

**STŘEDNÍ PRŮMYSLOVÁ ŠKOLA BRNO, PURKYŇOVA,
PŘÍSPĚVKOVÁ ORGANIZACE**



DAŇOVÁ EVIDENCE

JAN ŠVÁBÍK

V4D

**PROFILOVÁ ČÁST MATURITNÍ ZKOUŠKY
MATURITNÍ PRÁCE**

BRNO 2018

ZADÁNÍ MATURITNÍ PRÁCE

obhajované před zkušební komisí

Školní rok: 2017/2018

Předmět: Soubor odborných předmětů (zaměření na informační technologie)

Studijní obor: Informační technologie (18-20-M/01)

ŠVP: Informační technologie

Žák: Jan Švábík

Třída: V4D

Vedoucí práce: RNDr. Lenka Hrušková

Termín zadání MP: 1. prosince 2017

Termín odevzdání MP: 17. dubna 2018

Číslo zadání, verze zadání a název práce:

PROG1M: Daňová evidence

Zadání

Vytvořte webové stránky pro daňovou evidenci s aktuálními informacemi (ze strany podnikatele i ze strany státu). Aplikace bude podnikatelům po registraci a přihlášení umožňovat vedení vlastní daňové evidence, sama pak bude podle uživatelem zadaných údajů počítat daň a výši sociálního a zdravotního pojištění apod.). Aplikace bude umožňovat kromě ručního vedení také vzdálený přístup (API), uživatel by tak případně mohl účetnictví propojit s aplikací třetí strany, která by mohla automaticky vytvářet příjmy či výdaje prostřednictvím odeslání jednoduchého JSON řetězce. Kromě toho bude aplikace disponovat možností spojit se s účtem vedeným u podnikateli a firmami banky.

Výstupem práce bude


- Webová aplikace s danou funkcí
- Postup instalace SW
- Uživatelská příručka
- Dokumentace bude obsahovat záznam z testování aplikace a výsledky jednotkových testů
- Dokumentace a zdrojové kódy v elektronické podobě
- Vývoj aplikace bude možné sledovat pomocí verzovacího systému

Doporučené prostředky pro řešení, technické požadavky


- Node.js – prostředí pro zpracovávání JavaScriptu na straně serveru
- MongoDB – databázový systém typu NoSQL
- npm – Node package manager – správce balíčků/knihoven

Součástí zadání je Příloha zadání maturitní práce.


V Brně dne 1. 12. 2017



Podpis žáka



Podpis vedoucího práce



Podpis předsedy PK

Prohlášení

Prohlašuji, že jsem maturitní práci na téma „Daňová evidence“ vypracoval samostatně a použil jen zdroje uvedené v seznamu literatury.

Prohlašuji, že беру на vědomí, že zpráva o řešení maturitní práce a základní dokumentace k aplikaci bude uložena v elektronické podobě na intranetu Střední průmyslové školy Brno, Purkyňova, příspěvkové organizace.

Prohlašuji, že беру на vědomí, že bude má maturitní práce včetně zdrojových kódů uložena v knihovně SPŠ Brno, Purkyňova, p. o., dostupná k prezenčnímu nahlédnutí. Škola zajistí, že nebude pro nikoho možné pořizovat kopie jakékoliv části práce.

Prohlašuji, že беру на vědomí, že SPŠ Brno, Purkyňova, p. o., má právo celou moji práci použít k výukovým účelům a po mém souhlasu nevýdělečně moji práci užít ke své vnitřní potřebě.

Prohlašuji, že беру на vědomí, že pokud je součástí mojí práce jakýkoliv softwarový produkt, považují se za součást práce i zdrojové kódy, které jsou předmětem maturitní práce, případně soubory, ze kterých se práce skládá. Součástí práce není cizí ani vlastní software, který je pouze využíván za přesně definovaných podmínek, a není podstatou maturitní práce.

Jan Švábík
Popeláková 2297/9
628 00 Brno

V Brně dne 16. 4. 2018

.....
podpis

Poděkování

Tímto bych rád poděkoval své vedoucí maturitní práce RNDr. Lence Hruškové za veškerou ochotu, trpělivost, velice užitečnou metodickou pomoc a veškeré cenné odborné rady při zpracovávání maturitní práce.

Dále mé díky patří také Štěpánu Krejčířovi a Jakubu Medovi, kteří se ochotně a dobrovolně uvolili veškerou funkčnost mé práce otestovat, a bez kterých by má webová aplikace vypadala o poznání hůře.

Anotace

Práce „Daňová evidence“ popisuje postup při vývoji webové aplikace mj. pro vedení příjmů a výdajů a automatický výpočet sociálního a zdravotního pojištění a daně z příjmů. Aplikace je určena výhradně českým podnikatelům. Těm umožní také kromě ručního vedení evidence vzdálený přístup prostřednictvím API, možnost propojení s bankovním účtem u vybraných českých bank pro možnost kontrolování finančních toků a snazšího přidávání nových záznamů do evidence, které jsou založené na tocích na účtu. V budoucnu bude podporováno také například vedení databáze zaměstnanců, jejich mezd, správa faktur apod.

Klíčová slova

daňová evidence, pojištění, daň, výpočet, podnikatel, vzdálený přístup, API, propojení s bankovním účtem, Node.js, MongoDB, MVC

Annotation

This graduation work (named „Tax records“) describes the development process of the web application which allows users to store their financial records (incomes, expenditures) in database. Social and health insurance and taxes are automatically calculated based on user's data. It also allows users to work with their data via API which is usable for remote access or link third-party apps. It is possible to link bank accounts (of some Czech banks) to the app to control money flow and easier adding new records (based on the bank account flows). In the future, the app will support for example employee database management, employee salary management and invoice management.

Keywords

tax records, insurance, tax, calculation, entrepreneur, remote access, API, connecting bank account, Node.js, MongoDB, MVC

Obsah

Prohlášení.....	3
Poděkování	4
Anotace	5
Annotation	5
Obsah	6
Seznam obrázků	8
Seznam použitých zkratk a pojmů	9
1 Teoretický úvod	10
1.1 Uvedení do problematiky	10
1.2 Důvod výběru tohoto tématu	10
1.3 Cíl práce	10
1.4 Popis možných řešení	11
1.5 Zvolené řešení	14
2 Předběžný návrh postupu a rozvržení práce	15
3 Registrace domény, nastavení DNS	16
4 Příprava serverového prostředí	17
5 Příprava aplikace a jejích základních součástí	18
5.1 Připojení aplikace k serverovým částem a její spuštění	18
5.2 Designová stránka webové aplikace	20
6 Registrace a přihlašování uživatelů	21
6.1 Databázový model uživatele	21
6.2 Registrace	21
6.3 Přihlášení	22
6.4 Zapomenuté heslo	23
7 Daňová evidence	25
7.1 Přehled	25
7.2 Evidence – finanční záznamy	25

7.3	Přepnutí evidence	26
8	Uzávěrky – výpočet daně a pojištění	28
8.1	Předběžná uzávěrka.....	28
8.2	Finální uzávěrka	29
8.3	Automatická kontrola chybějící uzávěrky	30
8.4	Vracené hodnoty po výpočtu daně a pojištění.....	30
9	Propojení s bankovním účtem	31
9.1	Podpora bank	31
9.2	Propojení s bankovním účtem	31
9.3	Načítání transakcí ze serveru banky	32
9.4	Vytvoření finančního záznamu dle transakce.....	33
10	API.....	34
10.1	Povolení API	34
10.2	Vygenerování tokenu a autorizačního klíče	34
10.3	Komunikace pomocí API.....	35
11	Testování aplikace.....	37
11.1	Registrace prokazatelně neexistujícího e-mailu	37
11.2	Přijetí řetězce jako částky	37
11.3	Neinformování o neúspěšnosti vytvoření záznamu	37
	Závěr	38
	Seznam použitých zdrojů a literatury	39
	Seznam příloh.....	41

Seznam obrázků

Obrázek 1 – aplikace po přihlášení uživatele.....	23
Obrázek 2 – e-mail s aktivačním odkazem po příchodu do schránky	24
Obrázek 3 – stránka pro přidání nového záznamu	26
Obrázek 4 – seznam daňovým poradcem vedených podnikatelů.....	27
Obrázek 5 – dokončená finální uzávěrka s vypočtenými částkami.....	29
Obrázek 6 – informace o úspěšném propojení s bankovním účtem.....	32
Obrázek 7 – načtené transakce v zadaném období	33
Obrázek 8 – stránka s nastavením API.....	35

Seznam použitých zkratek a pojmů

AJAX	Asynchronous JavaScript and XML – technika umožňující odeslat nějaká data na server, ze kterého jsou následně vrácena jiná data (odpověď), která je možné zpracovat do stránky, aniž by bylo nutné takovou stránku znovu celou načítat (refreshovat)
API	application programming interface – rozhraní, které umožňuje vzdálené připojení či propojení – např. z aplikací třetích stran, které pak mohou přistupovat k datům prostřednictvím aplikace
CORS	cross-origin resource sharing – mechanismus, díky kterému je možné sdílet zdroje webových stránek (např. souborů nebo jiných dat) mezi různými doménami, existuje pouze u zabezpečeného připojení a je třeba jej povolit na dotazovaném serveru
CRUD	create, read, update, delete – čtyři základní operace prováděné nad daty v úložišti
GUI	graphical user interface (grafické uživatelské rozhraní) – rozhraní umožňující ovládat počítač nebo aplikaci prostřednictvím interaktivních prvků jako jsou tlačítka, posuvníky, menu aj.
JS	JavaScript – interpretovaný objektově orientovaný synchronní skriptovací jazyk vykonávaný (v případě frontendu webových stránek) na straně klienta, dnes je součástí všech prohlížečů a je tak prakticky multiplatformní
MFČR	Ministerstvo financí České republiky
MVC	model-view-controller – způsob rozdělení aplikace na datové modely, GUI a řídicí logiku aplikace
npm	Node package manager – správce balíčků nainstalovaných na serveru, je nainstalován v rámci instalaci Node.js enginu
RŽP	Registr živnostenského podnikání (živnostenský rejstřík)
SVČ	samostatně výdělečná činnost (podnikání)

1 Teoretický úvod

1.1 Uvedení do problematiky

Již několik let postupně v České republice narůstá počet aktivních podnikatelů – v roce 2017 počet činil 3 660 644 [1]. Náš stát byl v roce 2014 dokonce podílem podnikatelů ve státě v rámci Evropské unie pátý v řadě [2, s. 6]. Každý podnikatel s ročním obratem do 1 000 000 korun je povinen vést daňovou evidenci pro daný kalendářní rok obsahující údaje o veškerých příjmech, výdajích, pohledávkách a dluzích. Někteří z těchto podnikatelů si daňovou evidenci vedou v tabulkovém procesoru MS Excel ručně (byť daně a pojištění mohou být počítány automaticky), přitom by její vedení šlo ještě více usnadnit. Zároveň si musí sami mj. zjistit výši tzv. minimálního vyměřovacího základu pro daný rok, z něhož jsou teprve poté pojištění počítána. Bude-li se však o získání potřebných dat ze serverů nebo z webů státní správy starat třetí strana a aplikace takové třetí strany „Daňová evidence“ tak bude mít k datům neustále přístup, tato práce navíc pro podnikatele zmizí.

Vzácností již nejsou ani tzv. API bankovníctví, kdy je možné svůj bankovní účet propojit s aplikacemi třetích stran, prostřednictvím kterých pak k finančním tokům na účtu lze přistupovat či dokonce platby zadávat.

1.2 Důvod výběru tohoto tématu

Zásadně mé rozhodnutí zvolit si vývoj daňové evidence pro podnikatele ovlivnil fakt, že podnikatelem jsem i já sám. Věřím, že přehledná webová aplikace právě tohoto charakteru přijde vhod jak mně, tak i dalším osobám samostatně výdělečně činným.

Kromě toho mohu takto do své práce vložit mnohem více sebe samého, než kdybych vyvíjel něco, co bych sám nikdy nevyužil, co by se mi nelíbilo a co by mne možná ani nebavilo. Když vím, co má cílová skupina očekávat, mohu jí to snáze (a kvalitnější) nabídnout.

1.3 Cíl práce

Cílem mé práce je vyvinout webovou aplikaci umožňující daňovou evidenci vést, dostupnou online a použitelnou po vytvoření vlastního účtu a přihlášení.

Tato webová aplikace musí ve výsledku disponovat několika vlastnostmi – mít intuitivní GUI uzpůsobené i pro mobilní zařízení, pracovat rychle, musí obsahovat funkční API umožňující načítání dat pro potřeby jiných aplikací a vytváření nových záznamů v databázi prostřednictvím odeslání JSON řetězce na určitou URL adresu

a musí být řádně zabezpečena – zejména proti možným brute force útokům na přihlašovací formulář či proti využití API neoprávněnou osobou.

1.4 Popis možných řešení

1.4.1 Webová aplikace, PHP a MySQL

Při uvažování nad možnými řešeními problematiky vyvstane jako první využití nejpoužívanějších technologií, jako je PHP hosting a využití databáze MySQL.

Jde prakticky o nejlevnější řešení, obě zmíněné technologie jsou však v porovnání s dalším možným řešením zejména v rychlosti své činnosti znatelně pomalejší [3].

V PHP se kód typicky vykonává postupně shora dolů a každý příkaz čeká, než skončí ten předchozí, což svým způsobem celou činnost aplikace zpomaluje.

Spojování tabulek v MySQL databázích na rychlosti také nepřidává, navíc se někdy MySQL databáze chovají poměrně nepředvídatelně – ignorují nastavení řazení indexů [4], ve výchozím nastavení v jistých případech neohlásí chybu v momentě, kdy je do nějakého sloupce ukládán špatný datový typ (ve výchozím nastavení totiž nejsou zobrazovány chyby typu *warning*) [4] či ve chvíli, kdy se snažíme do buňky VARCHAR pro čtyři znaky vtěsnat znaků 5, nám oznámí, že nenastala žádná chyba, byť budou uloženy jen první čtyři znaky (zatímco např. v PostgreSQL by nám chyba zprávou `ERROR: value too long for type character varying(4)` oznámila byla) [5].

1.4.2 Webová aplikace, Node.js a MongoDB

Tyto modernější technologie, které se v současnosti stále více dostávají do popředí a nahrazují ty staré, umožní aplikaci plynulejší běh a vývojáři ulehčí spoustu práce.

Součástí Node.js je zároveň *npm*. Node package manager nám umožňuje správu balíčků – můžeme si instalovat potřebné balíčky, jako je např. validátor e-mailové adresy (*validator*), nebo knihovna pro práci s datem (*moment*), což nám ušetří spoustu času, jenž by byl pro vývoj potřebný. Balíčky bývají perfektně odladěné, a navíc je jich opravdu mnoho – v době psaní této maturitní práce čítal jejich počet 475 000 kousků [6]. Není tedy problém najít balíček umožňující komplexní práci s obrázky, e-maily nebo jakýmkoliv jiný, který nám pomůže vyřešit problém, který v daný čas vyřešit potřebujeme.

Node.js samotný na rozdíl od PHP může zpracovávat několik částí kódu zároveň – JavaScript je totiž *jazyk synchronní*. Nemusíme tak třeba vůbec čekat na obdržení vyžádaných dat od databáze – zároveň při čekání na hodnoty z databáze může být vykonávána jiná část kódu. Tímto způsobem může také být spuštěno několik dotazů

na databázi paralelně a zpracování takových dat může být vykonáno až po získání poslední očekávané odezvy. Rychlost aplikace bude podstatně vyšší, než kdybychom při každém dotazu museli čekat, jako tomu je u *asynchronních jazyků*, mezi které patří mj. již zmíněné PHP.

MongoDB, databáze typu NoSQL, která je tvořena nikoliv tabulkami, nýbrž tzv. kolekcemi (ty zase neobsahují řádky, ale dokumenty), pracuje rychle sama o sobě díky tomu, jak je koncipována. Dokument v databázi je prakticky řetězec podobný JSON. MongoDB jej nazývá BSON [7]. MongoDB je navržena pro takový způsob práce, kdy si z ní načteme jediným příkazem veškerá potřebná data a ta zpracujeme (např. matematické operace a výpočty s daty) až v Nodu.

Uvažme model kolekce Knih zvanou „Book“. U každé knihy v knihkupectví bude zapotřebí vést název, autora, informace o vydání a nakladatelství, typ vazby, počet stran a cenu za kus. Dokument v kolekci Book by pak vypadal asi následovně:

```
{
  _id: ObjectId('5a1bd6e19e5bc650831854cf'),
  name: 'Andělé a démoni',
  author: {
    name: 'Dan',
    surname: 'Brown',
  },
  published: {
    year: '2006',
    publishingHouse: 'Argo',
  },
  bookbinding: 'paperback',
  pages: 464,
  price: {
    value: 311,
    currency: 'CZK',
  },
}
```

Další kniha (další dokument v databázi) od Dana Browna by opět sám obsahoval tuto informaci (jméno a příjmení autora). Právě tento styl uložení informací (uložení všeho potřebného v rámci každého jednotlivého dokumentu) nám zmíněné načtení všeho potřebného jediným příkazem umožňuje – není třeba propojovat různé tabulky a tím server zbytečně zatěžovat a vše tak probíhá velice rychle.

Úprava/oprava dat by proběhla tak, že by byly vybrány dokumenty s příslušnými znaky (kupříkladu {author.name: 'Daniel', author.surname: 'Brown'}) a hromadně by se jim nastavila nová hodnota (např. úprava jména z „Daniel“ na „Dan“). Zápis trvá tedy o pár milisekund déle, ve výsledku nicméně umožňuje již vícekrát omílaný plynulejší běh zbytku webové aplikace.

Řešení je dražší, poněvadž je třeba mít k dispozici vlastní server nebo jej mít pronajatý (běžné hostingové služby disponují většinou pouze PHP interpreterem). Pro potřebu vedení evidence pár desítek podnikatelů by roční náklady za pronájem kvalitního serveru nečinily více než 2 000 Kč za kalendářní rok, nezanedbatelné by tedy nebyly.

Další nevýhodou pro poskytovatele takové online služby je fakt, že musí zajistit ideálně stoprocentní dostupnost serverů a potřeba výkonu se zvyšujícím se počtem uživatelů úměrně poroste – bude nutné pravidelně nakupovat nové servery nebo jiný hardware pro zajištění stability aplikace.

Na druhou stranu – cena takové online služby by zřejmě byla stanovena formou pravidelného předplatného. Uživatelé by tak poskytovateli zajišťovali pravidelné příjmy, byť by jejich výše byla ovlivněna marketingovou politikou poskytovatele a počtem uživatelů využívajících takovou službu.

1.4.3 Desktopová aplikace kontaktující CRUD server

Jinou možností by bylo vyvinout desktopovou aplikaci, jež by při požadavku kontaktovala vzdálený CRUD server, který by s daty příslušný požadavek vykonal a aplikaci vrátil jen elementární informaci o výsledku, v případě načítání dat pak JSON řetězec, který by byl v aplikaci interpretován dle potřeby.

Toto řešení přináší nesmírnou výhodu pro server, který je zatěžován pouze tím, že zpracovává textové požadavky – nemusí kromě toho generovat vzhled stránky prakticky při každém načtení stránky, ani zbytečně nezatěžuje databázi, která na něm běží. Server tak s daty ve výsledku pracuje mnohem rychleji.

Nevýhodou je např. nutnost do zařízení aplikaci nainstalovat, a tedy i nemožnost využít jakékoliv zařízení připojené k internetu disponující webovým prohlížečem.

1.4.4 Alternativní možnost – licence a instalace na server podnikatele

Alternativní možností řešení problematiky by byla namísto nabízení aplikace jako webové služby možnost nákupu licence k aplikaci a obdržení kopie takové aplikace. Podnikatel by si následně sám musel zajistit server nebo pronájem serveru, na němž by daňová evidence následně běžela. Kromě toho by si každý takový podnikatel musel sám obstarat zabezpečení serveru před různými typy útoků a někoho, kdo by aplikaci na serveru zprovoznil a udržoval aktuální.

Někteří podnikatelé by mohli tento způsob vedení daňové evidence oproti formě webové služby upřednostnit z toho důvodu, že jsou data uložena na jejich vlastním serveru a že je tak mají plně pod svojí kontrolou.

1.5 Zvolené řešení

Rozhodl jsem problematiku řešit způsobem popsáním v kapitole 1.4.2 – půjde tedy o webovou aplikaci postavenou na Node.js a databázi MongoDB. Podnikatelům bude poskytována jako webová služba. Veškeré výhody a nevýhody (tedy důvod rozhodnutí pro toto řešení) jsem popsal přímo ve zmíněné kapitole společně se základním vysvětlením principů využitých technologií.

2 Předběžný návrh postupu a rozvržení práce

Uvedené termíny jsou myšleny v roce 2018. Nejsou uvedeny činnosti, které budou vykonávány průběžně, jako například rozvrhování části pro podnikatele.

Úkol	Ověřitelné výstupy	Termín	Poznámka
registrace domény, nastavení DNS, příprava serveru, https, základní nastavení aplikace	přístupná doména přes zabezpečený protokol https, funkční zobrazování statických souborů (obrázky, styly apod.)	1. 1.	
vytvoření úvodní stránky	vzhledná úvodní stránka	9. 1.	
registrace a přihlašování	funkční registrace a přihlašování	10. 1.	model User
zadání a uložení údajů o podnikateli, změna hesla uživatele	data jsou správně ukládána a příslušně se to projevuje	14. 1.	
přidávání, odebírání a úprava záznamů	podnikatel může již tvořit, mazat, editovat zanést finanční záznamy	21. 1.	model FinancialRecord
výpočet daní a pojištění, uzávěrky	přesné a správné výpočty	28. 1.	model YearData
vývoj API (gen. tokenu, přijímání a zpracovávání požadavků)	funkční možnost využití API pro vytváření záznamů či jejich načítání nebo mazání z databáze	2. 2.	model APIData; dokumentace API – /e/api/
propojení s bank. účtem	jsou načítána data o účtu nebo finanční toky	10. 2.	model Conn.BankAcc.
přizpůsobení daňovým poradcům	účet typu <i>daňový poradce</i> má možnost vytvářet pod svým účtem podnikatele	14. 2.	
zdokonalování UX	drobné úpravy pro vylepšení user experience, snazší ovládání, porovnání s předchozími roky, aj.	28. 2.	
testování, kontrola kódu	nalezení drobných chyb → ladění	březen	ladění, opravy
sepsání dokumentace a uživatelské příručky	zdokumentovaná aplikace, popis postupů, dostupná příručka pro uživatele na adrese https://noilio.cz/prirucka	březen	
odevzdání maturitní práce a souvisejících dokumentů	https://noilio.cz/ , dokumentace, už. příručka, postup instalace SW	do 17. 4.	

3 Registrace domény, nastavení DNS

Aby mohla webová služba být přístupná (aniž by musela být používána pro přístup IP adresa serveru), bylo třeba zaregistrovat doménu.

Protože bude webová aplikace v budoucnu zřejmě reálně využívána, rozhodl jsem se zaregistrovat doménu druhého řádu, a sice **noilio.cz**.

Národní doménu druhého řádu .cz jsem zvolil bez většího rozmýšlení – služba je určena českým podnikajícím subjektům a není tak nutné připravovat se na expanzi do dalších států a registrovat tak doménu generickou.

Po registraci jsem v DNS nastavil potřebné údaje, aby služba fungovala přesně jak má – pro dostupnost webové aplikace je úhlavní A záznam, na který bude přeloženo doménové jméno při zadání do prohlížeče. Níže však uvádím celý stav DNS této domény v momentě psaní této práce.

	1800	A	89.211.215.52
	1800	CAA	0 issue "letsencrypt.org"
	1800	MX	1 noilio.cz
	1800	TXT	v=spf1 mx -all
*	1800	A	89.221.215.52
*	1800	MX	1 noilio.cz

4 Příprava serverového prostředí

Pro běh aplikace je potřeba mít na serveru nainstalovaný operační systém (ideálně Linux Debian 8 Jessie), řádně server zabezpečit, a nainstalovat webový server nginx, Node.js, MongoDB a Redis server¹.

Zároveň je nutné nakonfigurovat doménu, přes kterou bude daňová evidence dostupná, na straně webového serveru, vystavit pro ni https certifikát od uznávané certifikační autority, používání https řádně nastavit v nastavení domény webového serveru a zajistit v témže souboru (nginx konfigurační soubor domény) předávání požadavků na vybraný port, na kterém bude Node poslouchat, pomocí proxy_pass (běžně se pro tento účel u Nodu používá port 3000).

Z hlediska MongoDB databáze a Redis serveru je třeba oba dva nakonfigurovat a zajistit jejich běh – proces databáze MongoDB (mongod) na portu 27017 a proces redis-server na portu 6379.

¹ **Redis** – server používaný pro ukládání session, kdyby nebyly session uloženy tímto způsobem, ale bylo by použito výchozí nastavení Nodu (ukládání sessions do operační paměti serveru), data by se při případném ukončení běhu Nodu nenávratně ztratila, což by mělo za následek mj. odhlášení všech uživatelů)

5 Příprava aplikace a jejích základních součástí

Aby bylo možné programovat jednotlivé části aplikace, bylo třeba pro takové části vytvořit příhodné podmínky. Po připravení následujících základních kamenů bude již možné programovat jednotlivé části webové aplikace.

5.1 Připojení aplikace k serverovým částem a její spuštění

Aplikace data ukládá do dvou externích (mimo Node/serverových) systémů – data s nimiž pracuje daňová evidence samotná do databáze MongoDB a session data uživatelů do Redis serveru. Obě služby jsou na serveru již připraveny a běží.

5.1.1 Redis server

Propojení s Redis serverem jsem provedl přímo v souboru *app.js* načtením balíčků potřebných pro práci s ním (*redis*, *connect-redis*) a zadáním potřebných dat, jako je adresa serveru a port nebo název cookie, která se ukládá u přihlášeného uživatele a doba jejího trvání.

5.1.2 MongoDB

K databázi se aplikace připojuje v momentě načtení prvního databázového modelu (prakticky tedy při spuštění *app.js* na serveru). Aplikace je tak k databázi připojena a může přistupovat k datům v databázi.

Připojení jako takové je řešené v souboru */libs/db.js*, kde je z konfiguračního souboru */config.js* načtena hodnota *mongoUrl*, která obsahuje adresu serveru a port, kde má být připojení navázáno, uživatelské jméno a heslo² a název databáze.

5.1.3 Express – framework pro vývoj webových aplikací v Node.js

V *app.js* jsem dále předal frameworku potřebné informace o souborové struktuře na serveru. V momentě, kdy má být načten statický soubor (např. logo */img/logo.png*), bude Express vědět, že jej má hledat v adresáři */static/*, v případě pohledů pak, že je má načítat z adresáře */views/*.

Kromě toho bylo třeba připravit dva soubory (*/routers/evidence.js* a */routers/web.js*) určující *co se bude dít při přístupu na danou URL adresu*. V těchto *routerech* je uvedeno např. že pokud požadavek míří do části pro přihlášené (do evidence) (*/e/**), má být zkontrolováno, že uživatel opravdu přihlášený je – pokud ne, je přesměrován na

² v případě, že na jednom serveru běží více webů, je dobré každé jednotlivé webové aplikaci vytvořit v MongoDB samostatného uživatele, který bude mít přístup do své vlastní databáze

stránku pro přihlášení, která do této kontroly nespadá, byť také adresa začíná */e*. Že do této kontroly nespadá jsem nastavil prostým uvedením routu */e/login* **před** */e/**. Proč není v routech (cestách) uváděna část URL */e* je popsáno dále.

```
// login and checking that user is logged in

const loginControl = (req, res, next) => {
  if (req.session.user)
    next();
  else
    res.redirect('/e/login/');
}

const loggedIn = (req, res, next) => {
  if (req.session.user)
    res.redirect('/e/');
  else
    next();
}

router.post('/login', loggedIn, (req, res) => {
  loginController.login(req, res);
});

router.get('/login', loggedIn, (req, res) => {
  loginController.show(req, res);
});

router.all('/*', loginControl, (req, res, next) => {
  next();
});
```

Rád bych ještě popsal tři funkce Express routeru, které jsem v kódu tučně zvýraznil: `router.post`, `router.get` a `router.all`.

V momentě, kdy je odeslaný např. POST formulář (typicky přihlašovací formulář) na adresu */login*, je použita klauzule `router.post('/login', ...)`, která zavolá login controller, který se postará o samotnou požadovanou funkci – o ověření správnosti hesla a případné (ne)přihlášení uživatele. Následně je uživatel přesměrován do evidence nebo zpět na přihlašovací stránku.

Druhá možnost, `router.get('/login', ...)` je použita, pokud byl zadán požadavek GET (typicky prosté načtení nějaké stránky), v tomto případě je v controlleru *login* načten model pro přihlašovací stránku. Vygenerovaná stránka je pak poslána uživateli, kterému je zobrazena v prohlížeči.

Poslední případ `router.all` se používá, chceme-li něco provést bez ohledu na to, zda jsme využili POST nebo GET požadavek.

V routech není uváděna cesta */e*, protože router *evidence* je v *app.js* přiřazen právě všemu, co začíná adresou */e*.

```
// routers
const evidenceRouter = require('./routers/evidence');
const webRouter = require('./routers/web');

app.use('/e', evidenceRouter);
app.use('/', webRouter);
```

5.1.4 Naslouchání na daném portu – spuštění aplikace

Express musí naslouchat na portu, na nějž nginx posílá jednotlivé požadavky na server od uživatelů. Protože na serveru, který jsem pro běh webové aplikace využil, běží na Nodu další aplikace, nevyužil jsem defaultní port 3000, nýbrž další volný v pořadí, a sice 3003. Kód, který poslech na portu zajišťuje, je uvedený níže. Je-li poslech úspěšně zahájen a aplikace je tedy spuštěna, v serverové konzoli se vypíše datum a čas spuštění.

```
app.listen(3003, () => {
  console.log(moment().format('YYYY-MM-DD HH:mm:ss') + ' Listening on
port 3003 (noilio.cz)');
});
```

5.2 Designová stránka webové aplikace

Design veřejné části webu jsem nakódoval sám – jde o představení daňové evidence a jejích částí a funkcí jako online služby Noilio formou jediné stránky.

V případě samotné části správy daňové evidence (pro přihlášené podnikatele) jsem pro design použil šablonu Inspinia³ postavenou na Twitter Bootstrapu verze 3, k níž vlastním licenci typu *Multiple applications*.

Pro tuto a podobné druhy webových aplikací ji využívám pro její propracovanost jak z hlediska obsahu (nastylována je obrovská spousta věcí od formulářů, nástěnek s grafy a tabulek přes různé typy stránek jako je profil uživatele, správa kontaktů či e-mailová schránka, až po malé JavaScriptové utility jako je měřič síly hesla, PDF viewer, editor kódu, stromové zobrazení adresáře aj.), tak z hlediska použitelnosti na různých rozlišeních displeje a její rychlost.

³ <https://wrapbootstrap.com/theme/inspinia-responsive-admin-theme-WB0R5L90S>

6 Registrace a přihlašování uživatelů

6.1 Databázový model uživatele

Model uživatele jsem navrhl tak, jako všechny ostatní budoucí modely – aby byla data uspořádána přehledně. Celý model User jsem umístil do samostatné přílohy – zejména z důvodu jeho délky.

6.2 Registrace

Každý uživatel musí vyplnit e-mailovou adresu a heslo. Těmito údaji se bude ke svému účtu přihlašovat. Kromě toho je také povinné zvolit typ účtu (podnikatel nebo daňový poradce) a uvést informace o sobě samotném z hlediska podnikání (IČ, jméno, příjmení a sídlo).

6.2.1 Předvyplnění informací o podnikateli z databáze RŽP

Pro usnadnění registrace podnikatelům jsem využil API Ministerstva financí, jenž umožňuje připojit se do živnostenského rejstříku a podle zadaného IČ načíst data o podnikateli. Uživatel tak nemusí vypisovat spoustu informací ručně.

V registračním formuláři po dopsání poslední číslice IČ je možné buďto požadavek odeslat kliknutím na vedlejší tlačítko „Načíst data z RŽP“ nebo stisknutím klávesy Enter. Podle toho, zda podnikatel existuje, jsou údaje předvyplněny, nebo je zobrazena příslušná chybová zpráva.

Je třeba zmínit, že bylo nutné pomocí AJAXu kontaktovat vlastní server, který se teprve poté sám připojí k serveru MFČR, od kterého data o podnikateli získá. Připojit se přímo k serveru MFČR nebylo možné, protože na serveru ministerstvo při zabezpečeném připojení nemá povoleno CORS a kdyby bylo namísto https použito připojení nezabezpečené, na stránce s registračním formulářem by pak v prohlížeči nebyla zobrazena ikona, která uživateli oznamuje, že je spojení šifrované (např. v prohlížeči Google Chrome ikona zeleného zámku s textem „Zabezpečeno“). To by mohlo ovlivnit důvěru uživatele ve webovou službu.

Kromě toho ministerstvo v podmínkách používání uvádí, že při překročení počtu připojení v určitém časovém rozmezí může být zablokována IP adresa, ze které tyto požadavky přicházely. Stejně se může stát při zasílání stále stejných požadavků nebo při zasílání opakovaných neplatných požadavků. Proto je správnost zadaného IČ nejprve ověřena na serveru Noilio a pakliže je IČ ve správném tvaru, je uživateli zobrazen reCaptcha formulář, po jehož úspěšném překonání jsou data již bez nutnosti dalšího klikání automaticky z živnostenského rejstříku načtena. Díky těmto

krokům se nemůže žádným jednoduchým způsobem stát, že by u Ministerstva financí nastalo zablokování IP adresy serveru Noilio.

6.2.2 Výběr typu uživatele

Uživatel si přímo při registraci volí, zda plánuje službu využívat pouze pro své účely, nebo chce vést daňovou evidenci více podnikatelům (vhodném zejména pro daňové poradce). Evidence pak bude přizpůsobena přesně potřebám uživatele.

6.2.3 Hashování hesla

Pro hashování hesla uživatelů jsem použil knihovnu bcrypt. Zde hashování probíhá poněkud odlišným způsobem než u běžných hashovacích algoritmů např. v PHP (MD5, rodina algoritmů SHA aj.).

Bcrypt je založeno na šifře Blowfish [8], u které není dodnes znám efektivní způsob prolomení [9]. Jde o adaptivní šifrovací algoritmus [8], což znamená, že umožňuje uměle zvýšit počet iterací při výpočtu hashe, tím dosáhnout zpomalení získání hashe a zabránit tak možnosti využít brute-force útoky k prolomení hesla. Bcrypt zároveň automaticky generuje šestnáctibajtovou [8] *salt*, čímž zase znemožňuje pro prolomení hesla použít *rainbow tables*.

Výstupní řetězec bcrypt může vypadat například takto:

```
$2a$15$POQANjNjA97a4bNIv1xnZ.xhtK3EQzPrHvzPwL6DApe694y9/ihMi
```

Výstup je rozdělen na tři části znakem \$. První část (2a) označuje verzi algoritmu, druhá část (15) označuje počet iterací a třetí část obsahuje salt (prvních 16 znaků) a samotný hash. K ověření je potřeba celý tento výstupní řetězec.

Na serveru Noilio trvá ověření hesla (při 15 iteracích) přibližně 3,5 sekundy. Tato doba závisí na výpočetním výkonu serveru, na němž je bcrypt spuštěno.

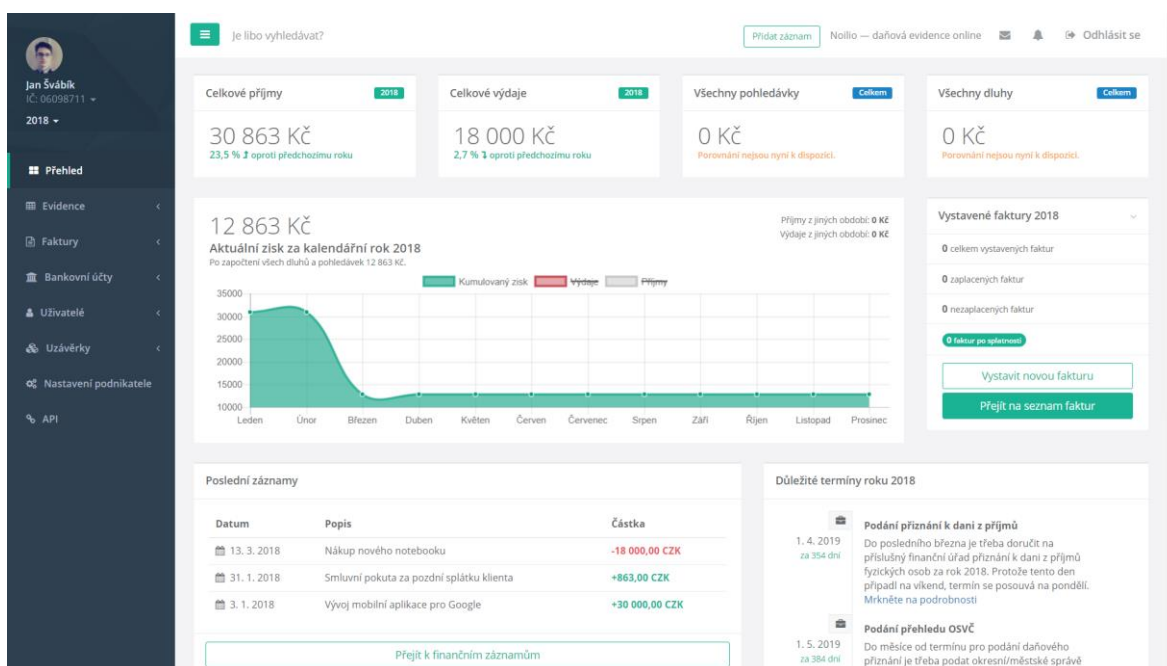
6.3 Přihlášení

Uživatelé se přihlašují svým e-mailem a heslem zadaným při registraci účtu Noilio. Pro ověření správnosti hesla je třeba řetězec z databáze předem získat a předat ho bcrypt knihovně, která si jej rozdělí na výše zmiňované části, znovu vytvoří hash ze zadaného hesla při přihlašování (se stejnou solí a stejným počtem iterací) a vykoná samotné porovnání obou hashů.

Do session `user` se po přihlášení uloží veškeré informace o uživateli a do session `openedEvidence` informace o výchozí evidenci přihlášeného uživatele. Obsah této session se může měnit v momentě, kdy má uživatel typ účtu pro daňové poradce –

při přepnutí evidence na jiného podnikatele. Díky tomu je následně možné např. zobrazit jen takové záznamy, které spadají pod aktuálně spravovaného podnikatele.

Protože je v MongoDB velikost jednoho dokumentu omezena na 16 MB [10], nejsou jednotlivé záznamy řešeny jako podobjekty dokumentu Evidence, ale tvoří novou samostatnou kolekci FinancialRecords. Důvodem je předcházení případu, kdy by podnikatel v databázi měl velké množství záznamů a jejich celková velikost by tak mohla maximální hodnoty dosáhnout (průměrná velikost záznamu je ~400 B, do dosažení limitu by databáze šla naplnit přibližně 42 000 záznamy).



Obrázek 1 – aplikace po přihlášení uživatele

6.4 Zapomenuté heslo

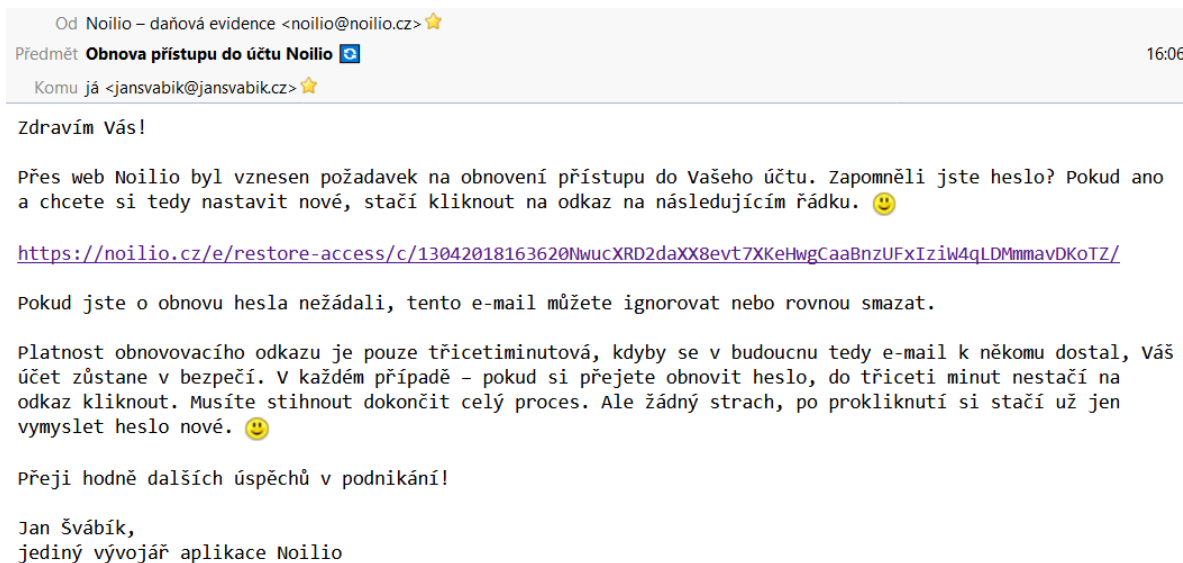
Pod přihlašovacím formulářem a tlačítkem pro vytvoření účtu se nachází odkaz na formulář pro resetování hesla. Pro reset je nutné znát svůj přihlašovací e-mail, na který bude po zadání do formuláře a ověření, zda formulář odesílá člověk, odeslán e-mail s odkazem na další formulář – pro finální nastavení hesla nového.

Při zažádání o takový odkaz se vygeneruje klíč, který se uloží do databáze k účtu, k němuž má být přístup obnoven. Platnost klíče je 30 minut od vygenerování. Klíč se skládá z data platnosti (14 znaků) a náhodného řetězce (50 znaků), jeho celková délka činí tedy 64 znaků. Tento je pak odeslán na e-mail uživatele.

Po prokliku z e-mailu se uživateli zpřístupní formulář pro zadání nového hesla. Po jeho odeslání server zkontroluje, zda již neuplynul třicetiminutový časový limit pro obnovení hesla. Pakliže ne, je vygenerován hash nového hesla a ten je následně

uložen do databáze. V případě chyby je uživatel patřičně informován. Po použití klíče pro obnovu hesla se takový klíč stává neplatným a odkaz není možné použít opětovně.

Třicetiminutový limit slouží pro zvýšení bezpečnosti uživatele – v případě, že by klíč nebyl použit a odkaz pro obnovení přístupu by se dostal do rukou třetí osoby, která by jej chtěla zneužít, v takové chvíli klíč již pravděpodobně dávno pozbyl platnosti a účet uživatele tak zůstává nadále chráněný dosavadním heslem.



Obrázek 2 – e-mail s aktivačním odkazem po příchodu do schránky

7 Daňová evidence

7.1 Přehled

Místo, kam je uživatel přesměrován po přihlášení. To nejdůležitější má pak na dosah žádného nebo jednoho kliku. Před renderováním stránky jsou z databáze načteny veškeré záznamy z daného období. Ty jsou následně roztrženy a vzájemně sečteny podle kategorie, do níž spadají – příjem, výdaj atd. Díky tomu pak uživatel ví o přesné výši peněz v každé z těchto kategorií.

Přehled zpestřuje graf zobrazující výši příjmů, výdajů a tzv. kumulovaného zisku v jednotlivých měsících společně se seznamem důležitých termínů pro dané období s vypočítaným zbývajícím počtem dní. Podnikatel tedy nemusí nikde shánět informaci, do kdy musí podat daňové přiznání nebo Přehled o příjmech a výdajích.

Pokud uživatel vstoupil do přehledu ze stránky pro přihlášení, je přivítán krátkou zprávou v pravém horním rohu obrazovky.

```
var welcome = false;
  if (typeof req.headers.referer !== 'undefined')
    if (url.parse(req.headers.referer).pathname == '/e/login/')
      welcome = true;
```

7.2 Evidence – finanční záznamy

Jde o nejdůležitější část celé aplikace – právě zde je vedena daňová evidence jako taková. Záznamy lze přidávat, procházet, editovat a archivovat (a poté také mazat).

Model samostatného záznamu (FinancialRecord) je možné si prohlédnout v příloze.

Po načtení záznamů je pro každý vygenerován nový řádek v tabulce a jednotlivé typy záznamů jsou barevně odlišeny. Aby bylo možné v tabulce rychle vyhledávat, řadit či data případně pro vnitřní potřebu exportovat nebo tisknout, použil jsem JS plugin DataTables⁴.

Pohledávky a dluhy je možné převést jedním klikem na příjmy nebo výdaje – to je využitelné v momentě, kdy nedochází k automatickému zpracování platby faktury a záznam je tak nutné upravit ručně.

7.2.1 Přidání záznamu

Vytvoření finančního záznamu je realizováno uložením nového dokumentu do kolekce FinancialRecords. Před uložením jsou zkontrolovány zadané údaje a je

⁴ <https://datatables.net/>

zabráněno případné snaze uložit do databáze jiná než povolená data (např. místo čísla řetězec nebo jiný řetězec, než je povolený – zejména u typů platby nebo také u typu záznamu jako takového).

Obrázek 3 – stránka pro přidání nového záznamu

7.2.2 Archivované záznamy a odstranění záznamu

Tyto záznamy mají nastaven příznak `archived: true`. Kdyby si podnikatel jejich vyřazení z aktivní evidence rozmyslel, může použít ikonu zelené šipky směřující vzhůru a záznam z archivu vrátit do aktivní evidence.

Druhou možností, jak s archivovaným záznamem naložit, je jej trvale a nevratně vymazat. V takovém případě dojde k reálnému odstranění záznamu z databáze.

7.3 Přepnutí evidence

Přepnutí spravované evidence na evidenci pro jiné období prostřednictvím levého menu zajišťuje funkce `swap` v controlleru `evidence`. V menu je vygenerován seznam vedených období, kde stačí na požadované kliknout a tím se do něj přepnout.

Z hlediska programu a databáze je každé období samostatnou evidencí. Pokud období, do něhož se uživatel snaží přejít, existuje, jsou o něm veškeré informace uloženy do session (`openedEvidence`). Následně je uživatel přesměrován zpět na stránku, z níž se do jiného období přepínal, jsou mu však již předložena data z období přepnutého.

U účtu daňového poradce existuje navíc přepínání mezi podnikateli. To funguje na stejném principu jako přepínání mezi obdobími. Kromě roku se však bere v potaz navíc i IČ podnikatele, jehož evidenci chce uživatel otevřít.

Pokud uživatel vede již evidenci mnoho a potřeboval by přepnout do některé ze starších, protože se v menu zobrazí jen nejaktuálnějších 5 evidencí, musí tak učinit přes stránku se seznamem všech vedených evidencí.

S obdobným případem se potýkají také daňoví poradci, kteří pro vyšší přehlednost mohou využít stránku se seznamem všech podnikatelů, do jejichž evidence se poté mohou snadno přepnout.

Seznam podnikatelů

Seznam podnikatelů pod Vaší správou

Zobrazit 50 záznamů

Hledat:

Kopírovat CSV Excel PDF Tisk

Zobrazeno 1 až 6 z 6 záznamů

IČ	Podnikatel	Vedených období	Akce
06098711	Jan Švábík	1 období	Přejít do evidence
07050301	JUDr. Jiří Daňový, CSc.	5 období	Přejít do evidence
20208858	Jana Blahová	1 období	Přejít do evidence
33333320	Ing. Josef Klásek	1 období	Přejít do evidence
78787999	Ing. Zdeněk Martinek	2 období	Přejít do evidence
89208900	Bohuslav Černocký	3 období	Přejít do evidence

Předchozí 1 Další

Copyright © Noilio.cz

Noilio. Daňová evidence.

Obrázek 4 – seznam daňovým poradcem vedených podnikatelů

8 Uzávěrky – výpočet daně a pojištění

Výpočet výše daně z příjmů a sociálního a zdravotního pojištění je proveden tzv. uzávěrkou. Noilio rozlišuje dva typy, tzv. *předběžnou uzávěrku* a uzávěrku *finální*.

8.1 Předběžná uzávěrka

Slouží k orientačnímu výpočtu daně a pojištění – lze ji provést kdykoliv, i pokud dané období ještě neskončilo. Nejsou-li v databázi uloženy údaje ze strany státu pro takové období, jsou použity takové, které mají k danému období nejbližší.

Toho je docíleno níže uvedenou agregací. Hodnota `year` každého z dokumentů v kolekci je odečtena od hodnoty `year` aktuální otevřené evidence. Záznamy jsou následně seřazeny vzestupně dle výsledku agregace a maximálně dva jsou vráceny – taková situace může nastat v momentě, kdy jsou v dokumentu daného roku již uloženy termíny pro podání daňového přiznání či Přehledu o příjmech a výdajích, ale stále chybí informace o výších daňových zvýhodnění či vyměřovacích základech pro daný kalendářní rok (model `YearData` přiložen v samostatné příloze).

```
YearData.aggregate([
  {
    $project: {
      diff: {
        $abs: {
          $subtract: [req.session.openedEvidence.year, '$year']
        }
      },
      doc: '$$ROOT'
    }
  },
  {
    $sort: {
      diff: 1
    }
  },
  {
    $limit: 2
  }
], (err, yearData) => {
  ...
});
```

Kromě toho jsou načteny všechny finanční záznamy z daného období po jejichž sumarizaci a dle údajů zadaných uživatelem při vytváření uzávěrky (výběr hlavní/vedlejší činnosti v jednotlivých měsících, uplatňovaná daňová zvýhodnění atd.) je vypočítána samotná výše daně z příjmů a sociálního a zdravotního pojištění dle již získaných údajů ze strany státu pro daný rok, případně dle nejbližších existujících informací – v takovém případě je na přibližnost uživatel i upozorněn.

Data z předběžné uzávěrky slouží pro potřeby podnikatele znát případnou aktuální výši daní a pojištění a vypočtené informace nejsou následně nikam ukládány. Při přidávání nových záznamů se budou hodnoty měnit.

8.2 Finální uzávěrka

Uzavírá rok – je provedena kontrola, zda daný rok už skončil – pokud ano, je možné uzávěrku provést, v opačném případě je zobrazen box informující o momentální nemožnosti uzávěrku provést. Na začátku jsou stejným způsobem jako u uzávěrky předběžné načteny informace o daném období ze strany státu. V případě finální uzávěrky by již informace ze strany státu měly být v databázi připravené.

Oproti předběžné uzávěrce je zde možnost rok tzv. uzavřít. Po kontrole správnosti údajů tak uživatel může jedním klikem učinit. V tom momentě je otevřená evidenci nastaven příznak `closed: true` a do objektu `closure` jsou uloženy sečtené hodnoty příjmů, výdajů, pohledávek a dluhů, vypočtený zisk a výše daně z příjmů a pojištění a datum uzavření evidence. Tato data jsou následně používána v přehledu pro meziroční porovnání výsledků hospodaření.

V uzavřené evidenci již není možné vytvářet nové finanční záznamy ani provádět jakékoliv změny. Uživatel je před uzavřením roku proto vyzván k řádné kontrole údajů o podnikateli či finančních záznamů.

Copyright © Noilio.cz

Noilio. Daňová evidence.

Obrázek 5 – dokončená finální uzávěrka s vypočtenými částkami

8.3 Automatická kontrola chybějící uzávěrky

Protože v budoucnu bude aplikace umožňovat automatické placení daně a obou pojištění, již nyní jsem zavedl kontrolu chybějících uzávěrek. Ta se provádí vždy při generování levého menu, kdy je v poli `req.session.user.entrepreneurEvidence`, jež obsahuje údaje o všech evidencích, které pod přihlášeného uživatele spadají, u každé evidence provedena kontrola.

U každé z vedených evidencí je zkontrolováno, zda není její období již v minulosti. Pokud je a zároveň nemá evidence nastaveno `closed` na hodnotu `true`, je počet na uzávěrku čekajících evidencí zvýšen o 1.

Jsou-li na konci čekající uzávěrky, nad menu je přidán výrazný odkaz. V případě jediné čekající uzávěrky vede na stránku pro provedení finální uzávěrky daného roku, text odkazu je pak např. „Uzávěrka 2018“. Je-li chybějících uzávěrek více, text odkazu není konkrétní, nýbrž obecný – „Chybějící uzávěrky“. Odkaz pak namísto na stránku pro provedení uzávěrky vede na stránku se seznamem evidencí, které nejsou uzavřeny.

V účtu daňových poradců je text „Chybějící uzávěrky“ zobrazen vždy (bez ohledu na počet). Přehledně tak vidí seznam podnikatelů a období, u kterých uzávěrka chybí a mohou si snadno zvolit, kterým podnikatelem a kterou evidencí začnou.

8.4 Vracené hodnoty po výpočtu daně a pojištění

Aby podnikatel viděl každou položku výpočtu daně a obou pojištění, jsou z funkce, která výpočet zajišťuje, vráceny hodnoty prakticky všech kroků výpočtu. Ty si následně může uživatel po potvrzení zadaných údajů prohlédnout v tabulce pod samotnými hodnotami daně a pojištění.

9 Propojení s bankovním účtem

Propojení s bankovním účtem podnikatele slouží k usnadnění vytváření finančních záznamů v daňové evidenci. V budoucnu díky němu bude možné implementovat plně automatizované placení daně a pojištění.

9.1 Podpora bank

Pro propojení s bankovním účtem je samozřejmě zapotřebí i spolupráce ze strany bankovní instituce. Ta přístup obvykle umožňuje pomocí svého API, které mohou vývojáři pro komunikaci využít.

Protože Fio banka v soutěži Zlatá koruna v kategorii Cena podnikatelů vyhrála se svým podnikatelským účtem několik ročníků po sobě [11, 12, 13], rozhodl jsem se do Noilia implementovat nejprve právě jejich API. Protože na fakturách často vídám kód banky 2010, který Fio bance patří, předpokládám, že stále mezi nejoblíbenější banky mezi podnikateli patří. S využíváním jejich API bankovníctví jsem navíc již měl zkušenosti – je skvěle zdokumentované a jednoduché na využívání.

Byť podle nového zákona o platebním styku, jenž vstoupil v platnost v lednu 2018, musejí možnost externího přístupu k účtu implementovat všechny banky [14], zatím tomu tak není. Ke konci října 2017 byly v Česku navíc pouze čtyři banky, které externí přístup k účtům klientů umožňovaly. Šlo o Českou spořitelnu, Fio banku, CREDITAS a Air Bank. [15, 16]

Podporu dalších bank budu do webové aplikace přidávat postupem času – podle situace a požadavků případných uživatelů aplikace.

9.2 Propojení s bankovním účtem

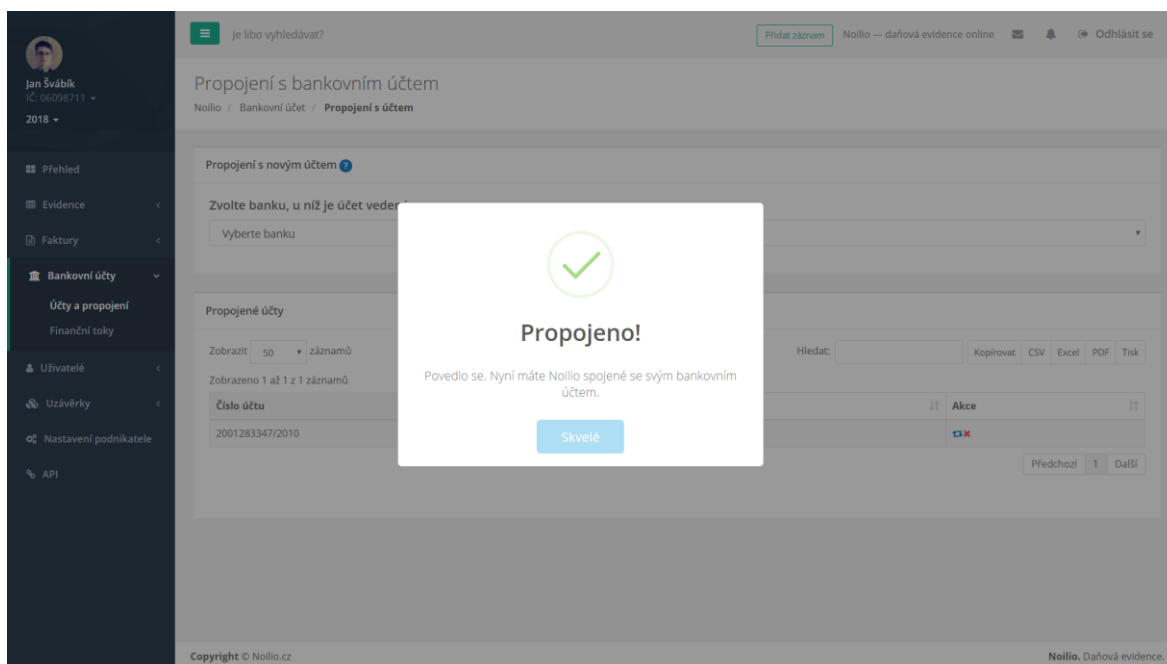
Uživatel na stránce pro propojení se svým bankovním účtem nejprve zvolí svoji banku (v době psaní této maturitní práce je jedinou možností Fio banka). Na základě vybrané banky se zobrazí další část formuláře, protože propojení může u každé banky fungovat jiným způsobem.

U Fio banky je zobrazen input pro tzv. *token*. Ten si může každý klient Fio banky vygenerovat v nastavení svého bankovního účtu. Při jeho vytváření lze také určit, zda může být použit pouze pro „čtení“, nebo povolit možností prostřednictvím daného tokenu i zadávání platebních transakcí.

Po vložení tokenu do pole a odeslání formuláře se Noilio na pozadí (AJAX voláním) spojí se serverem Fio banky – je proveden dotaz na základní informace o účtu pro

aktuální den. Podle odezvy vzdáleného serveru banky je zjištěno, zda token vůbec existuje. Pokud ano, je načteno číslo bankovního účtu a jeho měna – která musí být v CZK, protože Noilio v době psaní této maturitní práce nepodporuje práci s jinými měnami, pokud není účet veden v CZK, je o tom uživatel informován a propojení s bankovním účtem není provedeno.

V opačném případě je do kolekce bankovních účtů (ConnectedBankAccount, model si je možné prohlédnout v přílohách této práce) uložen nový dokument obsahující daný token, číslo účtu a kterému uživateli a evidenci podnikatele toto propojení náleží.



Obrázek 6 – informace o úspěšném propojení s bankovním účtem

Formulář je po propojení resetován a do tabulky se všemi propojenými účty je dané propojení ihned pomocí JavaScriptu vloženo.

9.3 Načítání transakcí ze serveru banky

Protože bude aplikace zřejmě v budoucnu podporovat více bank, bylo zapotřebí co nejdříve připravit takové rozhraní pro získávání dat od bank, aby nebylo zbytečně složité implementovat API další banky.

Koncept je takový, že každá banka bude mít vlastní funkci, které se pouze předají potřebné argumenty a funkce vrátí výsledky (transakce). Z databáze se načtou účty, které má uživatel se svým účtem propojené, a u každého z nich se zavolá funkce dle banky, které daný účet náleží. Data jsou rozparsována a následně v jednotném tvaru uložena do pole `transactions`, jež je poté odesláno na frontend k zobrazení.

Po každém využití propojeného účtu je do databáze uložen nový čas posledního použití. Například Fio banka doporučuje volat API pro daný token jednou za třicet sekund. Toto bylo třeba vzít v potaz a řádně implementovat. Na frontendu tedy není možné třicet sekund od posledního volání načíst transakce znovu.

9.4 Vytvoření finančního záznamu dle transakce

Pro vytvoření finančního záznamu podle některé z načtených transakcí se v tabulce nachází ikona zeleného dolaru, po jehož kliknutí je záznam automaticky uložen do aktuálně otevřené evidence. Tento postup jsem zvolil, abych dodržel myšlenku co nejpohodlnějšího a nejintuitivnějšího ovládání pro uživatele.

Samotný záznam je vytvořen AJAX voláním adresy, která je volána také při ručním vytváření záznamu. Jednotlivé informace jsou však v dotazu předány automaticky.

Jako popis záznamu je do databáze uložen typ pohybu (příjem nebo výdaj) a s jakým protiúčtem byla taková transakce provedena. Popis pak může vypadat například takto: „Příjem na bankovní účet 15635255/2010“.

je libo vyhledávat? Přidat záznam Nolllo — daňová evidence online

Transakce na bankovních účtech
Nolllo / Bankovní účet / Transakce na bankovních účtech

Výběr časového období
01. 09. 2017 až 14. 04. 2018 Načíst transakce

Finanční transakce uskutečněné na propojených bankovních účtech

Zobrazit 50 záznamů Hledat: Tisk

Zobrazeno 1 až 6 z 6 záznamů

Datum	Typ	Částka	Protiúčet	Symboly	Zpráva	Akce
14. 4. 2018	Příjem	8.87 CZK	85882653/0100		Zobrazit	\$
14. 4. 2018	Výdaj	-5 CZK	15991599/2010	8063544541 5544 96858	Zobrazit	\$
10. 4. 2018	Příjem	5 CZK	15991599/2010			\$
9. 11. 2017	Výdaj	-9940 CZK	15991599/2010			\$
4. 11. 2017	Příjem	9940 CZK	1805050053/0300	1101180052		\$
4. 9. 2017	Příjem	100 CZK	15991599/2010			\$

Předchozí 1 Další

Copyright © Nolllo.cz Nolllo. Daňová evidence.

Obrázek 7 – načtené transakce v zadaném období

10 API

Součástí webové aplikace je API umožňující přidávat do daňové evidence finanční záznamy prostřednictvím propojení s aplikací třetí strany. Podnikatel tak může například propojit Noilio se svým pokladním systémem a po každém prodeji může být příjem v evidenci automaticky evidován, aniž by musel kdokoliv ručně záznam vytvářet.

10.1 Povolení API

API je ve výchozím nastavení vypnuté. Po jeho povolení na stránce s nastavením API se na pozadí odešle na server AJAX požadavek, jehož obsahem je hodnota `true` nebo `false` – podle toho je API aktivováno nebo deaktivováno. Pokud požadavek proběhl v pořádku, v odpovědi se vrátí string `ok` a na stránce se zobrazí nový stav.

Informace o API pro dané evidence se ukládá do své vlastní kolekce dle modelu `APIData` (viz přílohy). Pokud záznam existoval, je nově vytvořen.

10.2 Vygenerování tokenu a autorizačního klíče

Pro umožnění komunikace třetí aplikace se serverem slouží dva řetězce – tzv. token a autorizační klíč. Token je náhodně vygenerovaná posloupnost znaků, autorizační klíč je pak složený řetězec z ID uživatele a IČ podnikatele z evidence doplněný na délku 64 znaků náhodnými znaky. ID a IČ jsou od vygenerovaných znaků odděleny lomítkem. Autorizační klíč slouží zejména k tomu, aby nenastala situace, kdy by dvě různé evidence (dvou různých podnikatelů) měly vygenerovaný stejný token (byť je taková situace krajně nepravděpodobná).

Pro generování je použita knihovna `randomstring`.

```
let generatedToken = randomstring.generate(64);

let securityString = req.session.user._id +
  parseInt(req.session.openedEvidence.companyID, 16) + '/';

let generatedAuthKey = securityString + randomstring.generate(64 -
  securityString.length);
```

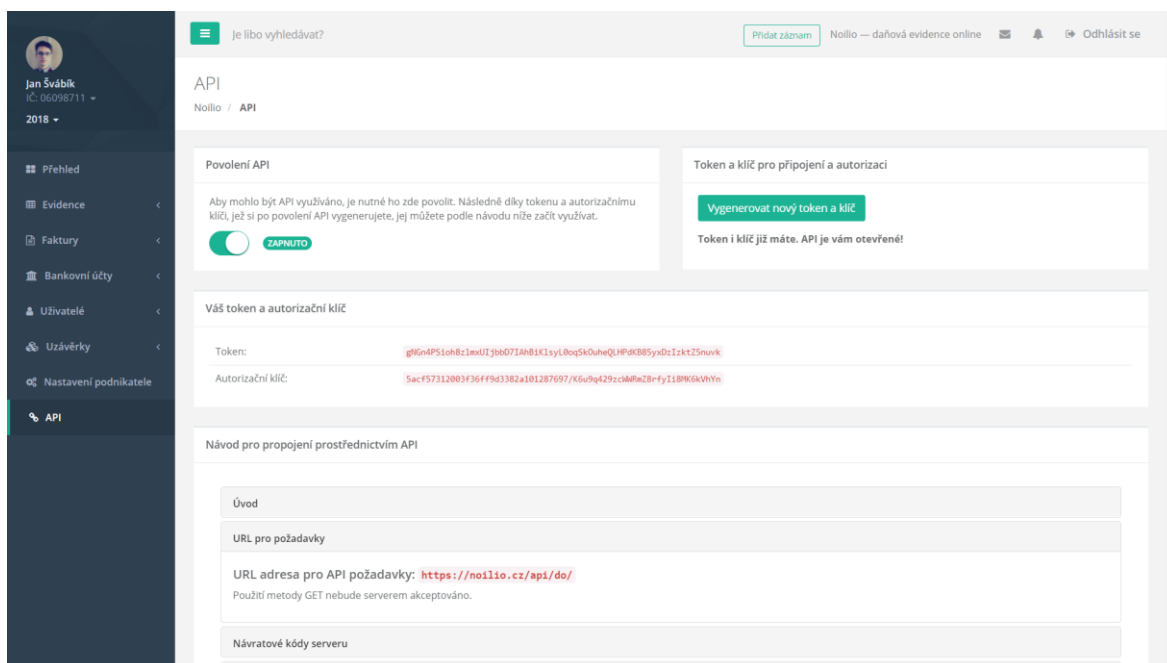
Pro možnost využívání API si uživatel musí token a klíč vygenerovat kliknutím na tlačítko vedle přepínače pro aktivaci rozhraní.

Ukázka vygenerovaného přístupového tokenu:

YI62iavtJz1tK7psQlOjl6FY3DCnCJzepdB8z6TXxlrR85Lhh83wkuy562wj4114

Ukázka vygenerovaného autorizačního klíče:

5acfbb39a8bfb51bd434a802117768961/XRGvUnHsrDgm9SCK51Q9f3b7CVblq6



Obrázek 8 – stránka s nastavením API

10.3 Komunikace pomocí API

Server očekává API požadavky na adrese <https://noilio.cz/api/do/>. Komunikace probíhá přes zabezpečený protokol a zpracována je pouze, pokud je požadavek odeslán pomocí metody POST. Data musí být odeslána ve formátu JSON.

JSON objekt musí obsahovat několik povinných údajů, zbylé jsou pak pro uživatele volitelné. Mezi povinné patří samozřejmě token a autorizační klíč, prostřednictvím jejichž kombinace je vybrána příslušná evidence, se kterou se bude dále pracovat.

V tabulce níže jsou jednotlivé možnosti uvedeny. V době psaní této maturitní práce umožňuje API pouze přidávat záznamy do evidence podle aktuálního roku.

Název	Datový typ	Povinný	Popis
token	string	ano	token pro připojení
authKey	string	ano	autorizační klíč
action	string	ano	činnost pro vykonání
type	string	ano	typ záznamu (příjem, ...)
description	string	ano	popis záznamu
amount	integer	ano	částka
tax	boolean	ano	daňový/nedaňový záznam
note	string	ne	poznámka
payment	string	ne	způsob platby
documentNumber	string	ne	číslo propojeného dokum.
internalDocumentNumber	string	ne	interní ozn. propojeného dokum.

Tato tabulka je včetně příkladů a podrobnějšího popisu přímo na stránce nastavení API – pro potřeby vývojářů, kteří budou případné propojení se svými aplikacemi zařizovat.

10.3.1 Návrátové kódy

Pro informování odesílající strany jsou používány tzv. návratové kódy. Jedná se o třiciferná čísla, kdy každé označuje nějaký konkrétní problém nebo naopak říká, že byl požadavek zpracován v pořádku. Některé z kódů jsou uvedeny níže.

Kód	Význam
000	požadavek z neznámého důvodu nebyl zpracován
001	chyba – špatná kombinace tokenu a autorizačního klíče
002	požadavek nemohl být zpracován – nebyly vyplněny všechny povinné hodnoty
003	požadavek nemohl být zpracován – v některém z polí byla zadána jiná hodnota, než je povolena
004	chyba – toto API je vypnuté
100	požadavek byl v pořádku zpracován

Seznam těchto návratových kódů je také uveden na stránce nastavení API.

10.3.2 Průběh zpracování požadavku

Server nejprve zkontroluje data, která na něj dorazila – zejména, zda nebyla žádná z povinných částí vynechána nebo zda nebyla některá část naplněna nesprávnými daty. Následně pomocí tokenu a autorizačního klíče vybere evidenci a zkontroluje, zda je pro ni API povoleno. Pokud ne, je zpracovávání ukončeno a nazpět je odeslán příslušný návratový kód.

Pakliže byla všechna data zadána v pořádku, je příslušný finanční záznam přidán do aktivní evidence.

10.3.3 Příklad JSON řetězce

```
{
  token: '...',
  authKey: '...',

  action: 'add',

  description: 'Magnetická nástěnka na zed',
  type: 'VY',
  amount: 499,
  tax: true,
}
```

11 Testování aplikace

Aplikace byla testována průběžně při vývoji. Každá nová funkce byla vždy ihned několikrát testována a případně odladěna. Ke konci vývoje jsem poprosil některé své přátele o co největší a nejkompexnější otestování, díky kterým bylo odhaleno několik drobných, ale i závažnějších chyb.

11.1 Registrace prokazatelně neexistujícího e-mailu

V controlleru, jenž zpracovává registrace, byla zjištěna zapomenutá implementace knihovny testující případnou nemožnost existence při registraci zadaného e-mailu.

Po běžném testu validity e-mailu (správnosti tvaru) měla být provedena kontrola existence domény a dále test existence MX záznamu, bez něž by e-mail s aktivačním odkazem nemohl být doručen (stejně jako v případě, že doména neexistuje).

Knihovnu, která se o toto pokročilé testování měla starat, jsem však implementovat zapomněl. Díky testování třetími osobami, jež zadaly při registraci absolutně nesmyslný e-mail, jsem na tento problém přišel a funkci do controlleru přidal. O testování se stará knihovna `legit`.

11.2 Přijetí řetězce jako částky

Byť jsem ve formulářích použil pro zadávání čísel `<input type="number">`, není nemožné odeslat požadavek například s řetězcem namísto čísla. Na backendu jsem zapomněl naprogramovat kontrolu datového typu, resp. zda je odeslaná hodnota číselná.

Ověření jsem tedy dodatečně implementoval, a kromě zmíněného bezpečnostního problému tím vyřešil zároveň další – ve starších prohlížečích, jež `type="number"` nepodporují, a do pole by šel tak psát snadno text, bude problém ošetřen také.

11.3 Neinformování o neúspěšnosti vytvoření záznamu

Při vytváření finančního záznamu mohlo dojít k různým chybám, jejichž ošetření bylo na straně serveru zajištěno – server vracel řetězec podle typu chyby. Na straně klienta však nebylo implementováno zobrazení chybové hlášky podle chyby a vždy se tak uživateli zobrazila zpráva, že byl záznam úspěšně vytvořen, byť vytvořen vůbec být nemusel. Neprodleně po zjištění problému jsem jej opravil.

Závěr

Ve své maturitní práci „Daňová evidence“ jsem splnil zadání v celém jeho rozsahu. Funkce aplikace jsem navíc ještě rozšířil a dbal přitom na dodržování zásady, aby ovládání aplikace bylo jednoduché, intuitivní a rychlé.

Protože jsem nikdy v podnikání nedosáhl stavu, kdy by má SVČ musela být vedena jako hlavní, bylo nutné doplnit vědomosti z oblasti výpočtů pojištění u hlavní činnosti a ty následně ve vývoji aplikovat.

Podařilo se mi vytvořit v praxi využitelnou aplikaci středního rozsahu, čímž jsem rozšířil své znalosti také z oblasti technologií Node.js a MongoDB, kterým se plánuji dlouhodobě věnovat.

Ve vývoji plánuji dále pokračovat – v plánu je zejména dokončení myšlenky plně automatického placení daně, možnost vystavování a vedení faktur a při propojení s bankovním účtem automatická kontrola zaplacenosti či např. zasílání e-mailových upozornění různého druhu (klient uhradil fakturu, dosažení stanoveného cílového zisku, nutnost podat daňové přiznání, ...).

Skvělé by bylo i vyvinutí možnosti propojení s datovou schránkou, a tedy kromě automatického placení přidat podporu automatického podání přiznání či Přehledů.

Přestože jsem vývoji věnoval maximum svého času, úsilí i zkušeností, v době odevzdání maturitní práce lze v aplikaci stále nalézt drobnosti, které, byť nebrání běžnému užívání, bude nutné před uvedením do ostrého provozu doladit, případně patřičně zdokonalit.

Seznam použitých zdrojů a literatury

- [1] Ministerstvo průmyslu a obchodu. *Roční přehled podnikatelů a živností*. [on-line]. 2018. Dokument ve formátu XLSX [cit. 2018-01-28].
<<https://www.mpo.cz/assets/cz/podnikani/zivnostenske-podnikani/statisticke-udaje-o-podnikatelich/2018/1/Rocni-prehled-podnikatelu-a-zivnosti.xlsx>>
- [2] PETRÁŇOVÁ, Marta a MEJSTŘÍK, Bohuslav. *Analýza*. [on-line]. 2015. Dokument ve formátu PDF [cit. 2018-01-05].
<<https://www.czso.cz/documents/10180/20568827/czam080315analyza.pdf>>
- [3] SANCHEZ, Roberto. *Comparing Node.js vs PHP Performance*. [on-line]. 2016. Webová stránka [cit. 2018-01-05].
<<http://www.hostingadvice.com/blog/comparing-node-js-vs-php-performance/>>
- [4] KARBAN, David. *MySQL sežere vaše data*. [on-line]. 2017. Dokument ve formátu PDF [cit. 2018-01-05].
<https://www.linuxdays.cz/2017/video/David_Karban-MySQL-sezere_vase_data.pdf>
- [5] PASHLEY, David. *MySQL silently truncating your data*. [on-line]. 2009. Webová stránka [cit. 2018-01-05]. <<http://www.davidpashley.com/2009/02/15/silently-truncated/>>
- [6] NPM, Inc. *npm*. [on-line]. 2017. Webová stránka [cit. 2018-01-05].
<<https://www.npmjs.com/>>
- [7] Wikipedia. *MongoDB*. [on-line]. 2017. Webová stránka [cit. 2018-01-05].
<<https://en.wikipedia.org/wiki/MongoDB>>
- [8] Wikipedia. *bcrypt*. [on-line]. 2018. Webová stránka [cit. 2018-01-28].
<<https://en.wikipedia.org/wiki/Bcrypt>>
- [9] Wikipedia. *Blowfish (cipher)*. [on-line]. 2018. Webová stránka [cit. 2018-01-28].
<[https://en.wikipedia.org/wiki/Blowfish_\(cipher\)](https://en.wikipedia.org/wiki/Blowfish_(cipher))>
- [10] MongoDB. *MongoDB Limits and Thresholds*. [on-line]. 2018. Webová stránka [cit. 2018-02-13]. <<https://docs.mongodb.com/manual/reference/limits/>>
- [11] Zlatá koruna. *Zlatá koruna 2013*. [on-line]. 2013. Webová stránka [cit. 2018-04-12]. <<http://www.zlatakoruna.info/soutez/2013>>

- [12] Zlatá koruna. *Zlatá koruna 2014*. [on-line]. 2014. Webová stránka [cit. 2018-04-12]. <<http://www.zlatakoruna.info/soutez/2014>>
- [13] Zlatá koruna. *Zlatá koruna 2015*. [on-line]. 2015. Webová stránka [cit. 2018-04-12]. <<http://www.zlatakoruna.info/soutez/2015>>
- [14] PETŘÍČEK, Martin. *PŘEHLEDNĚ: Bankovní účty pod kontrolou. Startuje revoluce v bankovníctví*. [on-line]. 2018. Webová stránka [cit. 2018-04-12]. <https://ekonomika.idnes.cz/psd2-banky-internet-0ok-/ekonomika.aspx?c=A180112_375717_ekonomika_rts>
- [15] SLÍŽEK, David. *Další banka začíná s API. Air Bank zatím rozhraní testuje s několika partnery*. [on-line]. 2017. Webová stránka [cit. 2018-04-12]. <<https://www.lupa.cz/aktuality/dalsi-banka-zacina-s-api-airbank-zatim-rozhrani-testuje-s-nekolika-partnery/>>
- [16] BUBÁK, Zdeněk. *Air Bank se připojila ke třem bankám nabízejícím bankovní rozhraní třetím stranám*. [on-line]. 2017. Webová stránka [cit. 2018-04-12]. <<http://www.finparada.cz/4668-Air-Bank-se-pripojila-k-bankam-nabizejici-bankovni-rozhrani-tretim-stranam.aspx>>

Seznam příloh

Příloha A – use case diagram aplikace

Příloha B – databázový model APIData

Příloha C – databázový model ConnectedBankAccount

Příloha D – databázový model FinancialRecord

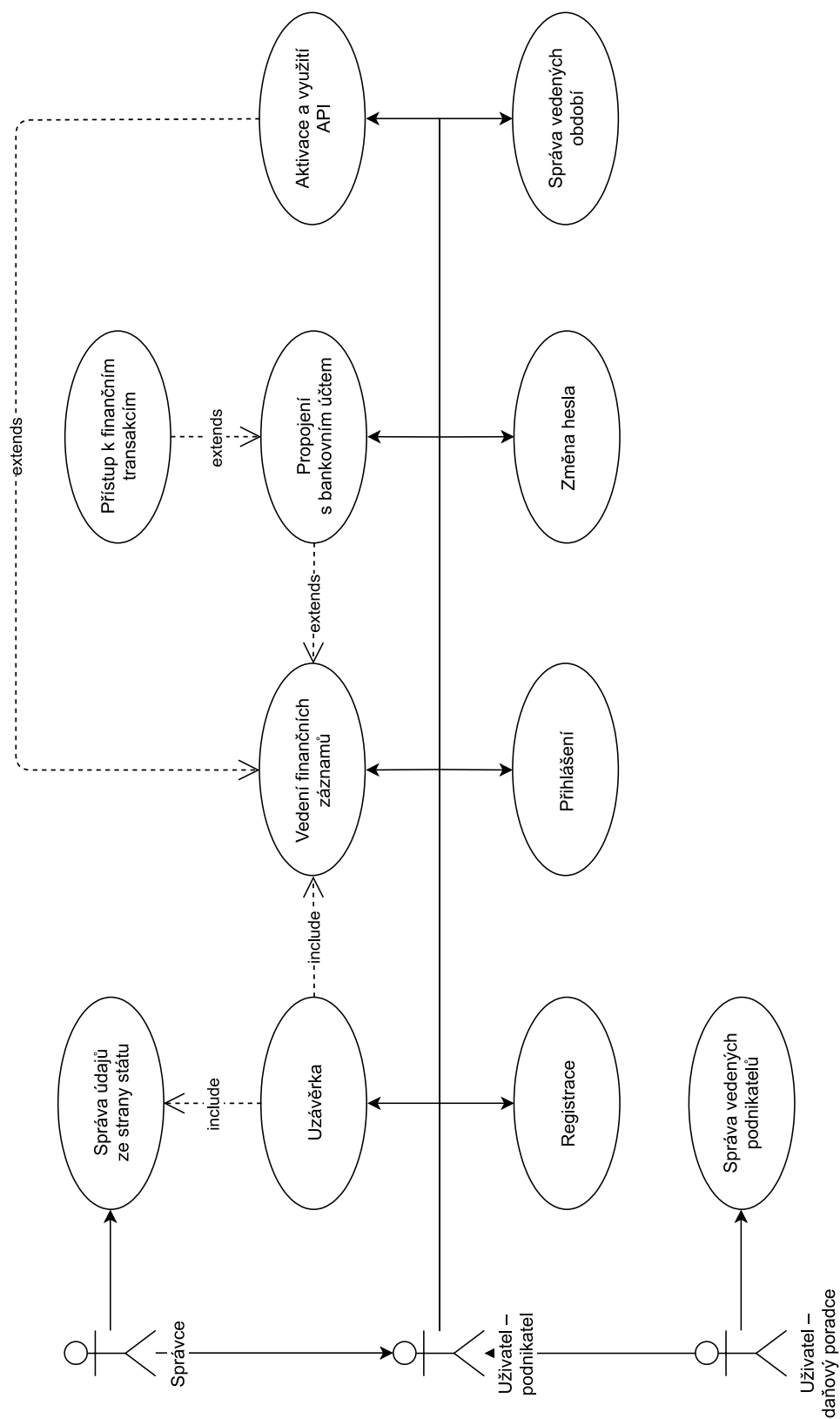
Příloha E – databázový model User

Příloha F – databázový model YearData

Příloha G – postup instalace softwaru

Příloha H – uživatelská příručka

Příloha A – use case diagram aplikace



Příloha B – databázový model APIData

```
{
  token: String,
  authKey: String,
  enabled: Boolean,
  belongsTo: {
    managingUserID: String,
    companyID: String,
  },
}
```

Příloha C – databázový model ConnectedBankAccount

```
{
  belongsTo: {
    managingUserID: String,
    companyID: String,
  },
  number: String,
  token: String,
  lastConnection: String,
  bank: String,
}
```

Příloha D – databázový model FinancialRecord

```
{
  belongsTo: String,
  description: String,
  date: Date,
  amount: {
    value: 'decimal',
    currency: {
      type: String,
      enum: [
        'CZK',
      ],
    },
  },
  recordType: {
    type: String,
    enum: [
      'income',
      'expense',
      'claim',
      'debt',
    ],
  },
  tax: Boolean,
  paymentMethod: {
    type: String,
    enum: [
      'bankAccount',
      'creditCard',
      'cash',
    ],
  },
  note: String,
  document: [{
    realMark: String,
    internalMark: String,
  }],
  archived: Boolean,
}
```

Příloha E – databázový model User

```
{
  email: String,
  password: String,
  name: {
    titleBefore: String,
    first: String,
    last: String,
    titleAfter: String,
  },
  residence: {
    street: String,
    numberO: String,
    numberP: String,
    city: String,
    ZIP: String,
  },
  companyID: String,
  dataBox: String,
  entrepreneurEvidence: [{
    year: Number,
    name: {
      titleBefore: String,
      first: String,
      last: String,
      titleAfter: String,
    },
    residence: {
      street: String,
      numberO: String,
      numberP: String,
      city: String,
      ZIP: String,
    },
    companyID: String,
    dataBox: String,
    closed: Boolean,
    closure: {
      date: Date,
      incomes: Number,
      expenditures: Number,
      claims: Number,
      debts: Number,
      profit: Number,
      tax: Number,
      socialInsurance: Number,
      healthInsurance: Number,
    },
  }],
  authorizedEvidence: [{
    managingUserID: String,
    companyID: String,
  }],
  status: {
    activated: Boolean,
    blocked: Boolean,
  },
}
```

```
controlKey: {
  accountActivation: String,
  accessRestore: String,
},
avatar: String,
userType: {
  type: String,
  enum: [
    'entrepreneur',
    'taxAdvisor',
  ],
},
seid: String,
serviceAdmin: Boolean,
}
```

Příloha F – databázový model YearData

```
{
  year: Number,
  tax: {
    child: [Number, Number, Number],
    childZTPP: [Number, Number, Number],
    student: Number,
    taxPayer: Number,
    dependentSpouse: Number,
    dependentSpouseZTPP: Number,
    disability: [Number, Number, Number],
    ZTPP: Number,
    maximumKindergarten: Number,
    EET: Number,
  },
  social: {
    primaryPensionMinimumFixedAmount: Number,
    primaryPensionMinimumMonthlyReductor: Number,
    primaryMinimumDeposit: Number,
    secondaryPensionMinimumFixedAmount: Number,
    secondaryPensionMinimumMonthlyReductor: Number,
    secondaryPensionFixedAmount: Number,
    secondaryPensionMonthlyReductor: Number,
    secondaryMinimumDeposit: Number,
    maximumFixedAmount: Number,
  },
  health: {
    minimumFixedAmount: Number,
    minimumDeposit: Number,
  },
  dates: [{
    date: Date,
    what: String,
  }],
}
```


Příloha G – postup instalace softwaru

Protože jde o webovou aplikaci postavenou na technologiích Node.js a MongoDB, je nutné na server tyto technologie nainstalovat. Samozřejmostí je pak správné nastavení webového serveru a předávání požadavků pomocí proxy aplikaci na port, na němž poslouchá.

Samotný software (webová aplikace) se pak instaluje umístěním veškerých souborů na připravený server do složky s doménou, na níž má aplikace běžet.

Aplikace je následně spuštěna spuštěním souboru `app.js` pod službou `node`. Typicky tak lze učinit zadáním příkazu `node app.js` do serverového terminálu.

Příloha H – uživatelská příručka

Uživatelská příručka je k dispozici na webové adrese <https://noilio.cz/prirucka/>.