak jsem se rozhodl jej použít i v rámci svojí diplomové práce.

Z Bootstrapu je v navrhovaném webovém rozhraní použito mnoho prvků od základu, kterým je úprava typografie veškerých textů přes tlačítka a formuláře až po vyskakovací okna a mizející upozornění.

4.3.1.2 jQuery

Bootstrap je s jQuery pevně spjatý a jelikož jsem neměl žádnou předchozí zkušenost s jiným JavaScriptovým frameworkem, rozhodl jsem se používat ve webovém rozhraní právě jQuery. Věřím však, že ty základní operace, které jsou ve webovém rozhraní použity by šly bez větších potíží dosáhnout jen se základním JavaScriptem. Použity jsou především funkce na nahrazení celého obsahu elementu, na získání hodnot vstupních polí a funkce na komunikaci s API.

4.3.2 Serverová část

Jak již bylo zmíněno, část úlohy, kterou je nutné provést na serveru je minimální možná. Server pouze předá klientovi stránku obsahující odkazy na všechny potřebné JavaScriptové a CSS soubory. Jedinou další důležitou informací, kterou server musí předat klientovi je adresa na které se nachází API se kterým má webové rozhraní komunikovat.

4.3.3 Klientská část

V podstatě veškerá režie probíhá na straně klienta. Všechny aktivní prvky webového rozhraní jsou vstupním bodem pro některou z funkcí, která nějak modifikuje obsah nebo na pozadí provede nějakou operaci.

4.3.3.1 Autentizace

Pro komunikaci s API je potřeba získat uživatelův *api_token*. Aby nebylo nutné se znovu přihlašovat při každé návštěvě webového rozhraní, je *api_token* po získání uložen do cookies s názvem `openfaktura_token`. První operací, kterou tedy webové rozhraní po načtení provede je kontrola, zda je v cookies přítomen *api_token*. Pokud ano, načte jeho pomocí z API informace o uživateli a zobrazí prvky, které byly doposud skryty.

4.3.3.2 Komunikace s API

Ke komunikaci s API je využívána především interní funkce knihovny jQuery `ajax(url [, settings ])`, která nabízí možnost poslat požadavek pomocí různých HTTP metod, přikládat k požadavku data atd. ale především poskytuje callback funkce při dokončení. Jelikož komunikace v JavaScriptu probíhá asynchronně, je nutné určité části kódu provádět až po dokončení předchozí

operace. Stačí tedy registrovat funkci, která se má zavolat v případě konkrétního stavu funkce `ajax()` a v ní požadovaný kód provést.

```
$.ajax({
        type: "get", // použitá http metoda
        url: apiUrl + "/users?token=" + api_token, //url
        success: function (data) {
                // kód který se provede při úspěchu
        },
        error: function (xhr, status) {
                // kód který se provede při neúspěchu požadavku
        }
});
```

4.3.3.3 Přepínání stránek

V kostře, kterou server posílá klientovi, jsou 3 významné elementy `<div>`. Jeden v sobě obsahuje horní lištu s menu, druhý pak drží místo pro případné vyskakovací upozornění a poslední, s `id=mainBody` slouží právě k vkládání obsahu stránek. Funkce, které mají za úkol změnit zobrazený obsah pak typicky využívají právě tento `<div>` ke změně obsahu pomocí jQuery #id selectoru a funkce `html()`.

```
$("#mainBody").html("/* novy obsah elementu*/");
```

4.3.3.4 Interaktivní prvky

Webové rozhraní kromě odkazů a tlačítek, která změní zobrazení obsahuje i další prvky, které sice nemají vliv na zobrazení stránky, ale přesto jsou pro funkčnost velmi důležité.

Příkladem takového prvku může být tlačítko „ARES“ umístěné ve formulářích, ve kterých se zadávají údaje o subjektech. Umožňuje vyplnit formulářová pole informacemi z registru ARES. Kliknutím se spustí funkce, která stáhne data z registru a vyplní hodnotami příslušná pole ve formuláři.

Dalším podobným interaktivním prvkem jsou seznamy na výběr. Nabízejí například výběr klienta z adresáře, výběr bankovního účtu, který se má použít nebo přidání položky z ceníku. Kliknutím na konkrétní prvek seznamu je spuštěna funkce, která z API získá detailní informace a zvoleném prvku a doplní je opět do patřičných polí formuláře.

Předposledním použitým interaktivním prvkem jsou vyskakovací okna, neboli v terminologii Bootstrapu `Modal Boxes`. Kód takového prvku je možné nadefinovat klidně v těle stránky odesílané ze serveru. Nezobrazí se, dokud

není aktivován například stiskem tlačítka. Stiskem dalšího tlačítka může být opět skryt. `Modal Box` je využíván například u přihlášení nebo odesílání faktury emailem.

Posledním interaktivní prvkem, který je v aplikaci použit a stojí za zmínku je `alert`. V kostře stránky je připraven speciální `<div>` právě pro zobrazování těchto upozornění. Využíván je především pro informování o výsledku prováděných operací.

Kapitola 5

Implementace

5.1 Nastavení

5.1.1 .htaccess

Během návrhu a vývoje vyšlo najevo několik nutných úprav konfiguračního souboru **.htaccess**, který ovlivňuje chování webového serveru, na kterém je aplikace umístěna. Aplikace je připravena pracovat i bez zásahu do tohoto konfiguračního souboru, avšak je potřeba podle toho náležitě upravit konfigurační soubory aplikace. Dále je například doporučeno nepoužívat k ukládání konfigurace soubory ve formátu **ini** (viz. 4.1.1).

5.1.1.1 Přesměrování na https

Celá aplikace je kriticky závislá na tom, aby veškerá komunikace probíhala šifrovaně. Aby tato podmínka byla zajištěna ve všech případech, je potřeba do souboru **.htaccess** přidat následující řádek, který zajistí, že klient přistupující nešifrovaně, přes protokol **http** je pomocí http kódu 301 přesměrován na adresu začínající protokolem **https**, tedy přístup pomocí šifrovaného spojení.

```
# aktivace rewrite enginu
RewriteEngine On

# presmerovani na https
RewriteCond %{HTTPS} !=on
RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

5.1.1.2 Podstrčení obsahu v API

V rámci aplikačního rozhraní veškerou komunikaci s klientem vyřizuje skript **index.php**. V URL tedy jako první za jménem domény následuje právě jméno tohoto skriptu a až pak v adrese nalezneme cestu k požadovanému zdroji. Například URL zdroje **users** má pak následující podobu:

`https://api.openfaktura.cz/index.php/users`

Systém je takto plně funkční, avšak elegantnější je, podobně jako u většiny existujících API, navázat adresou zdroje rovnou na jméno domény a dostat tedy adresu ve formátu:

`https://api.openfaktura.cz/users`

K dosažení tohoto chování stačí do **.htaccess** přidat další pravidlo, které při dotazu na URL, neobsahující **index.php**, klientovi podstrčí obsah z adresy, která jméno skriptu obsahuje. Pokud dotaz směřuje na existující soubor, pravidlo se neuplatní. Pokud však na zadané URL žádný soubor ani adresář dostupný není, dojde k podstrčení obsahu. Toho lze využít pro úpravu chybové hlášky **404 Not found**, nezávisle na nastavení serveru, na kterém aplikace běží.

```
# aktivace rewrite enginu, pokud jeste neni aktivovan
RewriteEngine On

#presmerovani veskereho provozu na index.php
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ /index.php/$1 [L,QSA]
```

Aplikace zvládá obě možné varianty, avšak v konfiguračním souboru je potřeba určit, zda je používána varianta s **index.php** či bez. Aplikace podle toho tvoří odkazy – aby se zamezilo odkazování na url obsahující `index.php`, když je používána varianta bez něj a naopak.

5.1.1.3 Zakázání přístupu k INI souborům

Dalším pravidlem, které doporučuji do **.htaccess** souboru uvést je zakázaní přístupu k souborům s koncovkou **.ini**. Nepříjemně mě překvapilo, že na některých webhostinzích jsou tyto soubory, ve výchozím nastavení, bez problémů čitelné pro běžného návštěvníka webu. Uchovávat v nich konfiguraci aplikace by tak bylo velké bezpečnostní riziko. Je tedy potřeba uvést následující pravidlo:

```
# zakaz pristupu k~souborum s~koncovkou ini
<Files *.ini>
Deny from all
</Files>
```

5.2 API

Jak již bylo zmíněno dříve v textu, jádro aplikace a aplikační rozhraní je implementováno v jazyce PHP. Aplikační rozhraní je dostupné online na adrese `https://api.openfaktura.cz`.

5.2.1 Rozdělení zdrojového kódu do souborů

Zdrojový kód aplikace by se dal rozdělit do tří kategorií – kód externích knihoven, kód tříd a zdrojový kód mimo třídy. Podle toho jsou jednotlivé části rozděleny do souborů a adresářů.

Externí knihovny jsou uloženy v adresáři `vendor`. Jedná se o knihovny **PHP QR Code**[27] a **mPDF**[29]. Název adresáře je záměrný, jelikož knihovna mPDF se dá nejsnadněji získat pomocí `composer`u. Pokud by tedy v budoucnu aplikace začala využívat composer například při instalaci, neměl by být nutný žádný zásah do kódu. Vývoj knihovny **PHP QR Code** skončil dříve než začal vývoj composeru a tak se jeho pomocí získat nedá. Byla však též přesunuta do adresáře `vendor` tak, aby externí knihovny nebyly umístěny ve více různých adresářích.

Kód všech vlastních tříd aplikace (viz 4.2.2) se nachází v adresáři `src` a jeho podadresářích, respektujících další větvení jmenného prostoru. Kód každé třídy je v samostatném souboru pojmenovaném shodně s názvem třídy.

Zbývající zdrojový kód aplikace je rozdělen do souborů v kořenovém adresáři API domény. Snažil jsem se kód seskupovat tak, aby části které spolu nějak logicky souvisí byly pohromadě. Najdeme zde tedy soubor **index.php**, který se stará o směrování veškerých dotazů do příslušných funkcí. Následujícím významným souborem je **core.php**, ve kterém je umístěno jakési „jádro“ aplikace. Tento soubor je připojen na začátku všech ostatních souborů a zajišťuje načtení konfiguračních souborů, vytvoření spojení s databází, pokouší se autentizovat uživatele a načítá sadu funkcí ze souboru **tools.php**. Tento soubor obsahuje především funkce využívané na několika místech v rámci aplikace. Následuje trojice souborů **auth.php**, **invoices.php** a **users.php**, které obsahují veškeré funkce na obsloužení dotazů vedoucích na stejnojmenné zdroje. Nakonec ještě dvojice skriptů, která s API přímo nesouvisí a to **paymentChecker.php**, který zajišťuje čtení zpráv z emailové schránky, jejich analýzu a párování s platbami v databázi a nakonec skript **qrPaymentGenerator.php**, který, jak již název napovídá, vytváří ze zadaných parametrů obrázky QR kódů pro použití na vyexportované faktuře.

5.2.2 Stavové kódy

Aplikace při každé odpovědi nastavuje stavový kód. Využita je jen podmnožina stavových kódů. Snažil jsem se držet doporučení kdy jaký kód vracet podle [32]. Aplikace tedy vrací tyto stavové kódy v těchto situacích:

200 **Ok** V případě validního požadavku, který proběhl bez problémů a nijak neovlivnil uložená data. Typickým příkladem jsou dotazy na zdroje metodou GET.

201 **Created** Odeslán v případě, že došlo k úspěšnému vytvoření nového zdroje. Odesílán například při vytvoření nového uživatele, nové faktury atd. Společně s tímto kódem je odesílána hlavička `Location` s URI nově vzniklého zdroje.

204 **No Content** Signalizuje úspěšné provedení požadované akce bez dalšího odeslaného obsahu – tělo odpovědi je prázdné. Typicky použito při úspěšném smazání nebo aktualizaci zdroje.

302 **Found** Využíván při přesměrování uživatele na jinou adresu.

400 **Bad Request** Tento kód je odeslán pokaždé, když je požadavek uživatele nějakým způsobem nesprávný – například pokud chybí povinná pole nebo jsou poskytnuty data ve špatném formátu.

401 **Unauthorized** Používán v situacích, kdy se uživatel, jehož se nepodařilo autentizovat snaží přistoupit kde zdroji, kde je vyžadováno přihlášení. Díky častému využívání této odpovědi je pro usnadnění v aplikaci definována funkce `_unauthorized()`, která zařídí odeslání chybové hlášky.

403 **Forbidden** Pokud se povedlo uživatele autentizovat, ale on se pokouší přistoupit ke zdroji, který patří jinému uživateli, pak je odeslán právě tento stavový kód doplněný textovým popisem chyby. Jako v předchozím případě je i zde implementována funkce `_accessForbidden()`. V některých případech je i v situaci, kdy uživatel přistupuje na adresu existujícího zdroje, odeslán kód 404, protože i samotná informace o existenci zdroje může být nevyžádaná.

404 **Not Found** Používán v případě dotazu na neexistující adresu souboru nebo zdroje. Používána však také u zdrojů, které sice existují, ale jejich existenci nechceme přiznat jiným uživatelům, pouze vlastníkovi. Pro usnadnění opět funkce `_resourceNotFound()`.

405 **Method Not Allowed** Tento kód je vrácen v případě použití HTTP metody, kterou daný zdroj nepodporuje. Například u dotazu na kolekci faktur je možné použít metody GET a POST, ale není možné použít metody PUT a DELETE. Použití jedné z těchto metod způsobí odeslání tohoto stavového kódu doplněného o hlavičku `Allow`, kde je uvedeno, které HTTP metody daný zdroj podporuje (např. `Allow: GET,POST`).

500 **Internal Server Error** Tento stavový kód je odeslán pokud došlo k jakékoliv chybě, která ale není způsobena špatným uživatelským vstupem. Příkladem může být třeba chyba komunikace s databází nebo nedostupnost registru ARES. Chyba je doplněna zprávou, kde je textový popis chyby a případně její detaily pro potřeby lazení.

5.2.3 Používané HTTP metody

Aplikace využívá k manipulaci se zdroji 4 základní HTTP metody – `GET`, `POST`, `PUT` a `DELETE`, s tím, že:

GET Slouží výhradně k čtení dat a nedochází k žádné změně zdrojů. Jedinou výjimkou je přihlašování uživatele, které též podporuje metodu GET a může změnit některá data uložená v profilu uživatele (čas jeho posledního přihlášení a počet neúspěšných pokusů o přihlášení).

POST Slouží k vytváření, především nových zdrojů, ale také k vytvoření nového *api_token* nebo k vytvoření a odeslání nového emailu s fakturou klientovi.

PUT Je používána výhradně pro úpravu již existujících zdrojů.

DELETE Slouží pouze k mazání zdrojů.

5.2.4 Funkce

V této části nastíním podstatné části implementace jednotlivých funkcí systému. Pro detailní pochopení všech postupů a funkcí pak doporučuji nahlédnout přímo do zdrojového kódu aplikace, který je v příloze. Ve všech funkcích zpracovávajících uživatelský vstup se tento vstup očekává ve formátu JSON, pokud není výslovně uvedeno jinak. Pokud na vstupu není validní JSON, je okamžitě odeslána chyba s kódem 400 a požadavek se dále nezpracovává.

5.2.4.1 Směrování poažadavku

Každý požadavek, který na API dorazí projde skrz skript **index.php**. V tomto skriptu je požadovaná URI zbavena parametrů a rozsekána podle lomítek:
`$path = explode("/", explode("?",$_SERVER['REQUEST_URI'])[0]);`
V závislosti na hodnotě prvku `$path[0]` je připojen patřičný soubor a následně zavolána funkce `_path_auth_router($path)`, `_path_users_router($path)` nebo funkce `_path_invoices_router($path)`. Pokud tento prvek nená ani jednu ze zmíněných hodnot, je uživateli odeslána chybová hláška s kódem 404.

Zmíněné funkce mají pouze jednu funkci a to ověřit, jestli jeden z registrovaných regulárních výrazů vyhovuje požadované URI a zavolat přiřazenou funkci, která již požadavek vyřeší nebo odeslat uživateli chybovou hlášku 404, pokud žádný z uvedených regulárních výrazů neodpovídá požadované URI.

```
function _path_invoices_router($path)
{
// ...
  if (preg_match('/.*[\/]{1}invoices[\/]{1}[0-9]+
  ↪  ([\/]{1}|)([?]{1}[^\/]*$|$)/', $_SERVER['REQUEST_URI'])) {
    _path_invoicesDetail($path);
```

```
    return;
  }
// ...
  _resourceNotFound();
  return;
}
```

5.2.4.2 Registrace uživatele

`POST https://api.openfaktura.cz/users?autoload={true/false}`

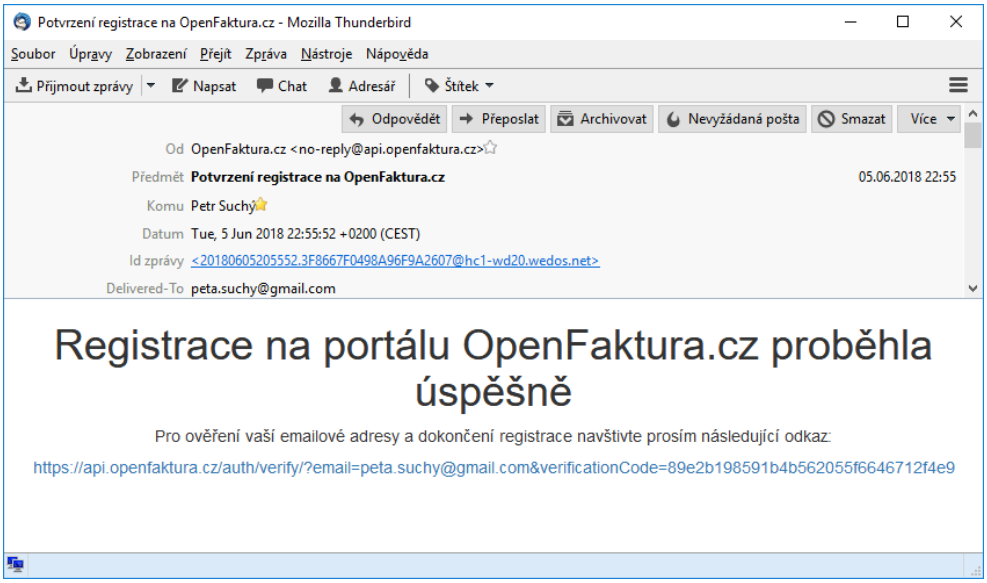
Jak již bylo zmíněno v návrhu, pro registraci nového uživatele je potřeba poskytnout emailovou adresu a heslo. Pokud nejsou tyto dva údaje poskytnuty, je požadavek okamžitě odmítnut s chybou 400. Pokud je parametr `autoload` nastaven na `true`, je v uživatelském vstupu očekáváno IČO, jehož vlastnosti se mají získat z registru ARES a použít jako informace o novém uživateli. Pokud tento parametr není nastaven na `true`, je brán v potaz pouze uživatelský vstup. Pokud uživatel využije načtení dat z registru ARES a přesto duplicitně zadá nějakou informaci (například PSČ) je použita hodnota z uživatelského vstupu. V případě registrace společnosti je doporučeno vyplnit pouze položku `name` a položku `surname` ponechat prázdnou. Při načítání z registru ARES je též vyplněno pouze `name`.

Následně dojde k zavolání funkce `createUser()`, která k zadaným informacím doplní informace o času registrace, vygeneruje náhodný unikátní *api_token*, pomocí vestavěné PHP funkce `password_hash()` provede hash hesla, nastaví výchozí hodnoty textů na faktuře a v emailu a všechny tyto informace vloží do databáze. Zároveň je na zadanou emailovou adresu odeslán email s odkazem na aktivaci účtu.

Pokud funkce `createUser()` zjistí, že uživatel se zadaným emailem již je v databázi uložen, vyhodí výjimku **UserAlreadyExistsException** a do databáze nejsou vložena žádná data.

Příklad validního uživatelského vstupu:

```
{
  "email":"petr@example.org",
  "password":"PetrJeNejlepsi",
  "name":"Petr",
  "surname":"Ukázkový",
  "street":"Dlouhá 32/344",
  "psc":"332 02",
  "city":"Horní Lhota",
  "country":"Česká republika",
  "ico":"02138221"
}
```



Obrázek 5.1: Screenshot přijatého ověřovacího emailu

5.2.4.3 Ověřeni emailové adresy

V předchozí sekci bylo zmíněno, že při registraci nového uživatele je mu na zadanou emailovou adresu odeslán email s odkaz na adresu, jejíž návštěvou emailovou adresu ověří. Formát a některé hlavičky je možno vidět na obrázku 5.1. Po navštívení tohoto odkazu je uživatelský účet aktivován tím, že je u záznamu v databázi nastavena hodnota sloupce `active` na `1` a od této chvíle je možné se k účtu přihlásit. Dokud není účet aktivován, budou všechny pokusy o přihlášení odmítnuty s chybou.

5.2.4.4 Přihlašování a autentizace uživatele

`GET https://api.openfaktura.cz/auth/login?password={}&email={}`
nebo
`POST https://api.openfaktura.cz/auth/login`
Přihlašovací údaje je možné předat buď jako parametry v adrese nebo v těle zprávy. Název parametrů je v obou případech shodný. Po úspěšném přihlášení je uživateli v rámci odpovědi odeslán *api_token*, který pak využívá ve zbytku aplikace k prokázání identity. Při neúspěšném pokusu je aktualizován počet neúspěšných pokusů v databázi. Po třech neúspěšných pokusech o přihlášení je účet na jednu hodinu uzamčen. Po hodině jsou umožněny další 3 pokusy o přihlášení. Pokud opět ani jeden ze 3 pokusů není úspěšný, účet je znovu uzamčen, tentokrát na dvě hodiny a tak dále. Doba uzamčení se vždy prodlužuje na dvojnásobek. Doba uzamčení se počítá od posledního neúspěšného

pokusu. Pokud je účet zamčen, aplikace vrací stejnou odpověď na všechny zadané údaje. Po úspěšném přihlášení se počet neúspěšných pokusů i doba uzamčení vynulují.

Ve zbytku aplikace prokazuje uživatel svoji totožnost pomocí *api_token*. Očekáván je jako parametr v URI `?token=a32bcad8a...`. Na následující adrese může uživatel získat nový *api_token*:
`POST https://api.openfaktura.cz/auth/token`

5.2.4.5 Načítání údajů z registru ARES

K načítání z registru ARES slouží funkce `loadInfoFromARES($ico)`. Tato funkce je použita na více místech v aplikaci a proto je její kód uložen v rámci souboru **tools.php**. Pro získání dat z registru ARES stačí odeslat požadavek metodou GET na adresu:
`http://wwwinfo.mfcr.cz/cgi-bin/ares/darv_bas.cgi?ico={ico}`
Kde `{ico}` je IČO požadovaného subjektu včetně případných počátečních nul. Odpověď je ve formátu XML. Je tedy potřeba odpověď zpracovat pomocí interní PHP knihovny **SimpleXML** [33]. Během hledání postupu jsem narazil na hotovou funkci určenou k načítání dat z registru ARES [34], kterou jsem pouze drobně upravil pro svoje potřeby. Načtené hodnoty jsou z funkce vráceny v podobě asociativního polo obsahujícího prvky `name`, `street`, `psc`, `city`, `country`, `ico` a `dic`. V případě chyby vyhazuje funkce výjimky **InvalidICONumberException** a **ARESUnreachableException**.

5.2.4.6 Ukládání subjektů do adresáře

Subjekty se do uživatelova adresáře mohou dostat dvěma způsoby – ručním přidáním nebo při vytvoření faktury subjektu, který zatím v adresáři není. Ruční přidání subjektu je možné na adrese:

```
POST https://api.openfaktura.cz/users/{user-id}/
↪  subjects?autoload={true/false}
```

Podobně jako při vytváření nového uživatele je i zde možnost nechat načíst data přímo z registru ARES, k tomu slouží parametr `autoload` a v uživatelském vstupu musí být v takovém případě přítomna hodnota `ico`. Při ručním zadání hodnot jsou názvy proměnných shodné s vstupem při vytváření nového uživatele. Jediným rozdílem je, že u subjektu se eviduje pouze celé jméno. Není již rozděleno na jméno a příjmení jako u uživatele. Dojde k vytvoření nového subjektu, jeho data jsou uložena do databáze a jeho URI je uživateli sdělena v odpovědi, pomocí hlavičky `Location`.

Pokud je subjekt přidán do adresáře vystavením faktury, veškeré operace proběhnou na pozadí a uživatel o nově vytvořeném subjektu není informován.

5.2.4.7 Tvorba faktury

Novou fakturu lze vytvořit odesláním jednoho ze tří následujících požadavků:

```
POST https://api.openfaktura.cz/invoices
POST https://api.openfaktura.cz/invoices?autoload={true/false}
POST https://api.openfaktura.cz/invoices?subject_id={subject-id}
```

První požadavek očekává na uživatelském vstupu informace o klientovi, tedy alespoň `customer_name`, `customer_street`, `customer_psc` a `customer_city`. Další parametry jako je `customer_ico` nebo `customer_email` jsou dobrovolné.

Druhý typ požadavku umožňuje načíst základní údaje o klientovi přímo z registru ARES. V takovém případě je v uživatelském vstupu vyžadována přítomnost pole `customer_ico`. I zde, pokud uživatel zadá nějaká data duplicitně, jsou preferovány údaje z uživatelského vstupu.

Třetí typ požadavku umožňuje jako klienta použít jeden ze subjektů v adresáři uživatele. Jako parametr v URI je potřeba předat jeho unikátní `id`, které uživatel zjistí v adresáři. Pokud je zároveň zadáno `subject_id` i nastaven `autoload` na `true`, provede se pouze načtení subjektu z adresáře.

Všechny tři metody se liší pouze ve způsobu získání informací o zákazníkovi. Zbytek procesu už je pro všechny shodný. Ač informace o klientovi k vytvoření faktury stačí, je možné zadat ještě například číslo faktury `number`. Pokud číslo faktury není zadáno, aplikace se jej pokusí odvodit pomocí funkce `getNextInvoiceNumber()`, která z databáze získá nejvyšší číslo vystavené faktury v tomto roce a zvýší jej o jedna. Pokud čísla nejsou čistě numerická a uživatel ve vstupu číslo faktury nezadal, je funkce ukončena a uživatelovi vrácena chybová hláška.

Mezi další parametry, které je možno nastavit jsou detaily platby jako způsob `payment_type`, měna `payment_currency` nebo variabilní a konstantní symbol `payment_vs` resp. `payment_ks`. Na závěr jde ještě upravit znění trojice textů přítomných na faktuře. Pokud nejsou texty přítomny v uživatelském vstupu, použijí se výchozí z účtu uživatele.

Na závěr jsou všechna data uložena do databázové tabulky `invoices`, vznikne též nový záznam v tabulce `payments`. Pokud klient ještě není v adresáři, je jeho záznam do adresáře přidán. ID nové faktury je uživateli opět předána pomocí hlavičky `Location`.

5.2.4.8 Správa zdrojů

V rámci aplikace jde většinu zdrojů upravovat. Kromě uživatelského účtu jde všechny zdroje smazat pomocí HTTP metody DELETE. Logicky lze mazat pouze konkrétní zdroje a ne jejich kolekce.

```
DELETE https://api.openfaktura.cz/invoices/{invoice-id}
```

Předchozí příklad smaže fakturu s `id` 20. Zároveň dojde ke smazání všech jejích položek a příslušného záznamu z tabulky `payments`, aby byla databáze udržována konzistentní. Při mazání i dalších operacích ovlivňujících víc než jednu tabulku zároveň jsou využívány transakce, aby bylo zajištěno, že změna proběhne ve všech tabulkách a ne jen v některých, v případě chyby. Pokud smazání proběhne úspěšně, je o tom uživatel informován pomocí hlavičky 204. Úspěšnost operace chápu tak, že zdroj na zadané URI po dokončení nebude existovat. Uživateli je tedy odeslána hlavička 204 i v případě, že zdroj na této URI neexistoval už před začátkem operace mazání.

Další operací se zdroji je jejich úprava pomocí metody PUT

```
PUT https://api.openfaktura.cz/invoices/{invoice-id}/
↪  items/{item-id}
```

Při úpravě jsou veškerá data brána pouze z uživatelského vstupu. Z databáze se vždy načtou aktuální hodnoty, pokud jsou proměnné přítomny v uživatelském vstupu, hodnoty se patřičně upraví. Uživatel tedy může zadat libovolný počet proměnných ke změně, neposkytnuté proměnné si zachovají svoji původní hodnotu. Jména proměnných jsou stejná jako při vytváření nových zdrojů pomocí metody POST. Na závěr jsou data aktualizována v databázi a uživatel je o úspěšném dokončení informován kódem 204. Úpravy informací o uživateli se promítnou do všech faktur na rozdíl od úprav informací o subjektu, které nijak neovlivní již vystavené faktury.

5.2.4.9 Tvorba QR platby

Vytvoření validního QR kódu se skládá ze dvou částí – vytvoření validního SPYAD řetězce (viz 4.2.1.7) a zakódování tohoto řetězce do podoby QR kódu. K vytvoření SPYAD řetězce je potřeba znát kód IBAN účtu, na který se má platba provést. Při hledání postupu výpočtu jsem narazil na kalkulátor přímo na stránkách České národní banky [35], kde v kódu stránky jsou přímo uvedeny funkce na výpočet kódu IBAN. Funkce jsem tedy z JavaScriptu přepsal do PHP a drobně upravil. Jako vstup tato sada funkcí bere číslo účtu rozdělené na část před a za pomlčkou a kód banky. Výstupem je pak IBAN kód účtu.

Další věcí, která stojí za zmínku ohledně SPYAD řetězce je zpráva příjemci. Podle specifikace [26] by řetězec měl obsahovat pouze čísla, a velká písmena A-Z. Je tedy potřeba se zbavit veškerých znaků s diakritikou a celý řetězec převést na velká písmena. Pro zbavení se diakritiky jsem využil funkci `remove_accents($text)` od Viliama Holuba z Karlovy univerzity [36]. Převod na velká písmena pak pomocí interní PHP funkce `strtoupper()`.

Celý řetězec je následně předán jako vstup knihovně **PHP QR Code**, jejímž výstupem je přímo obrázek, obsahující QR kód, odeslaný prohlížeči.

```
QRcode::png($SPYADString);
```

5.2.4.10 Export do PDF

Jak již bylo popsáno v návrhu, k exportu do formátu PDF se využívá knihovna, která jako vstup očekává HTML. Je tedy potřeba nejprve vysázet informace z faktury do HTML šablony. O to se stará třída `DefaultInvoiceTemplate`, která implementuje rozhraní `InvoiceTemplateInterface` (viz 4.2.2.2). Na vstupu tedy dostane instanci třídy `Invoice` a výstupem je HTML. Rozhraní je zde uděláno pro možnost v budoucnu nabídnout další šablony nebo šablony vytvořené přímo uživatelem. Třídy pro export zdaleka nejsou kompatibilní se všemi vlastnostmi CSS, bylo tedy potřeba šablonu několikrát upravovat a měnit, aby se po exportu do PDF zobrazila tak, jak je požadováno.

Vyexportované HTML je pak předáno jako vstup knihovny mPDF:

```
// setup MPDF
require_once __DIR__ . '/vendor/autoload.php';
ob_start(); // output buffering to prevent errors

// render PDF representation from HTML
$mpdf = new \Mpdf\Mpdf(['mode' => 'utf-8', 'format' => 'A4']);
$mpdf->SetHTMLFooter("<table class='table-format'
↪  style='font-size:90%; font-style: italic; width:
↪  100%'><tbody><tr><td style='text-align: left;' >Vytiskl: " .
↪  $author . " " . date('d.m.Y H:i', time()) . "</td><td
↪  style='text-align: left'>" . $inv->getText3() . "</td><td
↪  style='text-align: right'>Strana {PAGENO} / {nb}
↪  </td></tr></tbody></table></div>");
$mpdf->WriteHTML($html);
$mpdf->Output("Faktura-" . $inv->getNumber() . ".pdf",
↪  \Mpdf\Output\Destination::DOWNLOAD);
```

Nejprve se tedy nastaví rozměry dokumentu, který se bude exportovat, dále zápatí stránek ve vyexportovaném dokumentu, vloží se HTML vytvořené v šabloně a následně je zavolána funkce `Output()`, které jsou jako parametry předány jméno vyexportovaného souboru a konstanta značící vynucení ukládacího dialogu v prohlížeči uživatele. Ukázku vyexportované faktury je možné vidět na obrázku 5.2.

5.2.4.11 Párování plateb

Automatické přiřazování plateb je v aplikaci řešeno pomocí parsování emailů, zasílaných bankou, informujících o příchozí platbě. V konfiguraci lze nastavit, na jakou emailovou adresu jsou tyto emaily zasílány a přihlašovací údaje k této emailové schránce.

Po přihlášení ke schránce jsou staženy základní informace o schránce jako je počet zpráv, které obsahuje nebo adresáře, které se v ní nacházejí. Dále

Faktura 20180005

GUYS production

Datum vystavení: 11.06.2018
Datum splatnosti: 25.06.2018

Forma úhrady: Převodem

Dodavatel:

Petr Suchý
Smědčice 55
33824 Břasy
Česká republika
IČO: 02138221
Nejsme plátci DPH

Odběratel:

GoodCall s.r.o.
Václavské náměstí 846/1
110 00 Praha
Česká republika
IČO: 02765861
DIČ: CZ02765861

Platební údaje:

QR Platba

Způsob úhrady: Bankovní převod

Částka: **339.31 Kč**
Číslo účtu: **1025320267**
Kód banky: **6100**
Kód IBAN: CZ7961000000001025320267

Konstantní symbol: **0308**
Variabilní symbol: **20180005**

Fakturujeme Vám za dodané zboží či služby.

Označení položky	Cena	ks	Celkem
Lovecký salám	180	0.2	36
Vysočina	130	0.1	13
Párty chlebíček	30	2	60
Bio vídeňský párek	298	0.15	44.7
Domácí sekaná	138	1.345	185.61

Celkem k úhradě: 339.31 Kč

Zboží zůstává až do úplného uhrazení majetkem dodavatele.

Vytiskl: Petr Suchý 22.06.2018 00:12 *Faktura vytvořena v opensource systému OpenFaktura.* *Strana 1 / 1*

Obrázek 5.2: Ukázka faktury vyexportované do PDF

je zkontrolována přítomnost adresáře `.processed`. Pokud tento adresář není přítomen, aplikace jej vytvoří.

Všechny zprávy ve schránce se v cyklu projdou. U každé zprávy je nejprve z hlavičky zjištěna doména, ze které byla zpráva odeslána a podle toho je rozhodnuto, zda jde o podporovanou či nepodporovanou banku. Pokud banka není podporována, zpráva se rovnou označení k přesunutí do složky `.processed`. Pokud jde o banku, která je podporována a existuje pro ni parsovací funkce, je tato funkce zavolána a zpráva je jí předána.

```
switch ($message["header"]->from[0]->host) {
        case "airbank.cz":
                $res = parseAirBank($message);
                break;
        case "kb.cz":
                $res = parseKB($message);
                break;
        case "tbs.csob.cz":
                $res = parseCSOB($message);
                break;
        default:
                echo "Default parser: ";
                $res = UNKNOWN_BANK;
}
```

Parsovací funkce převede tělo konkrétní zprávy do `UTF-8` kódování a následně se pokouší pomocí regulárních výrazů z textu zprávy vyextrahovat potřebné informace. Pokud se jí podaří získat všechny potřebné informace (především číslo účtu na který byla platba přijata a variabilní symbol platby) a v databázi je nalezen záznam se shodujícími se údaji, je přijatá částka přičtena k dosavadní přijaté částce a zároveň je aktualizován čas poslední přijaté platby. Následně je zpráva považována za úspěšně přečtenou a je označena k přesunutí. Parsovací funkce vypadá například takto:

```
function parseKB($message){
        $text = iconv($message["charset"], "utf-8",
        ↪  $message["plainmsg"]);

        $matches = array();
        preg_match('/na účet číslo
        ↪  ([0-9]+([-]{1}[0-9]+|)\/[0-9]+)/', $text, $matches);
        if (!isset($matches[1]))
                return INVALID_SCHEMA;
        $accountNumber = $matches[1];

        preg_match('/částka ([0-9 ,]+)/', $text, $matches);
```

```
        if (!isset($matches[1]))
                return INVALID_SCHEMA;
        $amount = $matches[1];

        // ...

        return updatePaymentInDatabase($accountNumber, $amount,
        ↪  $date, $vs);
}
```

Pokud email neobsahuje nějakou povinnou informaci, nebo v databázi neexistuje záznam, ke kterému by šla platba přiřadit je zpráva označena k přesunutí bez dalších akcí. Po skončení skriptu **paymentChecker.php**, který zmíněnou funkcionalitu obsahuje, tedy ve schránce zůstanou pouze zprávy, které obsahovaly všechny potřebné údaje, ale nebylo je kvůli chybě v komunikaci s databází, možné přiřadit.

Momentálně je implementována parsovací funkce pro Komerční banku, ČSOB a Air Bank. Není obtížné vytvořit parsovací funkci pro další banky, ale bohužel se mi nepodařilo získat formát emailu, který posílají. Během implementace se dále ukázalo, že mBank a Equa bank neposílají v notifikačním emailu dostatečné množství informací pro jednoznačnou identifikaci platby.

5.2.4.12 Odesílání faktur emailem

Z analýzy vyplynulo, že určitá část uživatelů si přeje mít možnost odesílat faktury emailem přímo z prostředí aplikace. Tato funkčnost byla tedy implementována na adrese detailu faktury:

```
POST https://api.openfaktura.cz/invoices/{invoice-id}
↪  ?action=sendmail&useDefault={true/false}
```

Bez uvedení parametru `?action=sendmail` se neprovede žádná operace a je vrácena chyba 400 s hláškou o neznámé operaci. Parametr `useDefault` určuje, zda se jako předmět a tělo zprávy mají použít výchozí hodnoty z nastavení. Faktura se může poslat buď jako příloha emailu nebo v rámci těla jako odkaz na její stažení. Je potřeba to rozlišit hodnotou proměnné `body_type` v uživatelském vstupu. Přípustné jsou hodnoty `link` nebo `attachment`. Uživatel může tělo zprávy i předmět dodat pomocí proměnných `email_subject` a `email_body`, stejně tak jako adresu, na kterou se má zpráva odeslat pomocí proměnné `customer_email`.

Předmět zprávy i tělo zprávy mohou obsahovat zástupné řetězce, které jsou před odesláním nahrazeny konkrétními daty z údajů o faktuře. Jedná se o:

- `{invoice_number}` Nahradí se skutečným číslem faktury.

- `{invoice_amount}` Nahradí se částkou a měnou faktury.
- `{invoice_due_date}` Nahrazeno datem splatnosti.
- `{invoice_public_url_pdf}` Nahradí se URL, na které lze stáhnout fakturu ve formátu PDF. Fungují i varianty s koncovkou `html` a `json`.
- `{supplier_name}` Nahradí se jménem dodavatele.

Pokud jsou dostupná všechna potřebná data, jsou parametry předány funkci `sendEmail()`, která se postará o odeslání emailu. V závislosti na hodnotě proměnné `body_type` jsou případně předány ještě informace o příloze emailu, která je stažena a přiložena před odesláním k emailu.

5.2.4.13 Vyhledávání

Vyhledávání je implementováno u faktur, kde se prohledávají jména položek faktury, číslo faktury, částka faktury, všechny informace o zákazníkovi. Pokud některý z těchto parametrů obsahuje vyhledávaný výraz je faktura přidána do odpovědi. Nápodobně je vyhledávání implementováno také v adresáři uživatele, kde jsou kromě `id` a `owner_id` subjektu prohledávány všechny údaje. Vyhledávání dále podporují také položky ceníku, kde je prohledáváno jméno položky a posledním zdrojem, který podporuje vyhledávání jsou bankovní účty uživatele, kde se shoda hledá v názvu účtu a jeho čísle.

Vyhledávání ve všech zdrojích má shodnou podobu – podporováno je pouze u metody GET a vyhledávaný výraz je potřeba uvést jako hodnotu parametru `q`, tedy například:

```
GET https://api.openfaktura.cz/invoices?q=pivo
```

Výsledkem takového požadavku budou faktury, která mají v položkách například „Kvasnicové pivo“, jejichž klient se jmenuje třeba „Karel Pivovarník“ a nebo pokud klient bydlí v „Pivovarské“ ulici. Před porovnáváním s databází jsou všechny hodnoty převedeny na malá písmena. Vyhledávání je tedy case insensitive.

5.3 Frontend

Před implementací mojí diplomové práce jsem neměl v podstatě žádné zkušenosti s tvorbou aplikace v JavaScriptu. Vývoj mi tedy zabral výrazně více času, než jsem očekával. Webové rozhraní je dostupné na adrese:

```
https://openfaktura.cz
```

5.3.1 Úpravy API

Během implementace webového rozhraní vyšlo najevo několik nutných úprav API.

5.3.1.1 HTTP metoda OPTIONS

První úpravou je přidání podpory pro HTTP metodu `OPTIONS`. Ukázalo se totiž, že při používání funkce `jQuery.ajax()` je před skutečným odesláním požadavku odeslán takzvaný „pre-flight request“, který využívá právě metodu `OPTIONS`. Pokud na tento požadavek nedostane kladnou odpověď, k poslání skutečného požadavku vůbec nedojde a celá operace končí chybou. Možnost přístupu pomocí metody `OPTIONS` byla tedy přidána do všech zdrojů. Zároveň bylo zjištěno, že nestačí jako odpověď na dotaz metodou `OPTIONS` odpovídat hlavičkou `Allow`, ale je potřeba hlavička `Access-Control-Allow-Methods`.

5.3.1.2 Hlavička pro CORS

Tato úprava souvisí s předchozí úpravou. CORS je zkratkou pro „cross-origin resource sharing“ neboli sdílení prostředků z jiných domén. Pokud jsou totiž požadavky odesílány na jinou doménu, než na které web běží (v mém případě z domény `openfaktura.cz` na doménu `api.openfaktura.cz`), je z bezpečnostních důvodů vyžadováno, aby cíl požadavku (tedy `api.openfaktura.cz`) umožňoval přístup z jiných domén. Bylo tedy potřeba do všech odpovědí přidat hlavičku `Access-Control-Allow-Origin: *`, která značí, že požadavky mohou přicházet ze všech domén.

5.3.1.3 Úpravy funkcí

Kromě odhalení několika překlepů a chyb bylo rozpoznáno i několik nedostatků. Například při ukládání nové faktury bylo potřeba v odpovědi o úspěšném vytvoření nové faktury odesílat také její ID tak, aby webové rozhraní rovnou mohlo přidat i její položky, které uživatel zadal při vytváření faktury. Dále bylo nutné do API přidat funkci na získání následujícího čísla faktury, aby uživatel už při vyplňování údajů věděl, jaké bude mít nově vytvářená faktura číslo a případně ho mohl upravit.

5.3.2 Rozdělení zdrojového kódu do souborů

Zdrojové kódy webového rozhraní lze rozdělit do 3 skupin podle jazyka, ve kterém jsou vytvořeny – PHP kódy, CSS styly a JavaScript kódy. PHP soubory jsou pouze dva. Tím hlavním je **index.php**, který klientovi odešle kostru stránky obsahující informaci o umístění API a několik základních definovaných prvků. Nejdůležitějším obsahem souboru **index.php** tedy je:

```
<link rel="stylesheet" href="css/openfaktura.css">
<script>
<?php
echo "var apiUrl = \"https://api.openfaktura.cz\";\n";
?>
</script>
<!-- linky na knihovny Bootstrap a jQuery -->
<script src="js/functions.js"></script>
<script src="js/invoices.js"></script>
<script src="js/user.js"></script>
```

Obsah a význam souboru **aresLoader.php** je vysvětlen v sekci 5.3.3.4;

Soubory kaskádových stylů jsou uloženy v adresáři **css**. Během implementace vznikl jediný CSS soubor, ve kterém je pouze několik málo pravidel. K úpravě vzhledu, jsou v drtivé většině, používány nástroje knihovny Bootstrap.

JavaScriptové soubory během implementace vznikly 3 a jsou uloženy v adresáři **js**. Jde o soubory `user.js`, `invoices.js` a `functions.js`. V prvním jmenovaném souboru jsou funkce, které se nějak týkají uživatele – funkce na registraci, změnu nastavení atd. V souboru `invoices.js` jsou pak funkce týkající se faktur. V posledním souboru jsou funkce, které přímo nesouvisí ani s uživatel, ani s fakturami, ale přesto jsou pro fungování systému potřeba.

5.3.3 Funkce

Podobně jako v sekci API i zde popíšu implementaci významných funkcí a pro detailní pochopení odkážu čtenáře na zdrojový kód v příloze.

5.3.3.1 Registrace

Při první návštěvě webu uživatel uvidí v podstatě jen uvítací zprávu a horní lištu s menu, které je pro nepřihlášeného uživatele omezeno na dvě položky – „Registrace“ a „Přihlášení“. V této části se budeme věnovat registraci. Při kliknutí na „Registrace“ je zavolána funkce `drawRegistrationForm()`, která zajistí, že se v hlavním okně vykreslí formulář pro registraci nového uživatele, obsahující pole pro všechny hodnoty, která uživatel při registraci může zadat. Je zde přítomno i tlačítko „ARES“ pro načtení údajů z registru ARES na základě zadaného IČO. Ve chvíli kdy uživatel dokončí vyplňování formuláře a klikne na modré tlačítko „Registrovat“ pod formulářem, je zavolána funkce `registerNewUser()`, která zajistí načtení hodnot z formulářových polí, zkontroluje vyplnění povinných polí – případně zahlásí chybu a nakonec odešle hodnoty na registrační endpoint API. Uživatel je následně informován o výsledku registrace. Uživateli přijde na email potvrzovací odkaz, stejně jako kdyby se registroval přímo do API bez použití webového rozhraní.

5.3.3.2 Přihlašování, autentizace a odhlášení

Druhou položkou menu, pro nepřihlášeného uživatele, je tlačítko „Přihlášení“. Kliknutím na něj se otevře `Modal Box` obsahující políčka pro zadání emailu a hesla. Zároveň je zde přepínač pro dočasné zobrazení hesla. Stiskem tlačítka „Přihlásit“jsou data odeslána ke kontrole do API. Pokud přijde kladná odpověď, uživatel je přihlášen a `Modal Box` je zavřen. Zároveň dojde k překreslení rozhraní tak, aby uživatel viděl i položky, které jsou vidět jen pro přihlášené uživatele. Do proměnné `api_token` je uložen uživatelův *api_token*. Ten je zároveň uložen i do cookies s týdenní platností:

```
setCookie("openfaktura_token", result.token, 7);
```

Při každém načtení stránky, aplikace zkontroluje přítomnost zmíněné cookie. Pokud je přítomna, hodnotu znovu načte do proměnné `api_token`. Uživatel se tedy nemusí znovu přihlašovat. *Api_token* je využíván při odesílání veškerých požadavků na API – uživatel jím prokazuje svoji identitu.

Odhlášení uživatele je poměrně prosté – je zneplatněna zmíněná cookie a stránka je znovu načtena.

```
function logout() {
        setCookie("openfaktura\_token", "", -7);
        document.location.reload();
};
```

5.3.3.3 Zobrazení notifikací

V kostře kterou posílá server klientovi je připraven ještě jeden `<div>`, který slouží pro zobrazování notifikací. Funkce `showNotification()`, která se o zobrazení notifikací stará, zobrazí na vrchu stránky na několik vteřin Bootstrapový prvek `Alert`. Jsou podporovány různé druhy (barvy) notifikací, podle povahy notifikace, zda upozorňuje na úspěch či chybu. Typ notifikace i její obsah je předáván funkci jako parametr. Zobrazení notifikací je využíváno například při úspěšném vytvoření nové faktury, kdy je uživateli vykreslen seznam faktur a na vrchu stránky se mu zobrazí zelená notifikace s textem, že faktura byla úspěšně vytvořena.

5.3.3.4 Načítání dat z registru ARES

Aby uživatel mohl vidět data, již při zadávání, bylo potřeba i ve webovém rozhraní udělat funkci, umožňující načítání dat z registru ARES. Bohužel registr ARES nepřidává do odpovědí hlavičky zmíněné v sekci 5.3.1.2. Nebylo tedy možné posílat požadavky přímo. Vznikl tedy skript **aresLoader.php**, který slouží jako prostředník. Jeho obsahem je pouze funkce `loadInfoFromARES()`,

stejná jako v API. Požadavek je tedy místo na ARES odeslán na **aresLoader.php**, jedná se o stejnou doménu, není zde tedy problém s CORS. **aresLoader.php** pošle požadavek na ARES a jako odpověď webové aplikaci vrátí JSON objekt s hodnotami. Funkce na načítání dat se ve webovém rozhraní využívá všude tam, kde jsou zadávány údaje o subjektech. Skrývá se vždy pod tlačítkem „ARES“.

5.3.3.5 Nastavení

Webové rozhraní podporuje úpravu informací o uživateli, společně s úpravou jeho uživatelských nastavení. Náhled můžeme vidět na obrázku 5.3. Formulářová pole jsou velmi podobná těm při registraci. Oproti registraci je zde navíc pole „ID“, které je společně s polem „Email“ pouze informační a jeho hodnotu nelze změnit. Navíc zde uživatel může nastavit i preference ohledně výchozích textů emailů. Ukládání probíhá velmi podobně jako při registraci. Je zavolána funkce `updateUserSettings()`, která data vyčte z formulářových polí, zkontroluje přítomnost povinných polí a odešle data pomocí HTTP metody `PUT` na API.

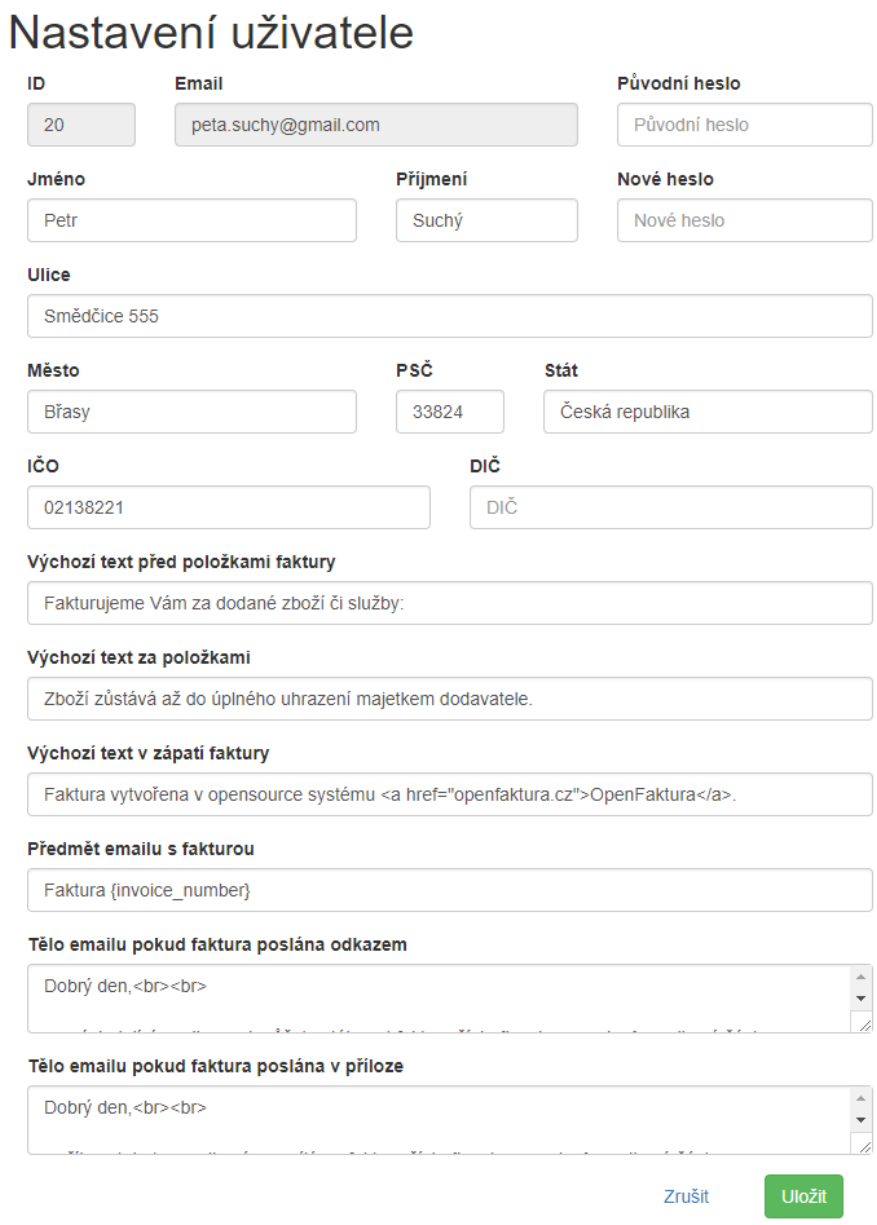
5.3.3.6 Tvorba a úprava faktur

Vytváření i úprava faktur probíhá ve stejném formuláři (viz obrázek 5.4). V obou případech je vykreslení formuláře zajištěno funkcí `drawInvoiceForm()`. Při vytváření faktury se již nic dalšího neděje, při úpravě jsou pomocí funkce `updateInvoiceCallback()` předvyplněny současné hodnoty. Uživatel má možnost přidávat nebo upravovat informace o zákazníkovi, a to buď hodnotami z registru ARES (pomocí tlačítka „ARES“ po zadání IČO) nebo také vyplnit pole údaji z adresáře (viz obrázek 5.5).

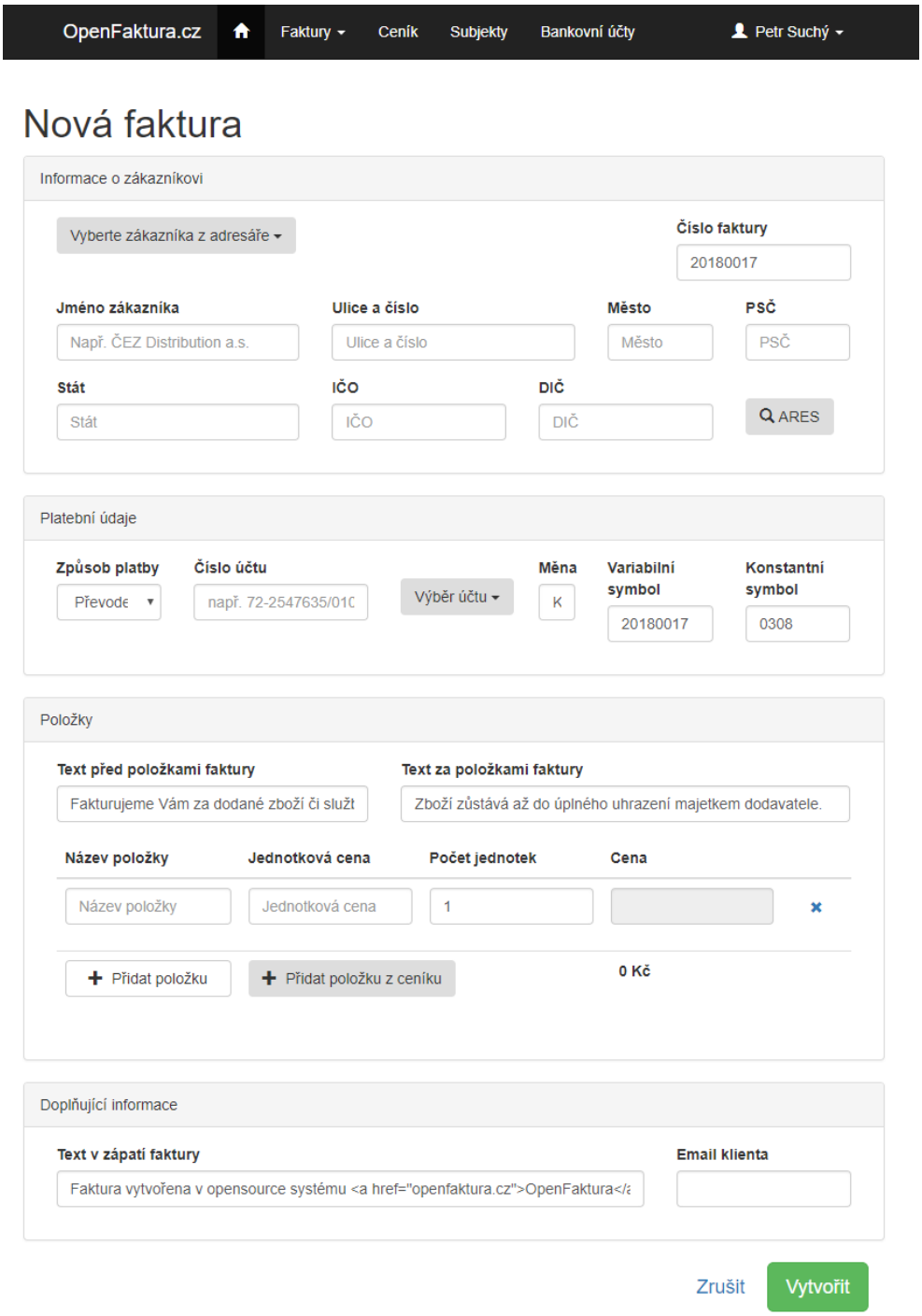
V sekci s údaji o platbě je uživateli umožněno předvyplnit číslo účtu číslem jednoho z účtů, které má uložené v `accounts`. Příklad takového přidání můžeme vidět na obrázku 5.6.

Implementačně nejnáročnější bylo přidávání položek. Při tvorbě nové faktury je zde ve výchozím stavu jedna prázdná položka. Uživatel může položky libovolně přidávat a odebírat pomocí příslušných tlačítek (viz obrázek 5.7). Při úpravě faktury jsou zobrazeny již existující položky, které stejně jako v případě tvorby nové faktury může uživatel libovolně upravovat. Pokud je u položky vyplněna jednotková cena a množství, je automaticky dopočítána celková cena položky. Zároveň je přepočítána i celková cena faktury. K přepočtení hodnot dojde po každé změně v ceně nebo množství položky.

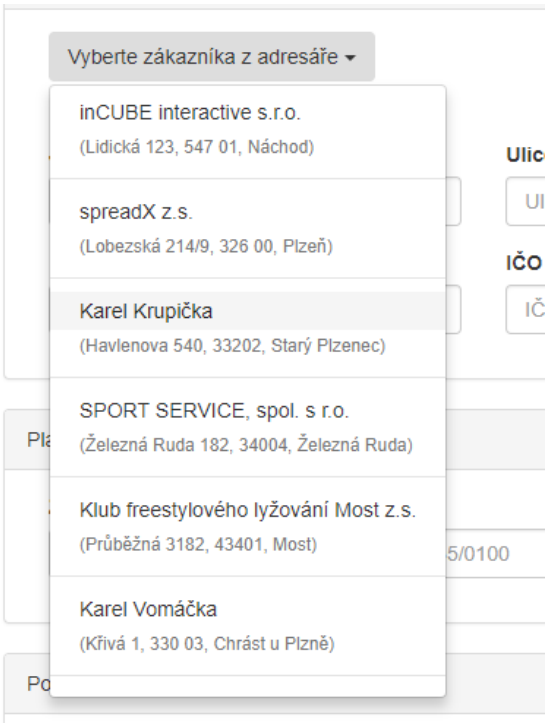
Při uložení nové faktury se nejprve v API vytvoří faktura a pak jsou k ní v cyklu přidávány jednotlivé položky, které uživatel zadal. Přidány jsou pouze



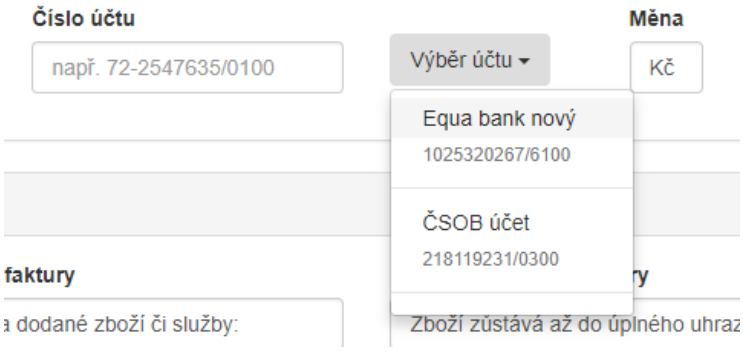
Obrázek 5.3: Screenshot uživatelského nastavení ve webovém rozhraní aplikace



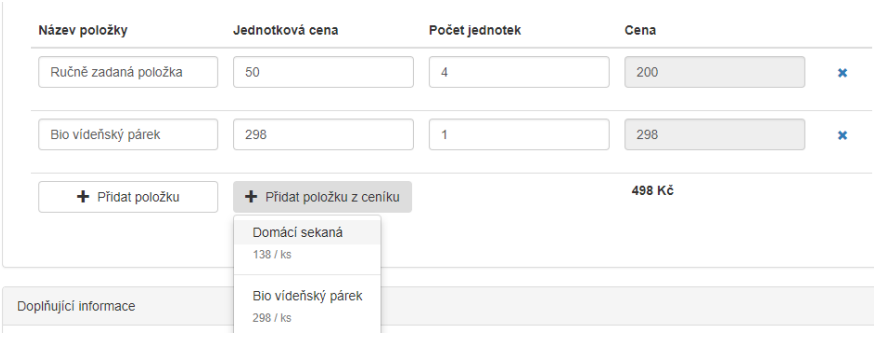
Obrázek 5.4: Screenshot webového rozhraní tvorby nové faktury – celek



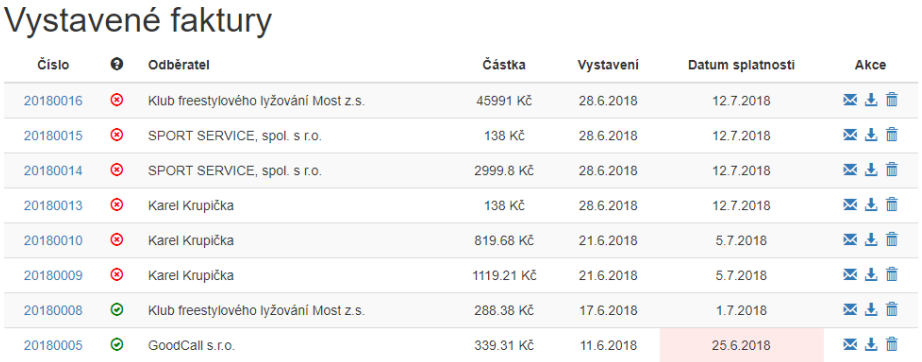
Obrázek 5.5: Webové rozhraní tvorby nové faktury – výběr zákazníka



Obrázek 5.6: Webové rozhraní tvorby nové faktury – výběr účtu



Obrázek 5.7: Webové rozhraní tvorby nové faktury – přidání položky



Obrázek 5.8: Webové rozhraní – přehled faktur

položky, které mají vyplněny jméno, jednotkovou cenu i množství. Pokud tato podmínka není splněna, je položka ignorována. Při úpravě existující faktury je potřeba rozlišit položky, které bylo pouze upraveny, položky, které byly nově přidány a položky které byly naopak odebrány. Nové položky je potřeba k faktuře přidat pomocí `POST`, existující položky je potřeba aktualizovat pomocí metody `PUT` a zaniklé položky je třeba z aplikace smazat pomocí metody `DELETE`.

5.3.3.7 Operace s fakturami

V rámci přehledu všech faktur uživatele (obrázek 5.8), je kromě základních informací o faktuře, vpravo od čísla faktury, zobrazen piktogram, značící, zda je faktura uhrazena. Pozadí pole ve sloupci „Datum splatnosti“ je červené, pokud faktura není uhrazena a zároveň už je po jejím datu splatnosti. Tři ikonky v právem sloupci pak umožňují konkrétní fakturu odeslat emailem, stáhnout

nebo smazat. Stažení i smazání jsou celkem přímočaré operace. V případě kliknutí na ikonku obálky se otevře `Modal Box` s předvyplněným předmětem a tělem emailu. Pokud byla k faktuře přiřazena emailová adresa, je též předvyplněna. Uživatel může údaje upravit a následně email i s fakturou odeslat přímo zákazníkovi.

5.3.4 Neimplementovaná funkcionalita

Z důvodu neočekávaně pomalého postupu vývoje frontendu jsem bohužel nestihl do webového rozhraní implementovat funkce na úpravu bankovních účtů uživatele, subjektů v adresáři a položek ceníku. V API tato funkčnost je a položky přidané v API lze používat i ve webovém rozhraní při vytváření faktur. Není však zatím možné je přidávat a upravovat přímo z webového rozhraní.

Kapitola 6

Testování a další rozvoj

V této kapitole popíšu testování, kterým aplikace prošla a nastíním několik návrhů na další rozvoj aplikace v blízké i vzdálenější budoucnosti.

6.1 Testování

Aplikace byla podrobena zátěžovému testování API a uživatelskému testování webového rozhraní. Bohužel v návaznosti na zdlouhavý vývoj webového rozhraní nezbylo moc času na testování ani případné úpravy aplikace.

6.1.1 Zátěžové testy

Provedl jsem malou sadu zátěžových testů. Testy jsem dělal v noci, aby servery, na kterých aplikace běží měly co nejmenší úroveň zatížení. Testoval jsem vždy sérii stejných operací a měřil minimální, maximální a průměrný čas dokončení operace. Testoval jsem z prostředí PHP pomocí knihovny cURL.

6.1.1.1 Dotaz faktury uživatele

V tomto testu bylo v cyklu přistupováno na URL:

```
GET https://api.openfaktura.cz/invoices
```

Tabulka 6.1: Shrnutí zátěžových testů API

Cíl dotazu	Minimální čas [s]	Maximální čas [s]	Průměrný čas [s]
GET /invoices	0.00784	0.576082	0.013734
GET /user/20	0.005002	0.27088	0.008989
GET /invoices?q=bio	0.066093	0.220067	0.080547
POST /invoices	0.011607	0.343666	0.017996

Provedl jsem 10 000 opakování v každé sérii. Celkově tři série. Průměrná doba na dokončení jednoho požadavku je v tomto případě 14 ms. Minimální doba odezvy byla 7 ms a maximální pak 576 ms, což je poměrně velký rozptyl.

6.1.1.2 Dotaz uživatelský profil

V tomto testu bylo v cyklu přistupováno na URL:

```
GET https://api.openfaktura.cz/users/20
```

Provedl jsem 10 000 opakování v každé sérii. Celkově tři série. Průměrná doba na dokončení jednoho požadavku byla v tomto případě 9 ms. Minimální doba odezvy byla 5 ms a maximální pak 270 ms. Zde je rozptyl o polovinu menší než v případě dotazu na faktury.

6.1.1.3 Dotaz vyhledávání ve fakturách

V tomto testu bylo v cyklu přistupováno na URL:

```
GET https://api.openfaktura.cz/invoices?q=bio
```

Provedl jsem 1 000 opakování v každé sérii. Celkově pět sérií. Při větším množství opakování jsem narážel na časový limit pro běh PHP skriptu. Průměrná doba na dokončení jednoho požadavku byla v tomto případě 80 ms. Minimální doba odezvy byla 66 ms a maximální pak 220 ms. Rozptyl je zde ze všech testů nejmenší, avšak průměrná doba na dokončení požadavku je zde o řád vyšší.

6.1.1.4 Dotaz vytvoření faktury

V tomto testu bylo v cyklu přistupováno na URL:

```
POST https://api.openfaktura.cz/invoices
```

Provedl jsem 5 000 opakování v každé sérii. Celkově dvě série. Průměrná doba na dokončení jednoho požadavku byla v tomto případě 18 ms. Minimální doba odezvy byla 11 ms a maximální pak 344 ms. Zde je rozptyl taky poměrně výrazný

6.1.1.5 Shrnutí testů

Jak je vidět v souhrnné tabulce 6.1, suverénně nejnáročnější operací je vyhledávání ve fakturách. A to je zatím systém poměrně čistý (v databázi je několik málo desítek faktur). S rostoucím množstvím bude nejspíš čas zpracování vyhledávacího dotazu ještě narůstat. Nemyslím si ale, že lineárně s počtem faktur v databázi. Osobně největší zátěž tohoto dotazu shledávám v prohledávání položek faktur. Pokud se to dlouhodobě ukáže jako výkonnostně příliš náročné, bude možná nutné od této funkcionality upustit.

6.1.2 Uživatelské testování

Výsledné webové rozhraní bylo otestováno několika uživateli. Během testování vzešlo několik návrhů a připomínek, které se zde pokusím popsat. Většina připomínek směřovala ke grafice systému, což jsem očekával.

6.1.2.1 Potvrzení účtu

První připomínkou bylo, že při ověřování emailové adresy je uživatel odkázán na API. Funkčně je to v pořádku, ale návštěva odkazu, který uživatel obdržel emailem ho přivede na stránku která zobrazí velmi prostý textový výpis. Pro uživatele je to matoucí a mají pocit, že se něco pokazilo. Řešením je při návštěvě ověřovacího odkazu nějak rozpoznávat, zda je klient stroj či nikoliv a podle toho upravit zobrazení stránky.

6.1.2.2 Grafická nesourodost

Vlivem přejímání kusů kódu z různých tutoriálů a ukázek může webové rozhraní působit graficky nejednotně. S touto výtkou souhlasím. Je potřeba v rámci aplikace sladit alespoň barvy a styl tlačítek a dialogů. Stejně tak hlášení chyb. Při chybě ve formuláři je chyba vypsána přímo v rámci formuláře. V případě chyby v odeslání formuláře do API je chyba zobrazena jako notifikace.

6.1.2.3 Jazyk chybových hlášek

Celé API jsem navrhoval v angličtině, respektive s anglickými názvy proměnných a bohužel i anglickými texty odpovědí na dotazy. To se ukázalo jako velmi nepraktické. Když při chybě přímo vypíšu chybovou zprávu, vypíše se do českého webového rozhraní anglický text. Řešením je buď přepsat všechny odpovědi do češtiny nebo poskytování odpovědi ve více jazykových mutacích. Otázkou je, zda to má u aplikace, která se zaměřuje jen na české uživatele, smysl.

6.1.2.4 Kompatibilita

Aplikace se zobrazovala shodně ve všech 4 testovaných prohlížečích – Google Chrome, Microsoft Edge, Mozzilla Firefox a Safari. V Mozzille Firefox docházelo k občasným aktualizačním chyba. Přisuzuji to ale spíše používání cache uvnitř prohlížeče. Na smartphonu bohužel aplikace nefunguje podle představ. Bude zřejmě potřeba vytvořit verzi rozhraní, která bude lépe optimalizována pro mobilní zařízení.

6.2 Další rozvoj

Aplikace je můj vlastní nápad a za celou dobu vývoje mě neomrzel. Vývoji aplikace se tedy chci věnovat i nadále. Aplikace se může rozvíjet mnoha směry a o mnoho funkcí. Troufnu si tvrdit, že funkcionalita, která je momentálně implementována je méně než 10 % toho, co by aplikace měla umět, aby byla v současné situaci opravdu konkurenceschopná. Zatím může většině existujících řešení konkurovat pouze cenou a ne funkcionalitou. Další rozvoj aplikace jsem si dovolil rozdělit do dvou kategorií – krátkodobý a dlouhodobý plán.

6.2.1 Krátkodobý plán

Základním krátkodobým cílem je upravit aplikaci tak, aby byla více robustní. V současné situaci se u spousty vstupních proměnných nijak neověřuje jejich validita. S tím souvisí i zotavení z chyb. V tuto chvíli dokáže jedna nesprávná hodnota celou aplikaci dostat do kritické chyby, ze které již není schopna se vzpamatovat.

Pokud by se aplikaci podařilo udělat více robustní a jít takzvaně s kůží na trh, vyvstává další důležitý bod krátkodobého plánu – legalita aplikace. S nastupujícím GDPR by bylo víc než vhodné aplikaci zanalyzovat z právního hlediska a případně vynutit u uživatelů při registraci souhlas s podmínkami nebo upravit funkcionalitu tak, aby nebyla v rozporu se zákonem.

Dalším krátkodobým cílem je pro mě zlepšení grafického vzhledu aplikace. A to jak webového rozhraní, tak třeba i vzhledu vyexportované faktury.

Posledním bodem krátkodobého plánu pro mě je rozšíření možností exportu – přidat možnost exportu do XLS a dalších formátů. Přesunout aplikaci na dedikovaný server, protože na současném hostingu je omezena limitem 500 odeslaných emailů denně. Dále rozšířit počet bank podporovaných automatickým párováním plateb a jako poslední si myslím, že je potřeba co nejdříve implementovat plnou podporu pro plátce DPH.

6.2.2 Dlouhodobý plán

Do dlouhodobého plánu řadím, navzdory výsledkům z dotazníku, propojení aplikace se systémem EET. Tím by aplikace získala značnou skupinu potenciálních uživatelů přicházejících s dalšími vlnami zavádění povinnosti elektronické evidence tržeb.

Za další dlouhodobý cíl považuji přepracování systému exportu faktur, aby aplikace mohla nabídnout více možností, jak fakturu vytvořit a ne jen pomocí jedné konkrétní knihovny s přesně definovaným vstupem. S tím souvisí i možnost exportu faktur v různých formátech.

Třetím dlouhodobým cílem je pro mě propojení s existujícími službami. Aby uživatelé mohli snadno změnit službu, ve které spravují svoje faktury. Součástí tohoto cíle je tedy především schopnost spolupracovat s API ostatní poskytovatelů online správy faktur.

Závěr

Cílem práce bylo zanalyzovat požadavky a na jejich základě navrhnout, implementovat a otestovat webovou aplikaci pro správu a tvorbu faktur s možností ovládání přes API i přes webové rozhraní. Důležitou součástí aplikace měla být schopnost exportovat faktury do formátu PDF a automatické párování faktur s příchozími platbami za pomoci aplikačního rozhraní bank.

V rámci rešeršní části práce při hlubším zkoumání vlastností a požadavků směrnice PSD2 vyšlo najevo, že její přínosy nebudou v dohledné době využitelné pro komunikaci s bankou přes aplikační rozhraní a tak jsem se rozhodl jeden z bodů zadání, kterým je párování plateb řešit přes notifikační emaily zasílané bankou.

Analytická část pak pomohla jasně definovat základní sadu funkcí, požadavky a preference uživatelů a umožnila následně navrhnout aplikaci, ovládnou přes aplikační rozhraní splňující standardy architektury REST. V rámci návrhu jsem se rozhodl aplikaci včetně rozhraní implementovat v jazyce PHP a veškerá uživatelská data ukládat do databáze. Díky volbě databázových knihoven je aplikace schopna pracovat se širokou škálou databází. Při návrhu webového rozhraní jsem se pak zaměřil na snadnou přenositelnost a co nejmenší závislost na serveru na kterém aplikace běží. I díky tomu jsem se rozhodl celé webové rozhraní implementovat jako aplikaci napsanou v jazyce JavaScript běžící kompletně na straně klienta.

Implementovaná aplikace splňuje požadavky zadání – nabízí správu faktur, jejich export do formátu PDF, automatické párování plateb, API i přístup přes webové rozhraní, které zatím nabízí jen základní funkce nutné pro vystavování a správu faktur.

Přínos aplikace zatím není velký, nabízí spíše minimální nutnou sadu funkcí, ale věřím že s jejím postupujícím vývojem dokáže oslovit drobné podnikatele a firmy a nabídnout jim alternativu k placeným komerčním řešením. Můj osobní cíl vytvořit si funkční alternativu pro správu faktur aplikace splnila.

Literatura

[1] ČSSZ: Přehled o počtu OSVČ podle stavu k 31.12.2017 [online]. 2018, [cit. 2018-04-10]. Dostupné z: http://www.cssz.cz/NR/rdonlyres/59C768E5-1CDD-4168-8D96-525B9B51B00E/0/prehled_o_poctu_osvc_dle_ossz_a_kraju_prosinec_2017.pdf

[2] Cígler, M.: iDoklad slaví 100 000 zákazníků [online]. 2016, [cit. 2018-04-10]. Dostupné z: https://www.idoklad.cz/blog/idoklad-slavi-100-000-zakazniku

[3] Cígler, M.: Zavádíme placené tarify. Vše, co potřebujete vědět, najdete zde [online]. 2017, [cit. 2018-04-10]. Dostupné z: https://www.idoklad.cz/blog/zavadime-placene-tarify-vse-co-potrebujete-vedet-najdete-zde

[4] Svobodová, E.: Podnikatel podle nového Občanského zákoníku [online]. 2013, [cit. 2018-04-13]. Dostupné z: http://www.accontes.cz/podnikatel-podle-noveho-obcanskeho-zakoniku

[5] Janák, T.: Náležitosti faktury: co musí obsahovat faktura [online]. 2014, [cit. 2018-04-13]. Dostupné z: https://www.superfaktura.cz/blog/nalezitosti-faktury-co-musi-obsahovat-faktura/

[6] Konečná, J.: Doklad a faktura neplátce [online]. 2017, [cit. 2018-04-13]. Dostupné z: http://www.jakpodnikat.cz/faktury-doklady.php

[7] Revue, I.: PSD2: malá revoluce v platebních službách [online]. 2017, [cit. 2018-05-03]. Dostupné z: https://ictrevue.ihned.cz/c3-65786220-0ICT00_d-65786220-psd2-mala-revoluce-v-platebnich-sluzbach

[8] Šimková, M.: Sdělení České bankovní asociace k přechodnému období PSD 2 a nového zákona o platebním styku [online]. 2018, [cit. 2018-05-03]. Dostupné z: https://www.czech-ba.cz/sites/default/files/tz_psd2_a_novy_zakon_o_platebnim_styku.pdf

[9] PSD2, nová směrnice o platebním styku [online]. 2018, [cit. 2018-05-03]. Dostupné z: https://www.csob.cz/portal/-/n180112

[10] Kristen, V.: PSD2 je to nejpodstatnější, co se v bankovním sektoru za poslední dekádu událo, říká expert [online]. 2018, [cit. 2018-05-03]. Dostupné z: http://www.info.cz/byznys/psd2-je-to-nejpodstatnejsi-co-se-v-bankovnim-sektoru-za-posledni-dekadu-udalo-rika-expert-22464.html

[11] Brejčák, P.: Přichází revoluční PSD2: Jak se změní svět fintech startupů? [online]. 2018, [cit. 2018-05-03]. Dostupné z: https://tyinternety.cz/startupy/prichazi-revolucni-psd2-se-zmeni-svet-fintech-startupu/

[12] Mosnáková, M.: PSD2 a nová platební služba: Nepřímé udělení platebního příkazu [online]. 2018, [cit. 2018-05-03]. Dostupné z: https://www.epravo.cz/top/clanky/psd2-a-nova-platebni-sluzba-neprime-udeleni-platebniho-prikazu-107127.html

[13] Suchý, P.: Spousta z vás jistě dělá faktury v nějakém online nástroji...[online]. 2018, [cit. 2018-05-14]. Dostupné z: https://www.facebook.com/groups/zivnostnici/permalink/1636679946401406/

[14] Suchý, P.: Online systémy pro správu faktur [online]. 2018, [cit. 2018-05-14]. Dostupné z: https://docs.google.com/forms/d/1Hkdd_EYykV1fWEZX96Ls4dF4poaJPlXMJrAeStzu1HM/viewanalytics

[15] Cígler, M.: Příběh iDokladu [online]. 2013, [cit. 2018-04-15]. Dostupné z: https://www.idoklad.cz/blog/pribeh-idokladu

[16] O nás [online]. 2017, [cit. 2018-04-15]. Dostupné z: https://www.idoklad.cz/o-nas

[17] Porovnání tarifů iDokladu [online]. 2016, [cit. 2018-04-19]. Dostupné z: https://app.idoklad.cz/Billing/Wizard?step=comparison

[18] Otevřené bankovnictví [online]. 2018, [cit. 2018-05-16]. Dostupné z: https://www.equabank.cz/pece-a-podpora/otevrene-bankovnictvi

[19] The ArrayAccess interface [online]. [cit. 2018-05-14]. Dostupné z: http://php.net/manual/en/class.arrayaccess.php

[20] PHP Data Objects [online]. [cit. 2018-06-19]. Dostupné z: http://php.net/manual/en/book.pdo.php

[21] PDO Drivers [online]. [cit. 2018-06-19]. Dostupné z: http://php.net/manual/en/pdo.drivers.php

[22] Marcus, D.: PHP PDO Prepared Statements Tutorial to Prevent SQL Injection [online]. 2017, [cit. 2018-06-19]. Dostupné z: https://websitebeaver.com/php-pdo-prepared-statements-to-prevent-sql-injection

[23] What is REST? [online]. [cit. 2018-06-24]. Dostupné z: https://www.codecademy.com/articles/what-is-rest

[24] HATEOAS Driven REST APIs [online]. [cit. 2018-06-24]. Dostupné z: https://restfulapi.net/hateoas/

[25] Administrativní registr ekonomických subjektů [online]. 2013, [cit. 2018-06-23]. Dostupné z: http://wwwinfo.mfcr.cz/ares/ares.html.cz

[26] Česká bankovní asociace: Standard ČBA – Formát pro sdílení platebních údajů v rámci tuzemského platebního styku v CZK prostřednictvím QR kódů [online]. 2015, [cit. 2018-06-24]. Dostupné z: https://www.czech-ba.cz/sites/default/files/standard_26_qr_externi_final_srpen_2015.pdf

[27] PHP QR Code [online]. [cit. 2018-06-24]. Dostupné z: http://phpqrcode.sourceforge.net/

[28] Dompdf [online]. [cit. 2018-06-24]. Dostupné z: https://github.com/dompdf/dompdf

[29] Back, I.: mPDF [online]. [cit. 2018-06-24]. Dostupné z: https://github.com/mpdf/mpdf

[30] Banka [online]. [cit. 2018-06-24]. Dostupné z: https://www.idoklad.cz/pub/Help/index.html?fin_banka.html

[31] REST Resource Naming Guide [online]. [cit. 2018-06-24]. Dostupné z: https://restfulapi.net/resource-naming/

[32] HTTP Status Codes [online]. [cit. 2018-06-25]. Dostupné z: https://restfulapi.net/http-status-codes/

[33] SimpleXML [online]. [cit. 2018-06-25]. Dostupné z: http://php.net/manual/en/book.simplexml.php

[34] ARES - získání dat pomocí php [online]. [cit. 2018-06-25]. Dostupné z: http://www.garth.cz/ostatni/ares-ziskani-dat-pomoci-php/

[35] Kalkulátor IBAN - Česká republika [online]. [cit. 2018-06-26]. Dostupné z: https://www.cnb.cz/cs/platebni_styk/iban/iban.html

[36] Holub, V.: Removing accents from a UTF-8 string in PHP [online]. [cit. 2018-06-26]. Dostupné z: http://d3s.mff.cuni.cz/~holub/sw/phpaccents/

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

OSVČ Osoba samostatně výdělečně činná

PDF Portable document format

IČO Identifikační číslo osoby

DIČ Daňové identifikační číslo

ČNB Česká národní banka

ČBA Česká bankovní asociace

ARES Administrativní registr ekonomických subjektů

DPH Daň z přidané hodnoty

API Application programing interface

SMTP Simple mail transfer protocol

EET Elektronická evidence tržeb

FAQ Frequently asked questions

CSS Cascade style sheet

CORS Cross-origin resource sharing

HTTP Hypertext Transfer Protocol

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `install.txt` ..... stručný návod k instalaci
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF