Ekonomická fakulta
Faculty of Economics
Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Diplomová práce

Implementace EET pro e-shopy

Vypracoval: Bc. Jakub Cígl
Vedoucí práce: doc. Ing. Ladislav Beránek, CSc.

České Budějovice 2017

ZADÁNÍ

Prohlášení

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích, dne

Rád bych poděkoval panu doc. Ing. Ladislavu Beránkovi, CSc. za cenné rady a odbornou pomoc při zpracování mé diplomové práce. Mé díky rovněž patří vedení společnosti tisknisi s.r.o., které umožnilo aplikaci mnou navrhnutého řešení do ostrého provozu. A v neposlední řadě Ing. Romaně Kaiserové, která mi poskytla cenné rady v oblasti účetnictví.

Obsah

1 Úvod

Práce popisuje návrh, vytvoření a následnou implementaci aplikace pro elektronickou evidenci tržeb do e-shopů, postavených hlavně tzv. na zelené louce – tzn., že jsou napsané buď na míru, tedy nejsou provozovány na redakčních systémech přímo k tomu určených, anebo na málo rozšířených systémech, které například nejsou pravidelně aktualizované, což je i případ mnou používaného e-shopu Tisknisi.cz. Systémům pro správu e-shopů se v oboru říká krabicová řešení. Ještě detailněji jde o to, že administrativní prostředí daného e-shopu mají všichni majitelé těchto softwarů téměř stejné, liší se kupříkladu doplňky, které se dají přikoupit a doinstalovat do administrace. To je důvodem, proč se moje práce zabývá e-shopy, které tyto doplňky nemají. Majitelé těchto krabicových řešení totiž doplněk pro elektronickou evidenci tržeb jistě pro své zákazníky zajistili sami, tedy naprogramovali pro ně doplněk do jejich administrací.

Důvodem, proč jsem se tedy rozhodl vytvořit toto řešení a napsat o tom diplomovou práci je fakt, že lidé, kteří nepoužívají krabicové řešení, nebo používají nerozšířené nebo již neudržované verze, by měli implementaci pro evidenci tržeb velmi finančně náročnou. Zpravidla mají totiž tito lidé k dispozici například jen jednoho programátora, který pro ně maličkosti na e-shopu upravuje. Často to bývá například jen změna nadpisů, úprava hlavní neboli vstupní stránky e-shopu, oprava chyb, které se časem objevily, změny obrázků anebo lehké úpravy detailů produktů, prodávaných v obchodě. Hodinové sazby těchto programátorů nejsou zanedbatelné, a tak jsem se rozhodl čas implementace rapidně zkrátit.

V teoretické části tedy popisuji všechny náležitosti, kterých se aplikace týká. Seznámím čtenáře se samotným zákonem o evidenci tržeb. Popíši základní diagramy používané pro návrhy softwaru. Vysvětlím základy programování webových aplikací, tedy detailně frontend i backend. A představím e-shop Tisknisi.cz, se kterým spolupracuji již dva roky a bylo mi umožněno použít na něm mnou navrhované řešení.

Praktická část detailně představí samotnou aplikaci, ukáže její architekturu pomocí diagramů, následně vytvořené wireframy, počeštěně drátěné modely, které pomáhají vytvořit náhled řešení. Nebudou chybět ukázky důležitých částí kódu. Dále popíši proces nasazení na server, testování a implementaci do e-shopu. Závěrem navrhnu cenovou politiku a vyhodnotím ekonomickou výhodnost tohoto řešení.

2 Cíle práce

Cílem této diplomové práce je vytvoření aplikace pro elektronickou evidenci tržeb, která usnadní implementaci do e-shopů, které jsou postavené tzv. na zelené louce. Práce současně popíše charakteristiku a základní pojmy elektronické evidence tržeb, ale také technologie používané při vývoji aplikace. Ta by měla jak usnadnit práci programátorům, protože většina složité logiky se bude odvíjet na straně aplikace, tak hlavně ušetřit peníze majitelům e-shopů. A to z toho důvodu, že implementace mého řešení oproti sestavení a implementaci řešení vlastního, které vyplývá z dokumentace Finanční správy, zkrátí čas na minimum. Pro ziskovost této aplikace s názvem E-EET navrhnu přijatelnou cenovou politiku.

3 Teoretická část

3.1 Elektronická evidence tržeb (EET)

V této kapitole budu čerpat hlavně z oficiálních stránek www.etrzby.cz, které jsou provozovány přímo Finanční správnou, a webem www.eltrzby.cz, který je provozován Asociací malých a středních podniků a živnostníků ČR, zkráceně AMSP ČR. Druzí jmenovaní lobovali při návrhu zákona i po jeho schválení o výjimky anebo změny. Bohužel totiž obsahoval mnoho nepřesností a obecností, které zhoršovaly schopnost porozumění této novince. Spoustu termínů nebo povinností si mohl každý vyložit jinak. Dále obsahuje chyby a je příliš krátký. Což například zmiňuje i pan Matzner ve svém článku na uznávaném webu Podnikatel.cz. (Matzner, 2015)

Samotné téma elektronická evidence tržeb, zkráceně EET, bylo jistě během let 2015 a 2016 jedním z nejvíce diskutovaných, a hlavně kontroverzních napříč celou Českou republikou.

Jde o elektronický způsob evidování každé platby, kde je obchodník nucen nejpozději v okamžiku zaevidování tržby poslat základní údaje o této transakci přímo na server Finanční správy, kde budou informace shromažďovány, následně obratem obdrží unikání kód, který je podnikatel povinen uvést na účtence i s dalšími údaji a poté účtenku poskytnout zákazníkovi. (Asociace malých a středních podniků a živnostníků ČR (AMSP ČR), 2016)

Z toho vyplývá, že každá platba má svůj unikátní kód, s kterým je spojena a lze tedy v budoucnu kdykoliv tuto platbu spárovat.

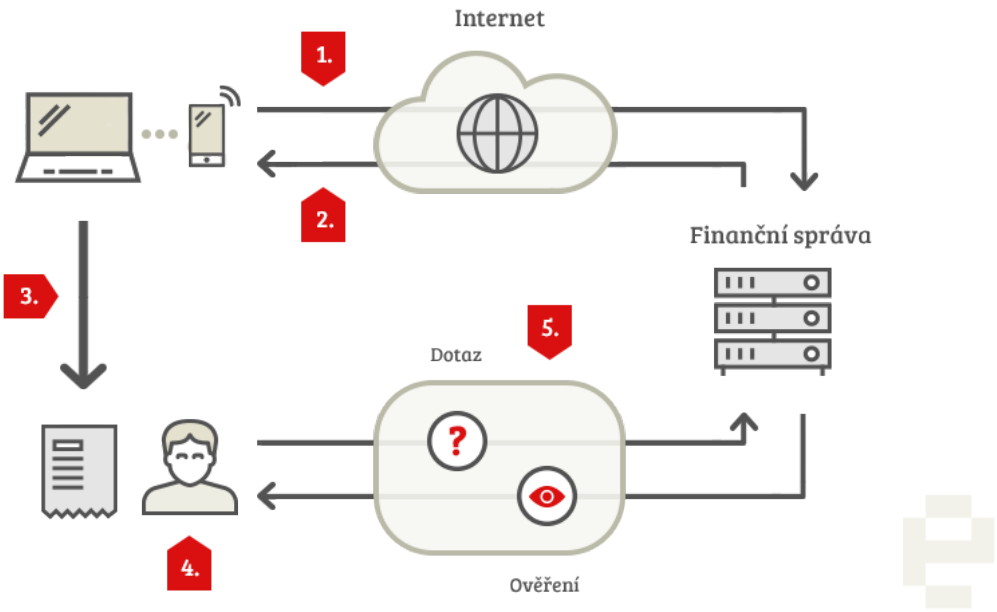
3.1.1 Princip a požadavky

Nyní cituji kroky procesu přímo z webu Finanční správy a přikládám obrázek pro jejich lepší porozumění: (Finanční správa ČR, 2016)

1. Podnikatel zašle datovou zprávu o transakci ve formátu XML Finanční správě.
2. Ze systému finanční správy je zasláno potvrzení o přijetí s unikátním kódem FIK.
3. Podnikatel vystaví účtenku (včetně FIK), kterou předá zákazníkovi.
4. Zákazník obdrží účtenku.

5. Zákazník si může ověřit svoji účtenku na Daňovém portále, podnikatel si ověří tržby evidované pod jeho jménem ve webové aplikaci Elektronická evidence tržeb.

Obrázek 1 - Jak to funguje?



Zdroj: http://www.etrzby.cz/cs/jak-to-funguje

Pro evidování tržby je potřeba zařízení, které v okamžiku přijetí platby dokáže poslat záznam na již zmiňovanou Finanční správu. Volba tohoto zařízení je přímo na podnikateli a liší se třeba i druhem podnikání. Může jim být počítač, tablet, mobil, pokladní zařízení, nebo software/aplikace běžící na serveru. Poslední zmiňovaná je i náš případ a týká se hlavně elektronických obchodů. Je pro ně určitě jednoduší implementovat posílání informací o tržbě elektronicky a automaticky při běhu e-shopu než po každé objednávce, například na pokladním zařízení, markovat.

Zajímavostí, oproti zahraničí, konkrétně Chorvatsku, kde se současný ministr financí ČR Ing. Andrej Babiš inspiroval, je, že podnikatelé mají volbu opravdu vlastní. Ve zmiňovaném Chorvatsku bylo certifikováno několik společností, které autorizované pokladny distribuovaly. Dle mého názoru se tomuto pan ministr rovnou vyhnul, aby nevznikaly další rozepře mezi společností a názory typu „obohacení pro majitele těchto společností“.

Dále je potřeba mít zajištěné připojení k internetu. V té souvislosti se dostáváme k velmi časté otázce, co když něco nebude fungovat? Na to ale zákon pamatuje, a tak určuje jisté výjimky nebo postupy. (Finanční správa ČR, 2016)

1. **Co když nejde internet?** Podnikatel má poté právo vydat účtenku bez unikátního kódu a údaje o tržbě poslat na Finanční správu bezprostředně po obnovení spojení, nejdéle však 48 hodin po uskutečnění platby.
2. **Co když nebude k dispozici vůbec žádné internetové připojení?** Jeli provozovna kupříkladu na odlehlém místě, dostane podnikatel výjimku a může vydávat účtenky bez kódů. Nicméně je povinen, nejpozději do pěti dnů od tržby, zaslat údaje na servery Finanční správy.
3. **Co když spadne spojení během transakce?** V případě výpadku, kde viníkem může být i Finanční správa, která může mít poruchu, platí obdobné pravidlo jako pro výpadek internetu. Jakmile to bude možné, nejpozději do 48 hodin od obdržení tržby, provést evidenci.
4. **Co když udělám chybu na pokladně a budu muset provést storno?** Jestliže se dopustíte chyby, jednoduše pošlete na Finanční správu údaje obdobným způsobem, jen jako mínusovou položku. Storno totiž není vázané na původní tržbu a není to tedy přímo spojené. Jinak řečeno, nestornujete špatně provedenou tržbu nebo její výši. Ale zašlete údaje s mínusem, pro finální přesnou částku, kterou disponujete.
5. **Co když se mi rozbije pokladna, to budu muset zavřít?** Podnikatel může svou činnost vykonávat samozřejmě dál. Musí ale dokázat, že učinil vše proto, aby mohl co nejdříve opět evidovat. Údaje, které nasbíral během nefunkčnosti zařízení, pošle zpětně.

3.1.2 Technické řešení

Po vysvětlení základních principů, bych rád přešel ke konkrétnímu případu, který nás zajímá. Jak evidence probíhá v online prostředí, neboli jak automaticky zaevidovat a bezpečně poslat tržby na servery Finanční správy, když nemáme k dispozici, a ani nechceme mít, nějaký druh pokladního zařízení, který je na to naprogramovaný. Jak jsem výše uvedl, evidovat tržby ručně po přijetí objednávky na e-shopu je zdlouhavé, neproduktivní a zbytečné.

V této kapitole budu vycházet hlavně z oficiálního dokumentu finanční správy s dlouhým názvem **Formát a struktura údajů o evidované tržbě a popis datového rozhraní pro příjem datových zpráv evidovaných tržeb.** (Finanční správa ČR, 2016) Zároveň postrádá smysl citovat a přepisovat celý dokument, proto popíšu jenom části důležité pro pochopení problematiky a části, kterých se naše řešení dotkne. Pro plné porozumění danému tématu doporučuji přečíst zmiňovaný dokument.

Cílem této diplomové práce není nauka IT základů, z toho důvodu vynechávám i popisy základních informačních prvků jako server, SOAP, XML, ASCII, UTF-8 a další. Následující text počítá s jejich základní znalostí.

3.1.2.1 Komunikační scénář

Pokladní zařízení, v našem případě e-shop, tedy zasílá data na servery finanční zprávy. Jestliže jsou zasílaná data v pořádku a vyhovují požadavkům, kterými jsou, cituji:

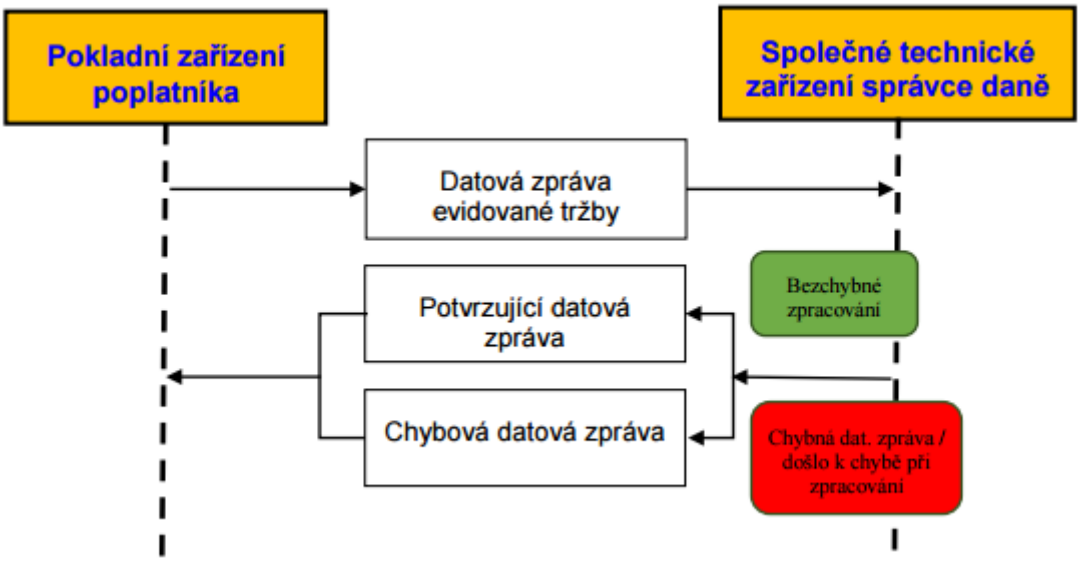
1. kontrola kódování XML dokumentu – předepsáno je kódování UTF-8
2. kontrola na konkrétní XML schéma (*.xsd) datové zprávy evidované tržby, které obsahuje přesnou definici struktury dat a formátů jednotlivých datových položek a kontrola přítomnosti povinných položek
3. kontrola elektronického podpisu datové zprávy (certifikát poplatníka je součástí SOAP obálky datové zprávy dle standardu WS-Security):
    a. kontrola vydavatele certifikátu
    b. kontrola platnosti certifikátu včetně kontroly CRL, které jsou aktuálně technickému zařízení dostupné
    c. kontrola správnosti podpisu
4. kontrola toho, že BKP přísluší k uvedenému PKP
5. kontrola integrity DIČ poplatníka
6. kontrola celkové délky datové zprávy evidované tržby (tj. zpráva včetně SOAP obálky), která nesmí přesáhnout 12 kB

Tak jsou uloženy na datové uložiště správce daně neboli Finanční správy ČR. Následně se vytvoří potvrzovací zpráva, která je poslána zpět na e-shop, který zprávu poslal. Komunikace tedy probíhá na principu požadavek – odpověď. Odpovědí je potvrzovací zpráva, která říká, že data zasílaná z e-shopu jsou správná, respektive mají správný formát. Tyto datové zprávy jsou spolu svázány **bezpečnostním kódem poplatníka**

(BKP) a číslem provozovny. Odpověď obsahuje již zmíněný **fiskální identifikační kód (FIK)**, který je generován na straně správce daně a je pro každou transakci jedinečný.

V případě, že datová zpráva z e-shopu nesplňuje některé z kritérií výše uvedených, nebo nastane nějaký technický problém na straně správce daně a zprávu nebude možné zpracovat, vrátí se chybová datová zpráva, pokud to ovšem vlastnost chyby umožní.

Obrázek 2 - Komunikační scénář



Zdroj: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

Pro testování a vývoj softwarů i pokladních zařízení existují dva módy datové zprávy a dvě prostředí, ve kterých si mohou vývojáři vyzkoušet funkčnost jejich řešení. S oběma prostředími je možné komunikovat oběma módy.

Ostrý mód

Slouží pro samotnou evidenci tržeb a obdržení fiskálního identifikačního kódu. V datové zprávě musí být hlavička hodnoty, která mód určuje. Musí být buď nulová, nebo nastavená na hodnotu false.

Ověřovací mód

Údaj v hlavičce obsahuje naopak hodnotu true. Slouží k ověření správnosti komunikace i ostatních nastavení. Hodnota tržby není ukládána do evidence.

Produkční prostřední

Slouží pro rutinní provoz, tedy pro zasílání údajů o tržbě a obdržení správných údajů zpět. Tržby budou zaevidovány.

Neprodukční prostředí (playground)

Spuštěno hlavně pro vývojáře, kteří mohli testovat jimi navrhnuté řešení. Není určeno pro koncové uživatele, kteří musí evidovat tržby. Údaje zasílané na toto prostředí nejsou evidovány na straně správce daně a FIK, který toto prostředí vrací, je neplatný. Existují cvičné certifikáty pro navázání komunikace s tímto prostředím.

Standardy síťové komunikace

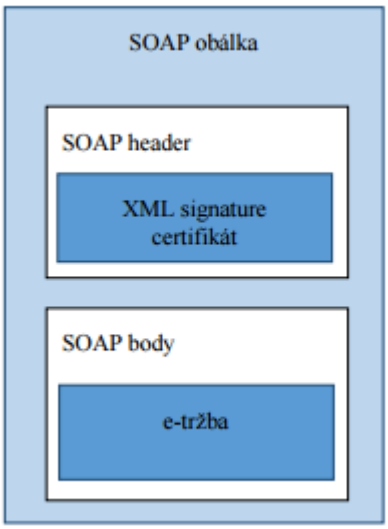
Použití HTTPS je povinné, bez autentizace klientskými certifikáty. Podporované verze TLS jsou TLS 1.1 a vyšší ale Finanční správa doporučuje TLS 1.2.

Dále je povinné použít prosté HTTP/1.1.

3.1.2.2 Struktura datových zpráv

Pro komunikaci se používá datový formát daný aplikačním protokolem SOAP, založeným na XML. Principem tedy je zabalení struktury XML do tzv. SOAP obálky.

Obrázek 3 - Struktura datové zprávy evidované tržby



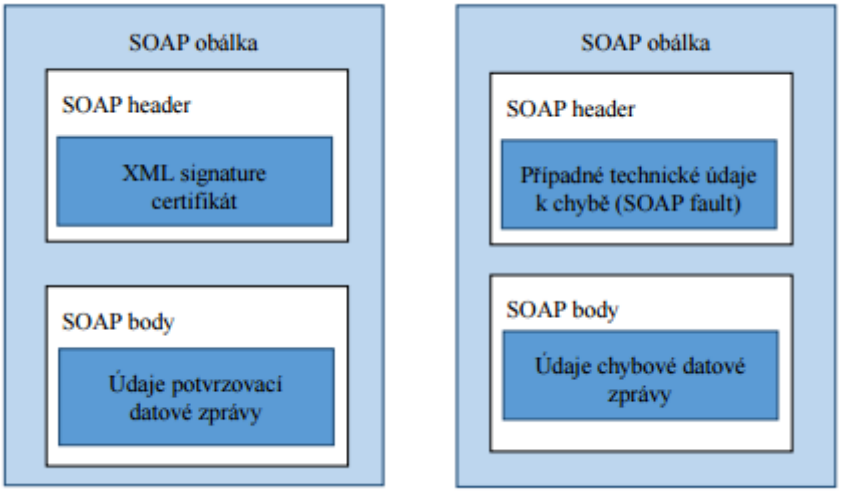
Na obrázku 3 vidíme schéma datové zprávy evidované tržby, která jde ze strany e-shopu na servery Finanční správy ČR.

Na obrázku 4 naopak vidíme zprávy, které cestují opačným směrem. Všimněme si, že chybová zpráva vpravo neobsahuje podpis/certifikát, ale co nejvíce informací o chybě.

Začátek textového řetězce se značí ^ a pro konec textového řetězce je zvolen $.

Zdroj: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

Obrázek 4 - Struktura potvrzovací a chybové datové zprávy



Zdroj: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

Kódování XML dokumentů je povinně UTF-8, takže první řádek SOAP obálky bude mít tvar:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Všechny elementy e-tržby patří do jmenného prostoru. Maskou datového formátu rozumíme správnou strukturu XML schématu pro jednotlivé položky, které obsahuje a jsou detailně definovány v dokumentu zmíněném výše.

Hexadecimální číslice větší než 9 je umožněno zapisovat jak velkými, tak malými písmeny.

V XML elementu <SOAP Envelope Header> je uložen XML signature a klíč, který byl použit pro jeho vytvoření.

Informace o evidované tržbě budou v elementu <Trba>, který bude vnořen do již zmíněného <SOAP Envelope Header>

Samotná **struktura e-tržby (posílané zprávy)** poté vypadá takto:

```
<eet:Trzba>
     <eet:Hlavicka atributy … />
     <eet:Data atributy … />
     <eet:KontrolniKody>
     hodnoty …
     </eet:KontrolniKody>
</eet:Trzba>
```

Zdroj: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

V tabulce níže naleznete atributy a hodnoty XML elementů. Vypíši pouze mnou používané, které lze posílat, bylo nutné je použít pro aplikaci na e-shopu. S tím souvisí i fakt, že ve sloupci *Povinné* můžeme najít hodnotu buď *Ano*, nebo *Ne*. Zvolený název je dle mého názoru Finanční správnou špatně zvolen. Ve skutečnosti totiž znamená, které údaje obsahuje každá datová zpráva a které údaje musí každý poplatník sám určit, zda je nucen je posílat také.

Tabulka 1 - Přehled položek datové zprávy o evidované tržbě

Datová oblast	**Pomocné číslo**	**Název položky**	**Povinné**	**XML jméno**
Hlavička	1	UUID zprávy	Ano	uuid_zpravy
	2	Datum a čas odeslání zprávy	Ano	dat_odesl
	3	První zaslání údajů o tržbě	Ano	prvni_zaslani
	4	Příznak ověřovacího módu odesílání	Ne	overeni
Data	5	DIČ poplatníka	Ano	dic_popl
	7	Označení provozovny	Ano	id_provoz
	8	Označení pokladního zařízení	Ano	id_pokl
	9	Pořadové číslo účtenky	Ano	porad_cis

	10	Datum a čas přijetí tržby	Ano	dat_trzby
	11	Celková částka tržby	Ano	celk_trzba
	12	Celková částka plnění osvobozených od DPH, ostatních plnění	Ne	zakl_nepodl_dph
	13	Celkový základ daně se základní sazbou DPH	Ne	zakl_dan1
	14	Celková DPH se základní sazbou	Ne	dan1
	25	Režim tržby	Ano	rezim
Kontrolní kódy	26	Podpisový kód poplatníka (PKP)	Ano	pkp
	27	Bezpečnostní kód poplatníka (BKP)	Ano	bkp

Zdroj: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

Následný technický popis uvedených položek doplňuji vysvětlujícím textem z dokumentu **Popis položek datové zprávy a příklady situací při evidenci tržeb** (Finanční správa ČR, 2016):

1. **UUID zprávy**

Doporučuje se používat UUID verze 4. Je to jedinečný identifikátor, který generuje e-shop a jednoznačně tak identifikuje datovou zprávu.

Délka řetězce má 36 znaků a tvar:

```
xxxxxxxx-xxxx-Mxxx-Nxxx-xxxxxxxxxxxx
```

Kde *x, M* a *N* značí hexadecimální číslice. Konkrétně poté *M* verzi UUID a *N* povinně binární hodnotu `10`. Z toho důvodu, že řetězec *Nxxx* smí nabývat jen hexadecimálních hodnot 8, 9, A, B.

2. **Datum a čas odeslání zprávy**

Přesný datum a čas k okamžiku odeslání datové zprávy. Ve formátem DateTime dle ISO 8601:

```
rrrr-mm-ddThh:mm:ss±hh:mm
```

Tedy rok, měsíc, den, fixní znak *T*, hodina, minuta, sekunda. Dále rozlišení, jestli je rozdíl proti světovému času kladný (+) nebo záporný (-) a jeho hodnotu. Je třeba myslet na český letní a zimní čas. Při letním bude hodnota +02:00 a při zimním +01:00.

3. **První zaslání údajů o tržbě**

Tento atribut slouží pro rozlišení, zda se jedná o první pokus zaevidování tržby nebo již několikátý. Nejčastěji se jedná o případ, kdy například vypadne internetové připojení nebo se ze serverů Finanční správy vrátí chybová informace. Posílají se tedy tytéž údaje jako při prvním pokusu, jenom s hodnotou *false* v tomto atributu. Při prvním pokusu se nastaví naopak logicky *true*.

4. **Příznak ověřovacího kódu odesílání**

Atribut nalezneme v hlavičce a značí, zda evidujeme, nebo spíše testujeme v ověřovacím módu, poté nabývá hodnoty *true* a údaje nejsou ukládány na server. Anebo tento atribut neuvedeme, případně nastavíme na *false* a tím říkáme, že komunikujeme v ostrém módu. Oba módy jsou popsány výše.

5. **DIČ poplatníka**

Je atributem již v datové části zprávy. Obsahuje to, co říká, tedy DIČ poplatníka. Musí obsahovat i kód země. A je to DIČ, které má obchodník v okamžiku provedení platby. V případě změny DIČ může odesílat datové zprávy s novým, i když certifikát, do kterého je DIČ promítnuté, ještě odpovídá starému. Takto může postupovat, dokud mu není vydán certifikát nový.

Délka je deset až dvanáct znaků. Maska datové zprávy vypadá následovně:

```
^CZ[0-9]{8,10}$
```

7. **Označení provozovny**

Jedná se o číselnou hodnotu, která je přiřazena poplatníkovi v aplikaci pro elektronickou evidenci tržeb. Unikátní označení provozovny je tedy dáno dvojicí údajů:

```
(dic_popl, id_provoz)
```

Délka je jeden až šest znaků, od hodnot 1 až 999999. Maska má tento tvar:

```
^[1-9][0-9]{0,5}$
```

8. Označení pokladního zařízení poplatníka

Musí být v rámci provozovny jedinečné. Proto platí, že kombinace těchto čtyř hodnot je unikátní:

```
(dic_popl, id_provoz, id_pokl, dat_trzby)
```

V našem případě se jedná o označení konkrétního e-shopu. Kdybychom měli v evidenci e-shopů více a provozovali je pod jedním DIČ, dostane každé své číslo.

Nabývá délky jeden až dvacet znaků a maska datového formátu vypadá takto:

```
^[0-9a-zA-Z\.,:;/#\-_ ]{1,20}$
```

Zajímavostí je, že poslední znak v hranaté závorce je mezera. Dekadický kód 32 v ASCII znakové sadě.

9. Pořadové číslo účtenky

Údaj, který volí poplatník, je na něm, jakou mu přiřadí hodnotu, samozřejmě ale musí dodržovat určitou masku. Ta je téměř stejná jako označení pokladního zařízení s tím rozdílem, že zde může nabývat delšího řetězce a to 25 znaků:

```
^[0-9a-zA-Z\.,:;/#\-_ ]{1,25}$
```

Doporučuje se volit takovou hodnotu, která má nějakou vypovídající schopnost a pomůže například platbu v budoucnu identifikovat. V e-shopu jsem zvolil nejjednodušší možnou variantu a to, že tento atribut dědí číslo objednávky. Mám tak spárovanou každou platbu ke konkrétní objednávce.

Opět platí, že v jeden okamžik musí existovat jediná kombinace účtenky, zařízení a poplatníka:

```
(dic_popl, id_provoz, id_pokl, porad_cis,
```

10. Datum a čas přijetí tržby

Formát je naprosto identický s již zmiňovaným Datem a časem odeslání zprávy. Může se zdát, že se tedy jedná o duplicitu, ale není tomu tak. Zde se uvádí čas, který je uveden i na účtence a s ohledem na to, že může dojít k chybě, nebo evidenci děláme zpětně kvůli výpadku internetu. Budou se poté tyto atributy lišit. Stejně tak může jít o případ, když poplatník je povinen evidovat tržbu až po přijetí platby. To znamená, že datum objednávky na e-shopu může být starší než datum odesílané zprávy.

11. Celková částka tržby

Tím se dostáváme k atributům, které nesou peněžní hodnoty v korunách. Jsou uvedeny v dekadické soustavě s dvěma povinnými desetinnými místy oddělenými tečkou. Hodnoty můžou být kladné, záporné i nulové.

Maska datového formátu:

```
^((0|-?[1-9]\d{0,7})\.\d\d|-0\.(0[1-9]|[1-9]\d))$
```

Velikost může být tedy od mínus 100 milionů Kč do plus 100 milionů Kč.

Celkovou částkou tržby se rozumí celková suma přijatá poplatníkem. Musíme ji tedy uvádět i s DPH, a dokonce i s platbou za dopravu anebo způsob platby, která bývá v e-shopech v případě dobírky zpoplatněna. Uvádíme v Kč, v případě příjmu cizí měny se přepočítává jiným vhodným kurzem, přičemž se doporučuje aktuální kurz ČNB.

Zde nastává jedna zvláštnost. Je důležité si uvědomit, že máme uvádět částku, kterou jsme skutečně přijali. V případě hotovosti tedy zaokrouhlujeme, protože haléře již nejsou platidlem. V případě převodu na účet nebo platby kartou, ale můžou vzniknout desetinná čísla. Zejména připočítáváním DPH a jinými operacemi jakými jsou slevy. Která mi skutečně obdržíme. Tedy dávat si na to pozor, protože z pohledu několika let by halíře udělaly velký rozdíl.

12. Celková částka plnění osvobozených od DPH, ostatních plnění

Jednoduše, neekonomicky řečeno, se zde v mém případě uvádí částka bez DPH. Tento atribut se posílá sám, bez atributů číslo 13 a 14 v případě, kdy eviduji tržbu na Slovensko a zákazník je plátce daně.

13. Celkový základ daně se základní sazbou DPH

Tento údaj taktéž obsahuje částku bez DPH a posílá se společně s číslem 14, tedy s *Celkovým DPH se základní sazbou*. Datová zpráva ho obsahuje pouze v případě fakturace pro českého zákazníka a v případě fakturace na Slovensko, jestliže je zákazník neplátce.

14. Celková DPH se základní sazbou

Obsahuje samotnou výši DPH. Posílá se společně s údajem číslo 13 a ve stejných případech plátcovství. Z toho plyne, že datová zpráva obsahuje buď údaj 12, nebo údaje 13 a 14.

25. Režim tržby

Má pouze jeden znak a ten se liší zvoleným režimem:

- 0 = běžný režim = online
- 1 = zjednodušený režim = off-line

Off-line režim je v současné době možné použít pouze v případě výjimky udělané správcem daně, tedy Finanční správou ČR.

Atributy 26 a 25 popíši v jejich vlastní kapitole 2.1.2.3, kvůli jejich rozsahu.

Co se týká **formátu potvrzovací zprávy**, tak formát XML vypadá následovně:

```
<eet:Odpoved>
     <eet:Hlavicka atributy … />
     <eet:Potvrzeni atributy … />
     <eet:Varovani atributy … >
     Hodnoty …
     </eet:Varovani>
     <eet:Varovani atributy … >
     hodnoty …
     </eet:Varovani>
     …
</eet:Odpoved>
```

Zdroj: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

Přehled datových položek je uveden v tabulce níže:

Tabulka 2 - Přehled datových položek potvrzení

Datová oblast	Pomocné číslo	Název položky	Povinné	XML jméno
Hlavička	1	UUID zprávy	Ano	uuid_zpravy
	2	Datum a čas přijetí zprávy	Ano	dat_prij

	3	Bezpečnostní kód poplatníka	Ano	bkp
Potvrzení	4	Fiskální identifikační kód	Ano	fik
	5	Příznak neprodukčního prostředí	Ne	test
Varování	6	Kód varování	Ne	kod_varov
	7	Textový popis varování	Ne	varovani

Zdroj: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

Atribut číslo 1 již známe. Číslo 2 má opět stejné náležitosti, jen říká, kdy systémy Finanční správy evidenci přijaly. Atribut 3 popíši v sekci 2.1.2.3.

4. **Fiskální identifikační kód (FIK)**

První údaj, který se nachází v části <Potvrzeni>. Jde o jedinečný údaj, který generuje strana Finanční správy a propisuje se do něj konkrétní označení zařízení, které evidenci přijalo. Například konkrétní server a jeho ID. Jeho délka je 39 znaků a masku zapíšeme takto:

```
^[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-4[0-9a-fA-F]{3}-
[89abAB][0-9a-fAF]{3}-[0-9a-fA-F]{12}-[0-9a-fA-F]{2}$
```

Pokud je FIK tvořen v neprodukčním prostředí, nabývají poslední dva znaky hodnoty *ff*.

5. **Příznak neprodukčního prostředí**

Oznamuje poplatníkovi, kam zaslal datovou zprávu. V případě, že je ve správě uvedeno *true*, byla datová zpráva přijata do neprodukčního prostředí. Není-li tento údaj uveden, nebo nabývá hodnoty *false*, byly údaje evidované do produkčního prostředí.

6. **Kód varování**

Je celé a kladné dekadické číslo o maximálně třech znacích, nacházející se v části <Varovani>, které dle definované tabulky určuje dané varování. Zajímavostí je, že je zatím definováno pouze 5 varování a v oficiální zprávě je napsáno, že zbylá čísla jsou rezervována do budoucna. Působí to na mě tak, že budou na lidech testovat, jaké chyby dokážou udělat a poté je budou definovat.

7. **Textový popis varování**

Stručná zpráva o tom, o jaké varování se jedná. Zpráva neobsahuje diakritiku a má maximální délku sto znaků.

Posledním možnou zprávou je chybová datová zpráva, která má XML strukturu takto:

```
<eet:Odpoved>
     <eet:Hlavicka atributy … />
     <eet:Chyba atributy … />
     Hodnoty …
     </eet:Chyba>
     <eet:Varovani atributy … >
     Hodnoty …
     </eet:Varovani>
</eet:Odpoved>
```

Zdroj: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

Přehled datových položek je:

Tabulka 3 - Přehled datových položek chyby

Datová oblast	**Pomocné číslo**	**Název položky**	**Povinné**	**XML jméno**
Hlavička	1	UUID zprávy	Ne	uuid_zpravy
	2	Datum a čas přijetí zprávy	Ne	dat_prij
	3	Bezpečnostní kód poplatníka	Ne	bkp
Chyba	4	Chybový kód	Ano	Kod
	5	Textový popis chyby	Ano	chyba
	6	Příznak neprodukčního prostředí	Ne	test
Varování	7	Kód varování	Ne	kod_varov

	8	Textový popis varování	Ne	varovani

Zdroj: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

Většinu položek již známe z potvrzovací zprávy. Změnila se nám část **Chyba**, kde přibyl *Chybový kód* a *Textový popis chyby* – který má stejné formátové požadavky jako *Textový popis varování*, takže se mu nebudu věnovat.

4. **Chybový kód**

Zvláštností je, že má opět stejné formátové požadavky jako *Kód varování*, ale s tím rozdílem, že může nabývat i záporných hodnot. Pro ty může nabývat i 4 znaků, aby před trojici čísel mohlo být umístěno znaménko -. Maska se zapíše takto:

```
^-?\d{1,3}$
```

Seznam chyb je tentokrát rozsáhlejší a je v něm deset definovaných chyb s tím, že zbytek je opět rezervován pro budoucnost.

3.1.2.3 Kontrolní kódy PKP a BKP

Podpisový kód poplatníka – PKP

Pomocí tohoto klíče bude platba jednoznačně identifikována. Je to elektronický podpis kombinace položek obsahující datovou zprávu. Konkrétně se jedná o položky, které jsem již zmiňoval:

Tabulka 4 - Položky pro tvorbu PKP

Pomocné číslo	Název položky	XML jméno	Hodnota
5	DIČ poplatníka	dic_popl	CZ28147596
7	Označení provozovny	id_provoz	11
8	Označení pokladního zařízení	id_pokl	1
9	Pořadové číslo účtenky	porad_cis	117020355
10	Datum a čas přijetí tržby	dat_trzby	2017-06-06T17:33:22+01:00
11	Celková částka tržby	celk_trzba	359.00

Zdroj: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

V tabulce uvádím i příklady hodnot. Tzv. plaintext pro výpočet PKP poté vzniká zřetězením, kde jednotlivé položky dělí znak „|“, takže v našem případě by vypadal, takto:

```
CZ28147596|11|1|117020355|2017-06-06T17:33:22+01:00|359.00
```

Z tohoto řetězce se vypočte otisk algoritmem SHA256, ten se zašifruje neboli podepíše soukromým klíčem poplatníka, který ho získal z webové aplikace pro elektronickou evidenci tržeb. Tento soukromý klíč je třeba si skladovat na bezpečném místě. Výsledek šifrace je `rsa_text`, ten se následně zakóduje algoritmem Base64 do textového řetězce `rsa_text_base64` a ten je následně posílán jako hodnota XML elementu <pkp>.

Zmiňovaný privátní klíč je jednoznačně spjat s veřejným klíčem, jenž je součástí X509 certifikátu, který se posílá v <SOAP header> datové zprávy. Je totiž nutný pro dešifraci a porovnání zasílaných hodnot. Když se shodují, nedošlo k odposlechu a případnému nahrazení dat.

Bezpečnostní kód poplatníka – BKP

Pro určení této hodnoty se použije zmiňovaný řetězec `rsa_text` z PKP. Z něho se vytvoří otisk `hash_sha1` podle algoritmu SHA1. Následně se ještě hexadecimálně zakóduje do textového řetězce `hash_sha1_base16`. A posledním krokem je úprava do finálního tvaru a to tak, že mezi znaky 8 a 9, 16 a 17, 24 a 25, 32 a 33 vložíme znak „-“ (dekadický kód 45 v ASCII znakové sadě). Máme tedy pět bloků po osmi číslech dělené pomlčkou.

3.1.2.4 Údaje uváděné na účtence

Zákon stanovuje povinné údaje, které se musí objevit na vydávané účtence:

a) fiskální identifikační kód,
b) daňové identifikační číslo poplatníka,
c) označení provozovny, ve které je tržba uskutečněna,
d) označení pokladního zařízení, na kterém je tržba evidována,
e) pořadové číslo účtenky,
f) datum a čas přijetí tržby nebo vystavení účtenky, pokud je vystavena dříve,
g) celkovou částku tržby,
h) bezpečnostní kód poplatníka,

i) údaj, je-li tržba evidována v běžném nebo zjednodušeném režimu. (Česko, 2016)

3.2 UML

S rozvojem počítačových aplikací a IT obecně začala v devadesátých letech vznikat potřeba aplikace lépe plánovat. Nějak si je nakreslit. Stejně jako při stavbě budovy architekti naplánují její jednotlivé části, které poté vytvoří logický celek, tak i v oblasti IT muselo vzniknout podobné inženýrství.

V roce 1997 tedy přichází firma Object Management Group (OMG), která vydává Unified Modeling Language (UML). Vznikl sjednocením několika pokusů jednotlivých korporací něco podobného sestavit. Jeho účelem bylo vytvořit stabilní a běžný designerský jazyk, který bude moci být napříč organizacemi a kontinenty použit pro rozvoj a budování počítačových aplikací. Pro velký zájem o tento jazyk se UML udržela až dodneška. Organizace OMG spolupracuje s giganty jako IBM, Rational Software, Microsoft, Oracle. HP a další, a společně dohlíží na vývoj a specifikaci UML. (Bell, 2003)

Díky UML standardizaci můžou tedy týmy programátorů nebo architektů pracovat na jednom projektu zároveň. Doby, kdy aplikaci napsal jeden člověk, jsou totiž dávno pryč. Navíc není výjimkou, kdy se projekt změní za běhu, nebo si klient vzpomene na další funkčnost, kterou by aplikace měla splňovat. Díky UML můžeme pružně reagovat a také lépe sestavovat cenové odhady. I pro klienty zobrazuje zjednodušený návrh, v kterém se vyznají. (Čápka, 2013)

UML má tři základní použití:

Jako náčrt

Často ručně kreslené, zjednodušené návrhy aplikací pro upřesnění zadání s klientem. Přináší nám abstraktní pohled na věc.

Jako plán

Mnohem detailnější než náčrt, je to řada diagramů, které popíši níže. Slouží hlavně pro vývoj týmům programátorů. Ulehčuje i následnou implementaci.

Jako programovací jazyk

Z UML lze vygenerovat šablonu, kterou můžeme použít jako základ pro implementaci a vývoj. Nejčastěji používáno v databázích. (Čápka, 2013)

Diagramy, které jsem popisoval v části *UML jako plán,* se dělí na dvě základní části:

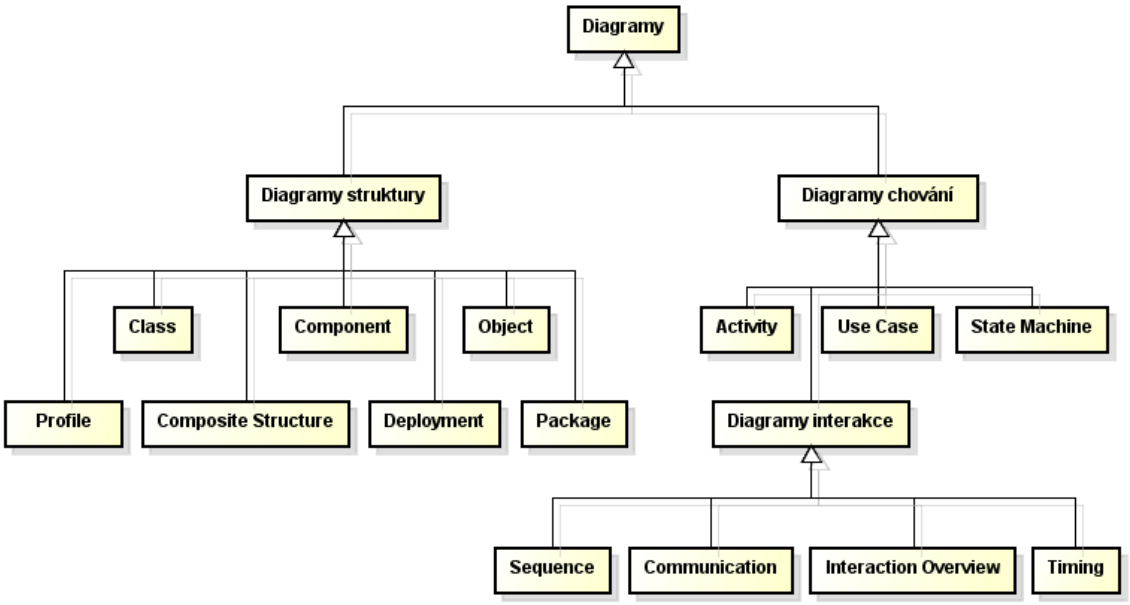
Diagramy struktury

Ukazují věci v systému, který modelujeme. V detailnějších diagramech zobrazují různé objekty.

Diagramy chování

Zobrazují, co by se stalo po nějaké interakci. Popisují, jak na sebe objekty vzájemně působí při vytváření fungujícího systému.

Obrázek 5 - Diagram UML diagramů



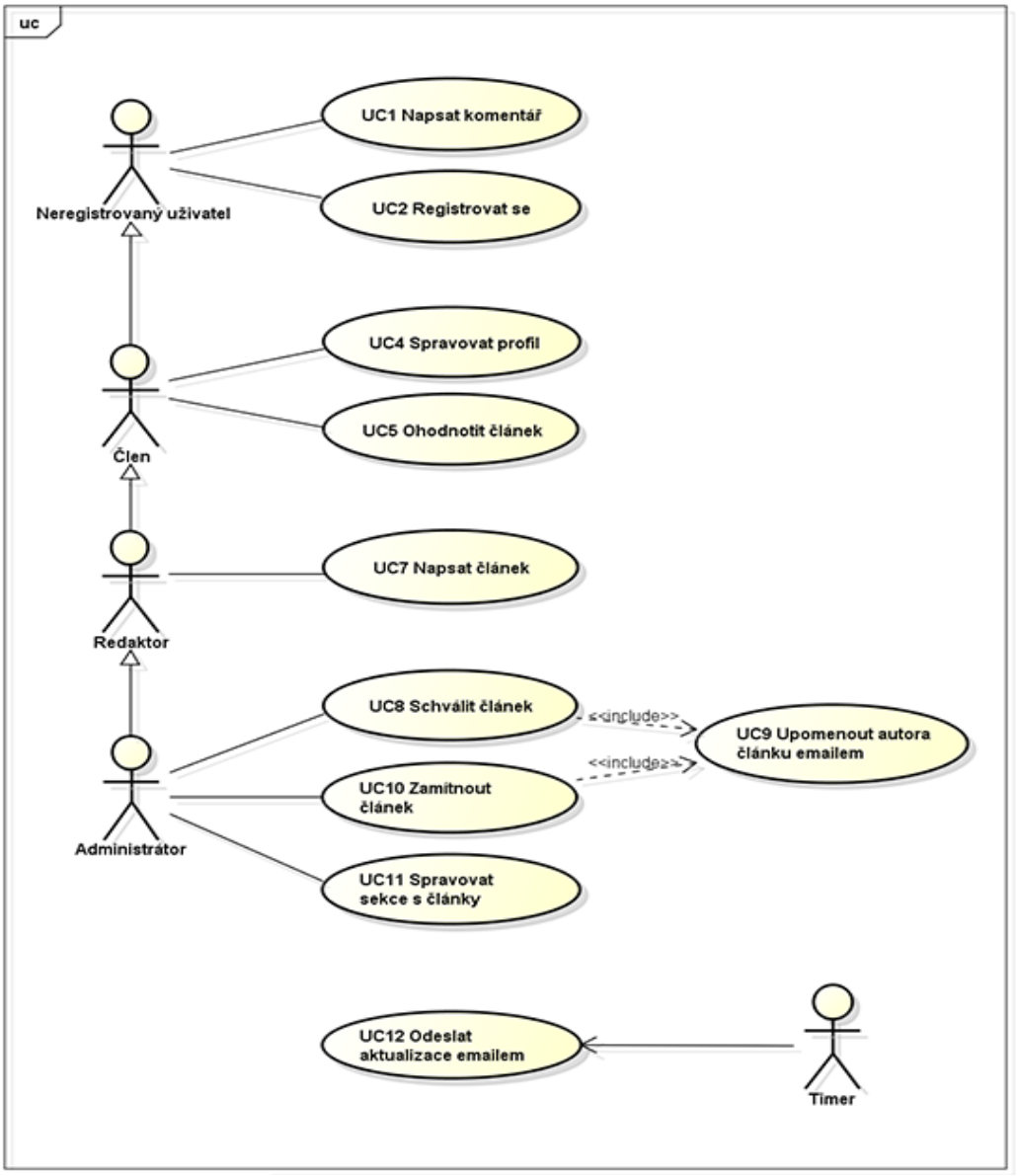
Zdroj: http://www.itnetwork.cz/navrhove-vzory/uml/uml-uvod-historie-vyznam-a-diagramy

Rozhodně není pravidlem, že pro každý projekt musíme vytvořit všechny diagramy. Vždy je třeba si říci, který nám pomůže a který je pro danou aplikaci potřeba.

3.2.1 Use Case Diagram

Česky diagram případů užití. Je nejznámějším a nejpoužívanějším typem z diagramů chování. Poskytuje provázaný grafický přehled subjektů a funkcí, které ony subjekty vykonávají. Je skvělým výchozím bodem projektu, protože si snadno identifikujete tzv. aktéry a hlavní procesy. (Silva, 2012)

Obrázek 6 - Ukázka digramu případů užití



Zdroj: http://www.itnetwork.cz/navrhove-vzory/uml/uml-use-case-diagram

Jak je vidět na obrázku, tento diagram má dva hlavní prvky. Aktéra (actors), což je na obrázku panáček, a případ užití (use case), který je znázorněn jako elipsa s názvem funkcionality uvnitř. Kromě těchto dvou prvků se používají ještě šipky, které znázorňují vztahy mezi aktéry a případy užití.

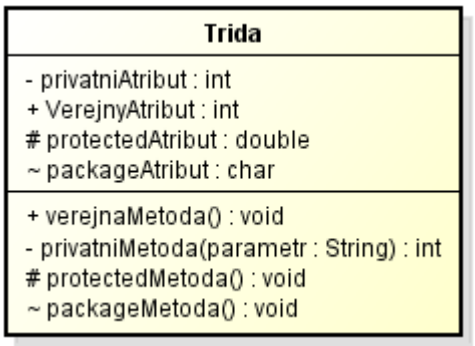
3.2.2 Class diagram

Diagram tříd, patří mezi nejpoužívanější z diagramů struktury. Je hlavním stavebním kamenem jakéhokoliv objektově orientovaného řešení. Ukazuje třídy, atributy, operace a vztahy mezi nimi. Různé vztahy ukazují odlišné typy šipek. (Silva, 2012)

Zobrazuje různé objekty (osoby, věci, data) a jejich vztahy z hlediska struktury systému, který vytváří. Digram tříd může být také použit pro zobrazení implementace třídy, což jsou věci, kterými se zabývají programátoři. Obecně platí, že ve větších systémech, aplikacích a hlavně korporacích vytváří tento diagram zkušený a drahý analytik a samotné programování, se poté může svěřit často o dost levnějšímu programátorovi, který s tím již nemá tolik práce. (Bell, 2003)

Ve většině modelovacích nástrojů má třída tři části. V horní nejdeme jméno třídy, v prostřední atributy a v nejnižší metody

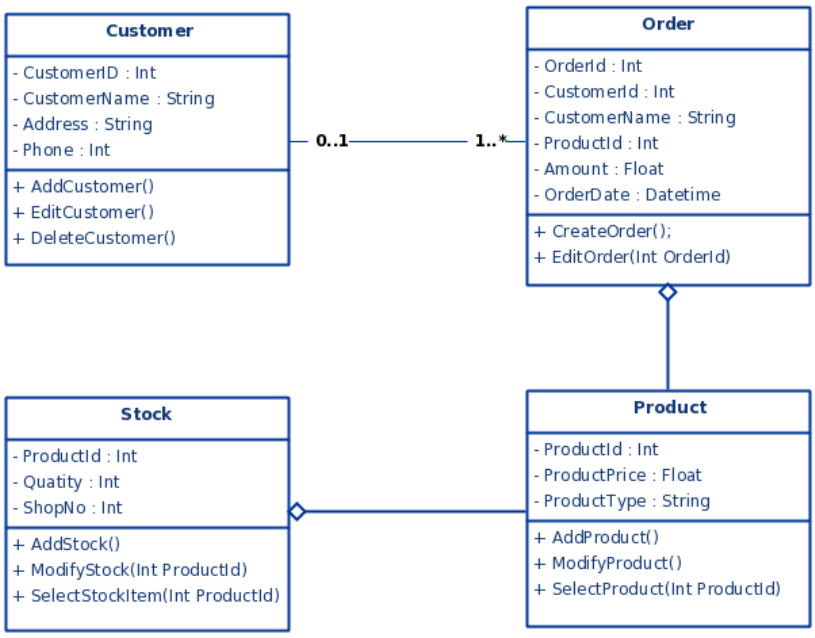
Obrázek 7 - Třída



Zdroj: http://www.itnetwork.cz/navrhove-vzory/uml/uml-class-diagram-tridni-model

Tyto třídy jsou seskupeny do velkých celků a pomocí zmiňovaných šipek, které reprezentují vztahy, utváří diagram tříd. Na obrázku 8 níže zobrazuji příklad digramu tříd, pro objednávku na e-shopu.

Obrázek 8 - Příklad diagramu tříd



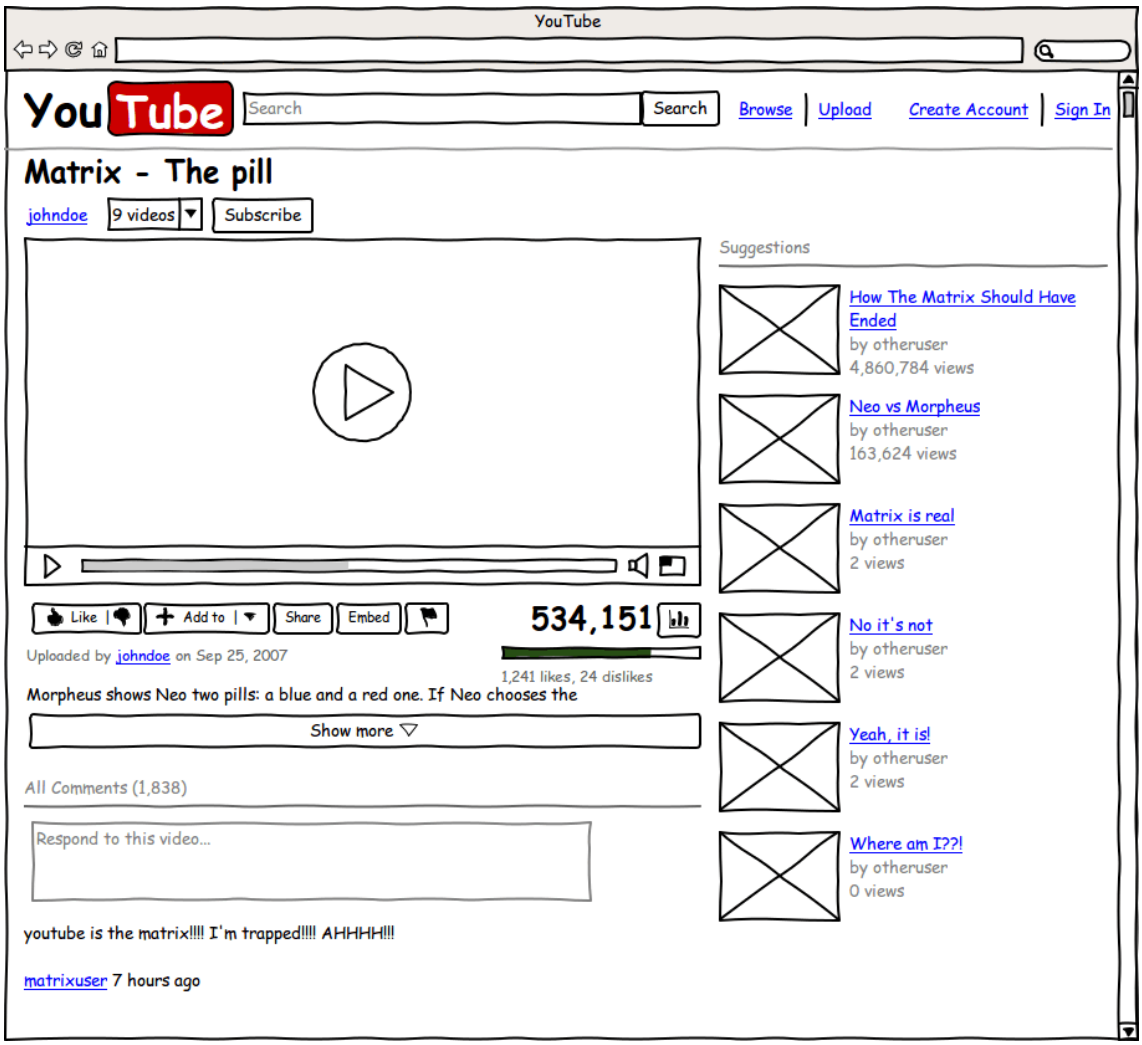
Zdroj: http://creately.com/blog/diagrams/uml-diagram-types-examples/#ClassDiagram

3.3 Wireframe

Česky drátěný model je důležitým krokem v každém procesu navrhování designové podoby výsledné aplikace. Především jde o hierarchické rozložení prvků na stránce tak, jak chcete, aby to uživatelé viděli. Navíc není zákazník ani navrhovatel rušen barvami (modely se dělají černobílé) ani volbou písma. Další výhodou je, že se při návrhu pohybujeme relativně svižně, často kopírujeme různé prvky, které značí například jen bloky textu v budoucnu. V návrhu je třeba pamatovat na to, že kupříkladu tlačítko musí být rozeznatelné i zde, aniž by mělo svítivé barvy nebo stíny. (Lim, 2012)

Tím se dostávám k tomu, že v této fázi se odráží i použitelnost webu, čímž je například fakt, že neumístíte nákupní košík doleva dolů, když ho většina obyvatel zeměkoule očekává vpravo nahoře.

Obrázek 9 - Ukázka wireframu



Zdroj: http://wireframesketcher.com

3.4 Frontend

Pojem vychází z oblasti programování webových aplikací. Pro každou aplikaci, stránku nebo software, můžou být hranice mezi pojmy frontend a co backend trošku rozdílné. V případě e-shopu rozumíme frontendem tu část webu, která je viditelná uživatelům. Například produkty, nákupní košík nebo objednávkový formulář. (Adaptic, 2016)

Jiným označením může být „klientská strana“, která obsahuje vše, co uživatel vidí – text, tlačítka, obrázky, navigace. Kdybyste chtěli vytvořit jednoduchou prezenční stránku s několika fotografiemi a trochou textu, tak vše, co potřebujete, jsou frontendové technologie. (Ferguson, 2016)

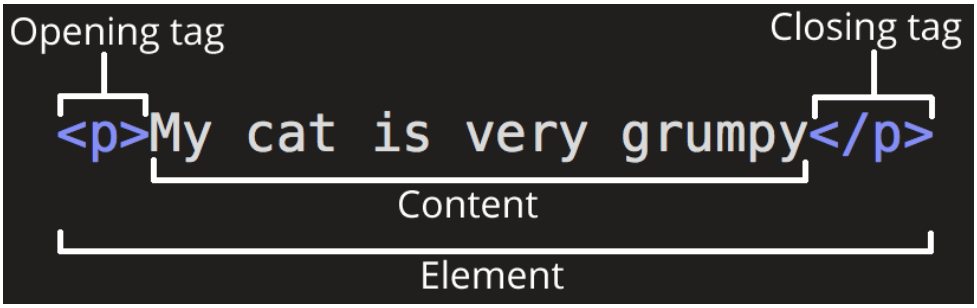
3.4.1 HTML

Vznikl v osmdesátých letech dvacátého století, když Tim Bernes-Lee narazil na problém, během jeho práce v CERNu. Jako vědec potřeboval přístup ke studiím svých kolegů, v té době vznikaly ale každý rok různé značkovací jazyky, jiné formáty a nekompatibilní rozhraní. Kvůli těmto problémům vznikala zpoždění, práce byla neefektivní a pomalá. Počítače tomu také moc nepomáhaly, byly náchylné k chybám a selhání. Proto se rozhodl vytvořit standard, který vytvořil revoluci v počítačové komunikaci. (Compute Staff, 2011)

Od té doby se HTML stalo základním kamenem každého webu, který definuje a popisuje obsah stránky. Kromě této webové technologie se používá ještě JavaScript, pro přidání funkcí a CSS, které nám na webu vykreslí grafiku.

Je důležité říci, že HTML není programovacím jazykem, je značkovacím jazykem. Jeho kompletní název v angličtině zní Hypertext Markup Langueage. Jazyk používá jednotlivé značky – anglicky tagy, které prohlížeči říkají, jak má zobrazit webové stránky. To proto, že každá značka definuje určitý obsah webu a každý se vykresluje jinak, nabývá jiných vlastností. (Mills, 2016)

Obrázek 10 - Element v HTML



Zdroj: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

Obrázek výše zobrazuje základní tzv. element. Skládá se tedy z tagu `<p>`, konkrétně z otevíracího (Opening tag) a uzavíracího (Closing tag), který značí odstavec, a obsahu (Content), který chceme zobrazit.

Obrázek 11 - Atribut

Zdroj: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

Elementy můžou mít také tzv. atributy, ty obsahují určité extra informace, které je třeba zobrazit nebo zmínit v aktuálním kontextu. V tomto případě vidíme `class`, to nám umožňuje stylovat všechny atributy s touto hodnotou, stejným způsobem. Atributy musí být oddělovány mezerou, za každým následuje rovnítko a hodnota uzavřená v uvozovkách z obou stran.

Základní struktura dokumentu vypadá pro představu takto:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Titulek
        stránky</title>
    </head>
    <body>
        <h1>Nadpis</h1>
        <p>Odstavec</p>
    </body>
```

- `<!DOCTYPE html>` - říká prohlížeči, že se jedná o dokument v HTML5
- `<html>` - základní tag, který ohraničuje stránku
- `<head>` - obsahuje hlavní, tzv. meta, informace, připojují se zde CSS styly nebo skripty

- `<title>` - element určující název dokumentu
- `<body>` - v tomto elementu se nachází viditelná část stránky
- `<h1>` - nejvyšší a největší úroveň nadpisu
- `<p>` - element definující odstavec

3.4.2 CSS

Cascading Style Sheets (CSS) je jazyk používaný k určení vzhledu dokumentu napsaného v HTML nebo XML. Popisuje, jak by měly být prvky vykresleny na obrazovce, papíru, v řeči nebo jiných médiích. Ušetří spoustu práce, protože může definovat několik prvků v různých webových stránkách najednou. Je jedním z hlavních jazyků internetu a je standardizován organizací W3C. Poslední verzí je CSS3. (Mozilla Developer Network, 2017)

CSS můžeme do HTML přidat třemi způsoby:

Inline

Zápis atributu přímo do elementu. Vlastnosti stylu budou působit pouze na daný element. Tento příklad obarví text na modro.

```
<h1 style="color:brown;">Tento text bude hnědý</h1>
```

Interně

Definuje se v `<head>` HTML stránky jako element `<style>`. Používá se pro stylování dané stránky. Všechny nadpisy by v tomto případě byly modré a odstavce červené.

```
<head>
     <style>
          h1   {color: blue;}
          p    {color: red;}
     </style>
</head>
```

Externě

Používá se externí CSS soubor a je využíváno hlavně pro stylování více stránek. Touto technikou můžete změnit vzhled celého webu tím, že změníte jeden soubor! Samotný soubor už poté můžete napsat v jakémkoliv textovém editoru a zakončit ho příponou CSS. Soubor stačí následně provázat v hlavičce. (w3schools, 2015)

```
<head>
     <link rel="stylesheet" href="styles.css">
</head>
```

3.5 Backend

Oproti frontendu je to část webu, která není přímo viditelná uživatelem e-shopu. Slouží tedy pro administraci a zpracování dat, nejčastěji editace produktů, jejich vlastností a cen. Vychází z anglického back-end, tedy něco jako zadní část. (Adaptic, 2016)

Jestliže nemáme výše zmiňované prezenční stránky a očekáváme od webu větší tok dat, který se například vypisuje dle interakce uživatele, budeme potřebovat do hlubšího zákulisí webových stránek. Backend je zodpovědný za ukládání a uspořádávání dat. Backend komunikuje s frontendem a posílá a přijímá informace, které mají být zobrazeny. Příkladem může být odeslání objednávky, ta se zaznamená na straně serveru a uloží do administrace provozovatelům e-shopu. Na frontendu se poté promítne změna stránky, například poděkování za nákup, nebo shrnutí objednávky. (Ferguson, 2016)

3.5.1 PHP 7.1

Je široce používaný open-source skriptovací jazyk, který je obzvláště vhodný pro vývoj webových aplikací nebo dynamických webových stránek. Původně byl vytvořen Rasmusem Lerdorfem v roce 1994. Zkratka názvu vznikla rekurzivně z Hypertex Preprocessor. (php.net, 2017)

PHP kód může být vložen do HTML kódu, nebo ho lze použít v kombinaci s různými webovými šablonami, systémy pro správu obsahu, a webovými frameworky. Většinou bývá zpracováván PHP interpretem, který je implementován jako modul na webovém serveru nebo jako spustitelný soubor Common Gateway Interface (CGI), což je protokol pro propojení externích aplikací s webovým serverem. Webový server kombinuje

výsledky interpretace a prováděného PHP kód, což může být jakýkoliv typ dat, včetně obrázků, které generuje webová stránka. PHP kód může být také proveden pomocí rozhraní příkazové řádky (CLI) a může být použit pro implementaci samotné grafické aplikace. (php.net, 2017)

Standardní PHP interpretace, poháněna Zend Enginem, což je skriptovací engine (virtuální stroj), je svobodně šiřitelný software pod licencí PHP. Díky své široké rozšířeností může být nasazen na většině webových serverů a téměř na každém operačním systému nebo platformě zdarma! (Coggeshall, 2001)

Zajímavostí je, že jazyk PHP se vyvíjel bez písemné formální specifikace nebo normy až do roku 2014, kdy se finální podoby specifikace konečně dočkal. (Jackson, 2014)

Poslední verzí je 7.1, která napravila chyby, na které se zapomnělo ve verzi 7, která přinesla řadu novinek. I přesto je stále nejrozšířenější a nejpoužívanější verze PHP5, která má dle březnových statistik roku 2017 stále 95.5 % podíl na stránkách používajících PHP, oproti tomu se zdá 3.5 %, která teď má PHP7, málo. (w3Techs, 2017)

Je důležité myslet na to, že spousty aplikací mohou být již neaktivních, a na webech se přesto vyskytují. Časem situace vykrystalizuje, moderní firmy budou používat PHP7 a situace se otočí.

3.5.2 Framework Laravel

Framework je pomocná softwarová struktura, která ulehčuje programování. Nabízí knihovny často používaných funkcí, API nebo podpůrné programy. V současné době je nejpoužívanějším volně šiřitelný (open-source) Laravel, pod licencí MIT. Vytvořil ho Taylor Otwell začátkem roku 2012 pro vývoj webových aplikací v návaznosti na model MVC (Model-View-Controller). Jeho jádro z velké části vychází z frameworku Symfony2. Spousta funkcí je modulárních, najdeme tam různé způsoby přístupů do databází, nástroje, které napomáhají k nasazení aplikací a jejich údržbu. Poslední zmiňovaná je jednou z hlavních výhod a je proto oblíbená. (Bean, 2015)

Tvůrci říkají, že Laravel je webový aplikační framework s výraznou a elegantní syntaxí. Je elegantní, inovativní, přístupný, výkonný, ale zároveň poskytuje nástroje pro velké aplikace. Pomocí jeho nástrojů sestavíte libovolné aplikace. Tvůrci se pokoušejí odstranit nepříjemnosti vývoje a usnadnit běžné úkoly používané ve většině webových projektů, například:

- podpora více databází (MySQL, PostgreSQL, atd.)
- podpora více systémů pro ukládání do mezipaměti
- vynikající IoC container
- robustní základ pro psaní jednotkových a integračních testů

Navíc, nemáte-li rádi čtení, poskytují více než 900 video návodů na širokou škálu témat. (Laravel, 2017)

Hlavními funkcionalitami, kterou jsou použity v mém EET řešení, a tedy rapidně urychlily jakýkoliv vývoj, jsou správa uživatelů (autentizace a autorizace) a podpora OAuth 2.0, což je standardizovaný protokol pro autorizaci. Zaměřuje se na jednoduchost pro vývojáře a zároveň poskytuje zvláštní povolení toků pro webové, desktopové i mobilní aplikace. Tato specifikace je vyvíjena v rámci IETF OAuth WG. (OAuth, 2012)

Zmíněné OAuth 2.0 je použito pro komunikaci aplikace s e-shopem klienta/zákazníka. Ten si v aplikaci vygeneruje token, který si umístí k sobě na server a pomocí něho probíhá následná komunikace. Za bezpečnost na serveru klient samozřejmě zodpovídá sám. Ztrátou nebo odcizením tokenu by mohlo dojít například k podvrhnutí částek posílaných na Finanční úřad a tím k navýšení daní.

3.5.3 Databáze PostgreSQL

Zkráceně Postgres je výkonná volně šiřitelná (open-source) objektově-relační databáze, která má za sebou více než patnáct let aktivního vývoje a osvědčené architektury, se silnou pověstí pro svou spolehlivost a integritu dat. Funguje na všech hlavních operačních systémech jako Linux, UNIX, Windows. Je plně ACID kompatibilní a má plnou podporu cizích klíčů, pohledů, triggerů a procedur v několika jazycích. Zahrnuje většinu SQL datových typů. Také podporuje ukládání binárních velkých objektů včetně obrázků, zvuků a videa. Její součástí je samozřejmě výjimečná dokumentace. (PostgreSQL, 2017)

PostgreSQL se může pochlubit sofistikovanými funkcemi jako MVCC, bod v čase obnovy, asynchronní repliky, vnořené transakce, online/offline zálohami nebo plánovač dotazů. Podporuje mezinárodní znakové sady, velká i malá písmena a formátování. Je vysoce škálovatelná a data mohou být tříděna i při současné práci více uživatelů. (PostgreSQL, 2017)

3.6 E-shop Tisknisi.cz

Vznikl jako sesterská firma tiskárny dot. DesignStudio s.r.o., která, jak název vypovídá, zajišťuje i práce v grafickém studiu. E-shop je funkční cca 4 roky, po které stále roste. Jeho hlavní činností je tisk vizitek. Okrajově se zapojují i další tiskoviny, aby se sběr zakázek pro tiskárnu ještě více rozšířil. Růst a chyby v začátcích donutily majitele k vývoji nyní již třetí verze e-shopu, každou vždy vyvíjela jiná firma. Nicméně právě vyvíjená třetí verze bude již myslet na zapojení další tiskovin, a tedy rozšíření obzoru a posunu na poli e-shopů nabízejících tyto komplexní služby. Já osobně jsem v e-shopu zapojen dva roky, patříme v současné době mezi třemi nejvýše postavené firmy v oboru v rámci ČR ve vyhledávači Google na klíčová slova spjatá s online tiskem. Podobných hodnot dosahujeme i v domácím Seznamu.

Co se týká frontendu, web používá HTML, CSS a Javascript. Většina již byla popsána v sekci Frontend

3.6.1 Foxxy CMS²

Je proprietární redakční systém psaný v PHP, na kterém e-shop běží. Zajišťuje hlavně administrativní část, která je modulární. Hlavními moduly jsou seznam objednávek, správa produktů, uživatelů a faktur. Modulů je samozřejmě více a upravují se dle požadavků daného e-shopu. Bohužel společnost Inizio s.r.o., která e-shop nasazovala na jimi vyvinuté prostředí, tuto situaci trošku podcenila a neuvědomila si specifičnost daného odvětví. Řada modulů nebyla pro tiskařské produkty připravená, a tak se hodně předělávaly, tím bohužel ale vznikaly často chyby a velké náklady na dodatečné úpravy a vícepráce. Navíc není systém aktualizovaný, je zastaralý se spoustou chyb, které pomalu vyplývají na povrch. Od toho se odvíjí i problém, že bez větších zásahů je téměř nemožné rozběhnout ho pod PHP7.

I když Inizio tvrdí, že běží na Nette, z kvalitního zdroje se ke mně dostalo, že framework bohužel nepoužívá žádný. Systém je psaný, jak říkají programátoři, v tzv. špagety kódu, což znamená, že soubory mají i tisíce řádků, které nejsou rozděleny do logických celků.

Používá databázi MySQL, která je relační, typu DBMS a vychází z jazyka SQL. Je to jedna z mála kvalitních věcí v tomto redakčním systému.

4 Metodika

Nejprve bylo nutné se v teoretické části seznámit s problematikou jak elektronické evidence tržeb, tak s technologiemi a postupy, jakými se moderní aplikace vyvíjí. Jelikož je téma velmi moderní, většina zdrojů, z kterých jsem čerpal informace a znalosti, pochází z internetu. Tam jsem vycházel pouze z ověřených zdrojů, nejčastěji stránek přímo firem zastřešujících danou technologii anebo institucí, které mají danou problematiku na starosti. Mluvím zde hlavně o webu Finanční správy ČR a webu etržby, který je primárním zdrojem informací pro elektronickou evidenci tržeb.

Po plném pochopení dané pragmatiky je možné se přesunout k praktické části neboli tvorbě aplikace E-EET. Nejdříve si samotnou aplikaci vysvětlíme a popíšeme její fungování. Pro pozdější jednodušší programování si aplikaci navrhneme pomocí dvou diagramů. Oba nám, stručně řečeno, pomůžou pochopit zapojené subjekty a funkce, které budeme potřebovat. Abychom měli představu o vizuální podobě, použijeme drátěné modely pro uspořádaní prvků na stránce. Následně si ukážeme technické řešení aplikace a její nejdůležitější části. Pro pochopení finální jednoduchosti nebude samozřejmě chybět ukázka implementace E-EET do e-shopu a srovnání s náročností v případě implementace řešení vlastního, které si bude chtít majitel e-shopu, či najatý programátor naprogramovat sám. V závěru navrhnu pro ziskovat aplikace cenovou politiku.

5 Praktická část

Jako správce e-shopu jsem byl v polovině roku 2016 nucen začít řešit situaci ohledně novinky jménem Elektronická evidence tržeb. Zatímco pokladní řešení se začínala objevovat jako houby po dešti, řešení pro e-shopy stále nepřicházelo. Bohužel se navíc na přelomu roku 2016-2017 objevila informace, že Ministerstvo financí situaci podcenilo a neuvědomilo si množství plateb, které jsou skrze e-shop prováděny. Díky lobování AMSP ČR si také začalo ministerstvo uvědomovat, že většina plateb je už elektronicky dohledatelná, takže nějaké úniky nepřicházely téměř v úvahu. V návaznosti na to navrhl pan Ing. Babiš výjimku, do které měly e-shopy zapadat. Troufám si říci, že i z důvodu populismu bohužel tento návrh neprošel. To se stalo cca před měsícem před ostrým spuštěním, které bylo naplánováno na 1. března 2017 (tedy 2. fáze EET). Znamenalo to, že platí původní návrh a evidovány budou všechny platby, kromě přímého převodu z bankovního účtu zákazníka na účet poplatníka (tedy majitele e-shopu). Dalších kosmetických úprav se dočkalo definování, kdy přesně je povinen poplatník platbu evidovat. Jestli je jeho maloobchod minoritním a spadá do této fáze, nebo může začít evidovat až v pozdější fázi, aj.

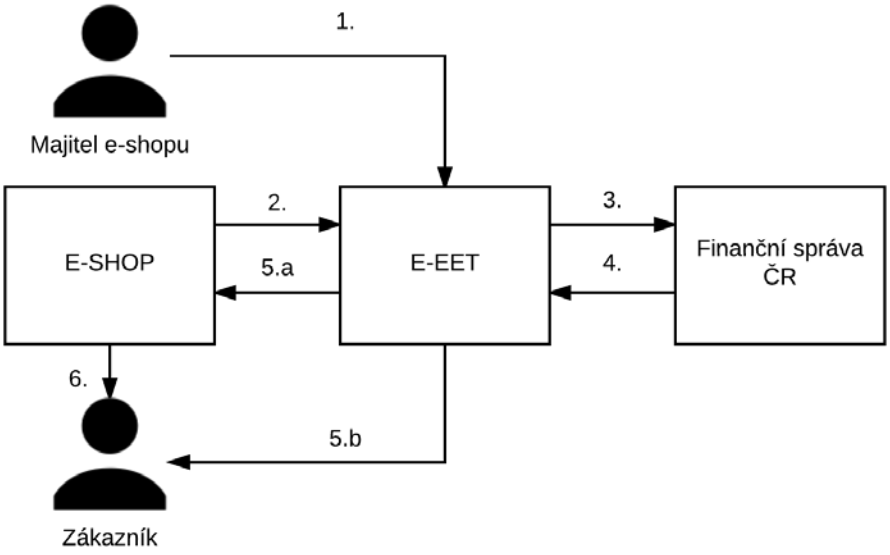
5.1 Popis aplikace

Naštěstí jsem předpokládal, že žádné změny v zákoně neprojdou, a tak jsem se pustil do vývoje aplikace již dříve. Vše zprovoznit poslední měsíc bych opravdu nechtěl a nezávidím větším korporacím, které musely řešení narychlo připravit. Mezi ně patří například GoPay, které řešení připravilo ve sprintu asi třech týdnů. Jisté neduhy ale bohužel řešilo ještě po spuštění.

Jelikož jsem se o EET zajímal již dříve, věděl jsem, že nasazení na e-shop nebude jednoduchá záležitost. Zároveň jsem si říkal, že existují jisté implementace do e-shopů, které jsou pro programátory jednoduché a většina logiky se zpracuje na straně serveru. Uviděl jsem jistý potenciál v řešení, které by mohlo hlásat, že zkracuje dobu implementace o 90 % a snižuje tudíž náklady na zavedení. Cena dobrých programátorů se totiž může pohybovat od 500-1000 Kč za hodinu, a proto majitelé a správci e-shopů ocení každé zkrácení těchto hodin.

Nuže po několika sezeních s kolegy z oboru, kde jsme řešili hlavně otázku zabezpečení posílaných dat, jsem navrhl řešení, které postupem času dostalo název **E-EET**. Pro lepší pochopení vymyšleného řešení přikládám schéma:

Obrázek 12 - Princip E-EET



Zdroj: vlastní zpracování

Jednotlivé kroky vypadají takto:

1. Jisté údaje se používají stále stejné a opakovaně, tak proč zatěžovat servery a neustále je posílat z e-shopu. Proto prvotní nastavení, provede majitel e-shopu v aplikaci, kde nastaví ve svém profilu povinné údaje, které jsou fixní pro každou zasílanou tržbu. Jsou jimi DIČ poplatníka, označení provozovny a označení pokladního zařízení. V této fázi také nahraje certifikát, který je nutný si vygenerovat v aplikaci EET, kde také získá fixní údaje, které popisuji v minulé větě.
2. Po implementaci řešení do e-shopu, je tedy princip následující. Po přijetí objednávky se zašlou údaje, které jsou nutné jak k evidenci, tak pro výpočet kódů na straně aplikace E-EET a k tomu, aby Finanční správa ve své aplikaci poznala, o koho se jedná a že se nejedná o podvrh. Údaje zde zmíněné se liší dle provozovny, jak jsem popisoval v části Struktura datových zpráv. Pro tisknisi.cz platí, že posílá dle tabulky 1 položky č. 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 25, 26 a 27. Pro rozšíření služby E-EET se počítá, že zasílat z e-shopu lze i e-mail zákazníka, který by se použil v případě 5.b.

3. Po vypočtení kódů a zabalení zprávy pošle aplikace E-EET evidovanou tržbu na servery Finanční správy.
4. Ta evidovanou tržbu zaznamená, zkontroluje a pošle zpět buď chybovou zprávu, anebo potvrzení o přijetí tržby. Budeme předpokládat, že potvrzení. Hlavně nám vrací FIK (fiskální identifikační kód).
5. Nyní můžou existovat dvě cesty:
    a. FIK se zašle zpět na e-shop, kde už je na programátorovi, jak s ním bude nakládat. Dále se pokračuje krokem 6.
    b. E-shop nám poskytl e-mailovou adresu a zaplatil si službu za zaslání účtenky, a tak posílá email s účtenkou zákazníkovi rovnou aplikace E-EET. Ubírá se tím další náklad ve smyslu hodin programátora, který musí připravit elektronickou podobu účtenky, nastavit email a nechávat ho zasílat. V tom případě zde koloběh končí.
6. Povinností poplatníka je, že musí vystavit účtenku. Jestli elektronickou nebo tisknutou je na něm. E-shopy budou ve většině případů volit elektronickou, tedy je nutné nějakým způsobem v nějakém souboru zaslat zákazníkovi na jeho e-email účtenku s povinnými údaji, o kterých jsem již mluvil v sekci 2.1.2.4 Údaje uváděné na účtence. Zasláním e-mailu s účtenkou koloběh končí.

Rád bych závěrem této sekce zmínil ještě jednu podstatnou věc. V následující kapitole vysvětluji výhody služby GitHub[1], to samozřejmě proces vývoje E-EET značně ulehčilo. Z toho plyne, že stejně tak mohl tento repozitář použít programátor a moje teorie o tom, kolik času šetřím při implementaci padá stejně tak jako ekonomický nebo podnikatelský záměr, protože se ta nejsložitější část nemusí programovat znovu. Ale jako člověk, který se pohybuje jako IT projektový manager mezi dvěma břehy, dobře vím, že programátor by zřejmě neprozradil majiteli e-shopu, že nějaký repozitář vůbec existuje, že ho použil a ušetřil 80 % svého času. Hezky by naúčtoval částku, jako kdyby to programoval od nuly. Majitel e-shopu do problematiky samozřejmě vůbec nevidí, a tak to nemůže nikdy zjistit. Takže ano, opravdu šetřím hodiny na programátora, protože na implementaci jednoduchého příkazu s několika řádky si hodiny vymyslet nemůže. Tím nechci říkat, že jsou programátoři špatní a přidávají si hodiny. Jen jsem na toto nepochopení narazil při diskuzích s kolegy v oboru, kteří stále nechápali, jak ušetřím programátorovi čas, když to může udělat jako já a mít to za stejnou dobu. Až po dlouhé diskuzi mým kolegům a

[1] Webová služba pro vývoj softwaru, s verzovacím nástroje Git.

programátorům došlo, že se nejedná o šetření času programátorovi ale nákladů majiteli e-shopu.

5.2 Návrh aplikace

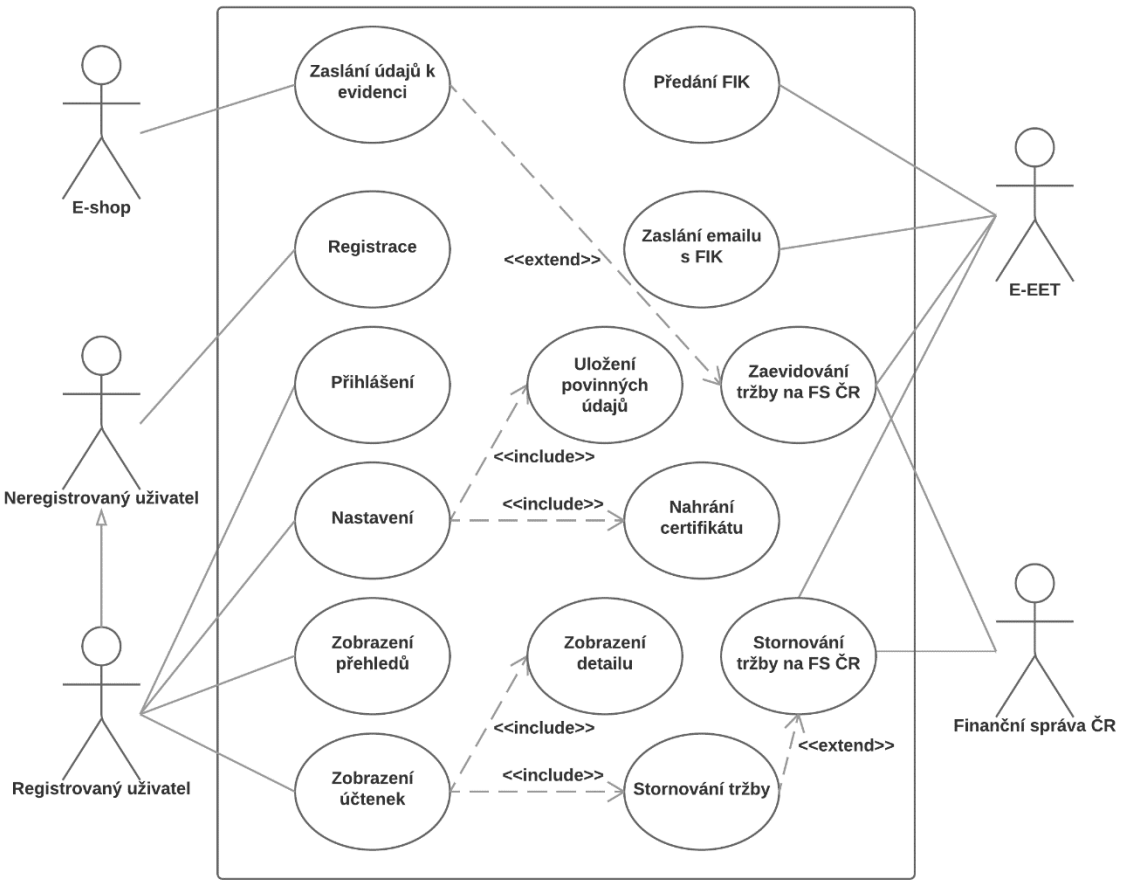
Pomocí základních modelovacích nástrojů jako UML a Wireframe zobrazím potřebné funkcionality a vzhled.

5.2.1 UML

5.2.1.1 Use Case Diagram

V diagramu jsem použil pět aktérů. Dva reprezentují skutečné osoby (neregistrovaný uživatel a registrovaný uživatel) a zbylé tři servery (e-shop, E-EET a Finanční správa ČR).

Obrázek 13 - Use Case Diagram



Zdroj: vlastní zpracování

Na obrázku se nám vyskytuje několik vazeb:

1. Čistá plná čára značí určitou asociaci a spojení s případem užití neboli funkcionalitou.

2. Šipku s dutým koncem nazýváme generalizace a značí, že aktér, z kterého šipka vychází, dědí případy užití aktéra, na kterého šipka směřuje. V našem případě se neregistrovaný uživatel může pouze registrovat, zatímco registrovaný může samozřejmě spravovat svůj profil, ale taktéž se i znovu registrovat jako jiný uživatel. V praxi se to asi dít nebude, ale tuhle možnost by měl mít.
3. Přerušovaná šipka značí, že je nějaká funkcionalita důležitá, a tak stojí za zmínění samostatně, místo prohlášení, že je součástí jiné.
    a. <<include>> značí hlavně zmíněné rozšíření nějaké další funkcionality. Na kterou funkcionalitu šipka ukazuje, ta je součástí funkcionality, z které vychází. V našem případě je v nastavení možnost nahrát certifikát, ale v sekci nahrávání certifikátu se nic nenastavuje.
    b. <<extend>> se používá v případech, kdy jedna funkce vyvolá jiné funkce, neboli následuje nějaký další krok. V našem příkladu jde například o to, že uživatel zvolí stornování účtenky/tržby a aplikace E-EET vytvoří stornovací zprávu.

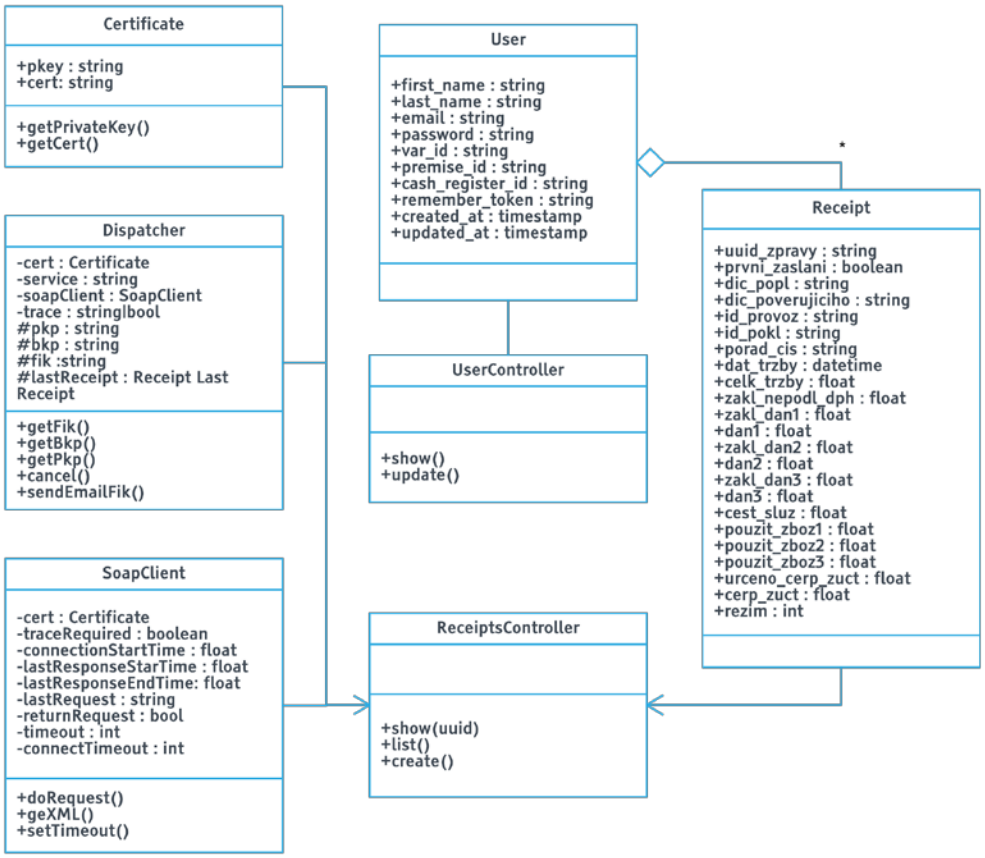
Věřím, že zbylé případy užití nepotřebují hlubší popis a z jejich názvů jasně vyplývají jejich funkcionality.

5.2.1.2 Class diagram

Aby byl diagram tříd plně pochopen, je zde nutné zmínit, že jádro E-EET je převzaté z GitHubu. V dnešní době je totiž řada řešení různých problematik již naprogramována a jelikož postrádá smysl je vytvářet stále znova, existují úložiště, kde programátoři sdílí své části kódu (viz zmíněný GitHub). Chtějí-li, tak pod otevřenou licencí MIT, což je i jádro, o kterém se zmiňuji. Pro aplikaci je nakonec vybrán repozitář od Filipa Šedivého s názvem PHP-EET[2], který nejlépe splňoval funkcionality a architekturu, které byly potřeba.

[2] Kompletně k nastudování zde: https://github.com/filipsedivy/PHP-EET

Obrázek 14 - Class Diagram



Zdroj: vlastní zpracování

Prostřední sloupec reprezentuje třídy aplikace E-EET a ostatní jsou převzaty z repozitáře pana Šedivého.

Tabulka **uživatel (User)** má v poli atributů informace o registrovaném uživateli, tedy majiteli e-shopu, který vyplňuje povinné údaje, o kterých jsem již psal.

UserController provádí operace s uživatelem, takže hlavně uložení jeho údajů a výpis.

ReceiptsController shromažďuje informace ze tříd repozitáře a dále s nimi pracuje. Vidíme metodu `create()`, která vytvoří záznam do databáze o každé účtence, která je posílána do evidence. Je to z toho důvodu, aby účtenky poté měl registrovaný uživatel k dispozici. Dále `list()`, která se používá pro výpis účtenek.

„**Certificate** slouží pro práci s certifikátem. Mimo jiné preparuje klíč p12 na privátní a veřejný klíč, který se dále používá v kódu.“ (Šedivý, 2017)

„**Dispatcher** slouží pro skládání požadavků a komunikuje se SoapClientem.“ (Šedivý, 2017)

„**SoapClient** slouží pro odesílání a přijímání soap požadavků.“ (Šedivý, 2017)

„**Receipt** je třída, která sdružuje veškeré vlastnosti příručky EET.“ (Šedivý, 2017)

Většina vazeb je 1:1 z důvodů přebírání a také toho, že jedna tržba rovná se jeden záznam. Rozdílem je vazba uživatel (User) a účtenka (Receipt), kde uživatel může mít žádnou nebo nekonečno účtenek, zatím co daná účtenka má právě jednoho uživatele (musí být totiž vždy jedinečná).

Vztah, který stojí za povšimnutí, je agregace opět mezi třídami uživatel a účtenka. Značí vztah celek a nějaká jeho část. Kde kosočtverec umístíme k třídě, která reprezentuje celek (Označované také jako majitel.). Rozdíl mezi vyplněným a dutým kosočtvercem je ten, že plný značí pevnější vazbu, jeden bez druhého často nedávají smysl, při zániku majitele zaniká i druhý. Dutý, což je náš případ, značí, že vazba není tak silná a zánik majitele nezapříčiní zánik odkazovaného objektu.

Klasická šipka z krajních tříd (Certificate, Dispatcher, SoapClient, Recepit) směřuje na **ReceiptsController**, což může číst například jakože certifikát (Certificate) spadá pod **ReceiptsController**, jinak řečeno, že ten s nimi pracuje.

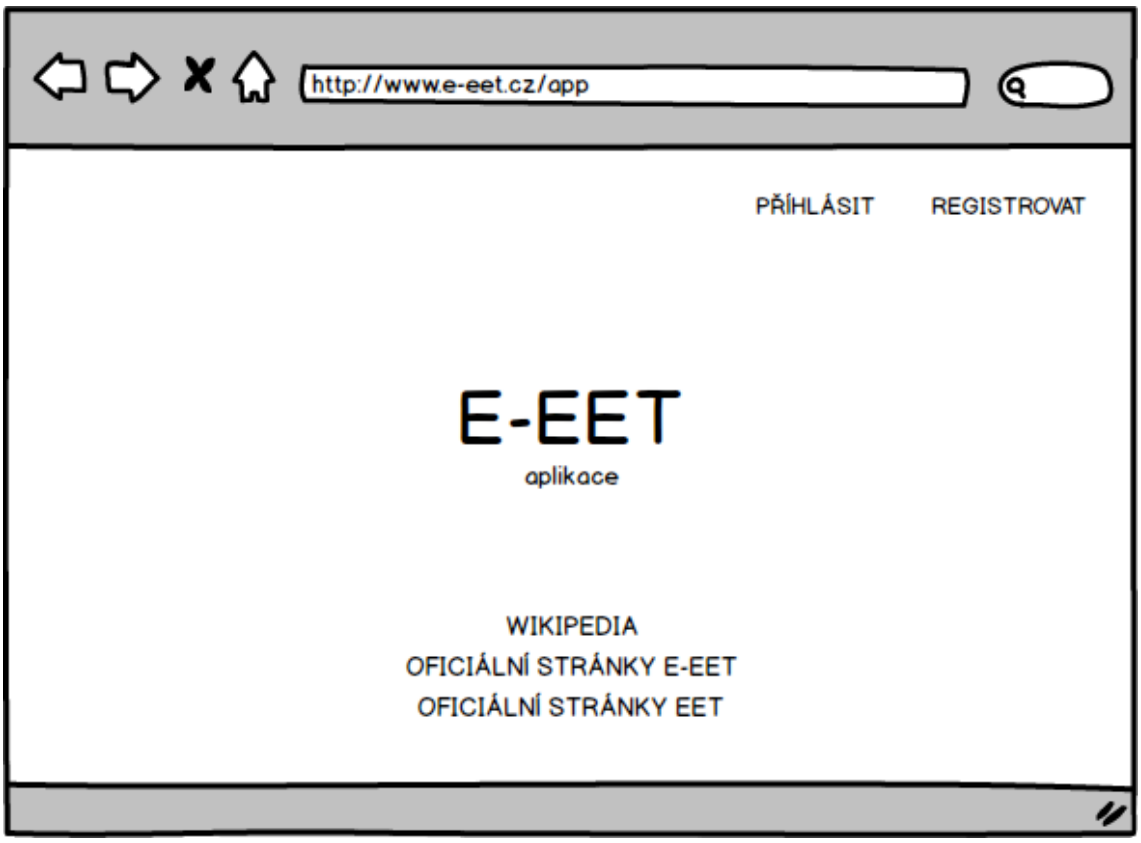
5.2.2 Wireframe

Po několika papírových verzích jsem měl jasnou představu, jak by měla aplikace vypadat. Rozhodl jsem se držet čistý a minimalistický vzhled napříč celou aplikací.

5.2.2.1 Úvodní stránka

Pro úvodní stránku po vstupu do aplikace E-EET jsem chtěl zvolit velice jednoduchý design, bez zbytečných grafických prvků. Výsledkem je tedy Obrázek 15 - Úvodní stránka aplikace.

Obrázek 15 - Úvodní stránka aplikace



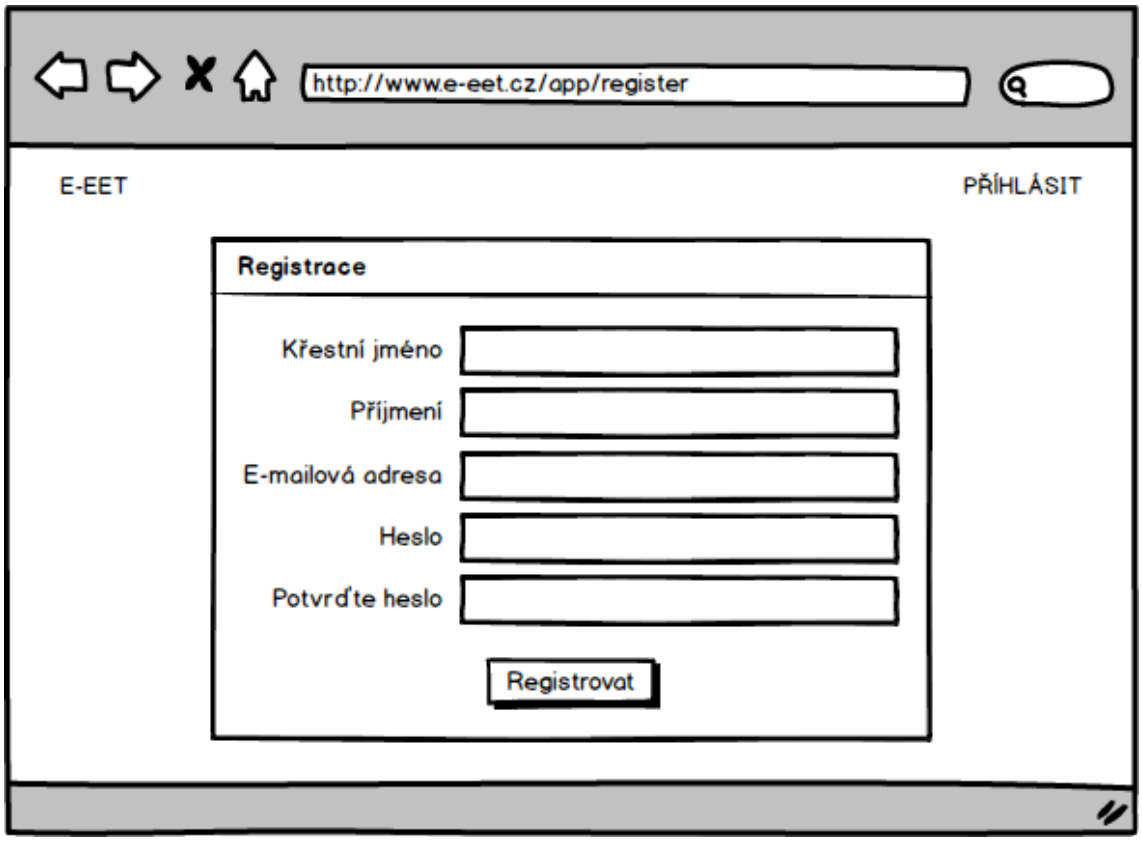
Zdroj: vlastní zpracování

V centru můžeme vidět nadpis v podobně názvu aplikace a podnadpis, který říká, že se jedná o aplikaci, ne prezenční webové stránky. Vpravo nahoře vidíme dva odkazy, kde jeden vybízí k registraci a další k přihlášení. Odkazy dole na stránce směřují na wikipedii ohledně aplikace, na oficiální prezenční stránky a na oficiální stránky Finanční správy pro elektronickou evidenci tržeb.

5.2.2.2 Registrace

Pro zajímavost je níže vidět stránka pro registraci, kde se vyplní opravdu jen základní údaje.

Obrázek 16 - Registrace

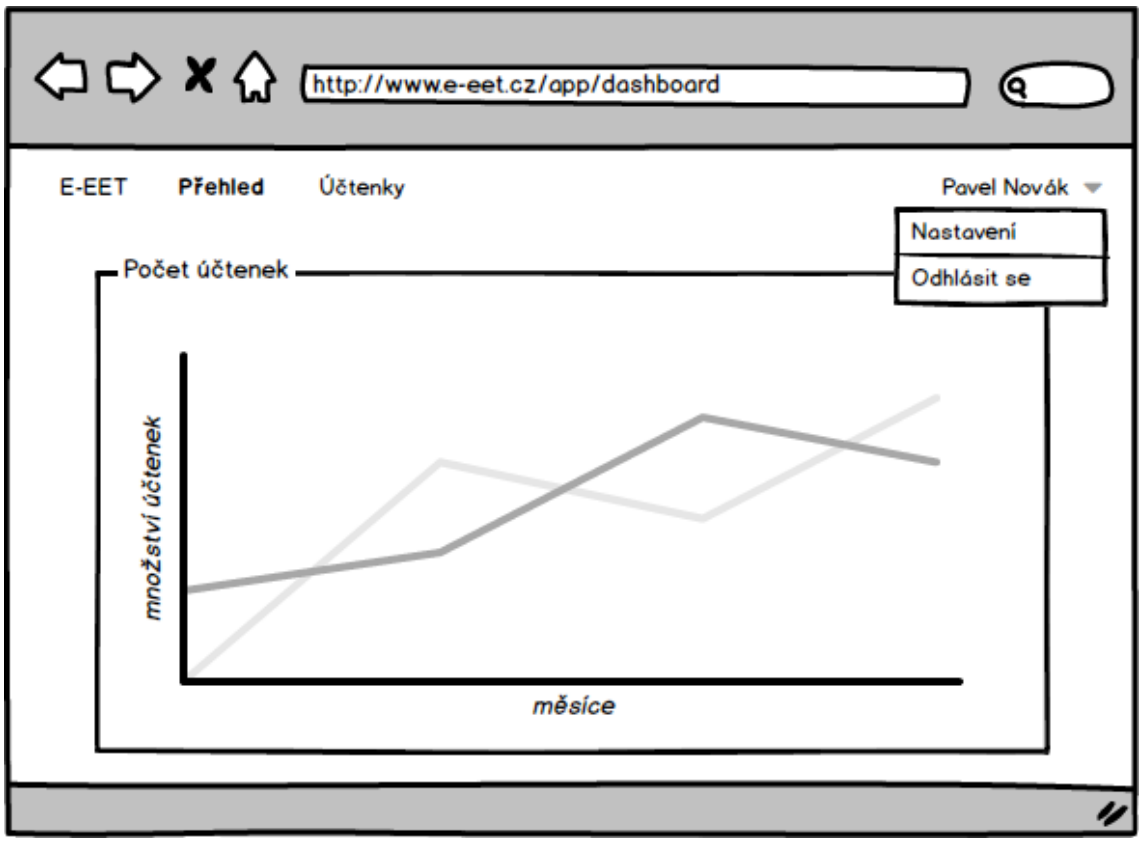


Zdroj: vlastní zpracování

5.2.2.3 Přehled

Po přihlášení uvidíme vizualizaci vydaných účtenek, jak je patrné z wireframu na Obrázek 17.

Obrázek 17 - Přehled účtenek



Zdroj: vlastní zpracování

Po registraci, případně přihlášení, se dostaneme na *Přehled*, kde můžeme vidět, kolik účtenek bylo vydáno. Na ose **x** bude uvedený časový rámec a na ose **y** množství vydaných účtenek.

V horním menu můžeme vidět tři nápisy. Zatímco první je jen název/logo aplikace, tak *Přehled* a *Účtenky* budou mít své stránky. Na *Přehled* se díváme, proto je tučný.

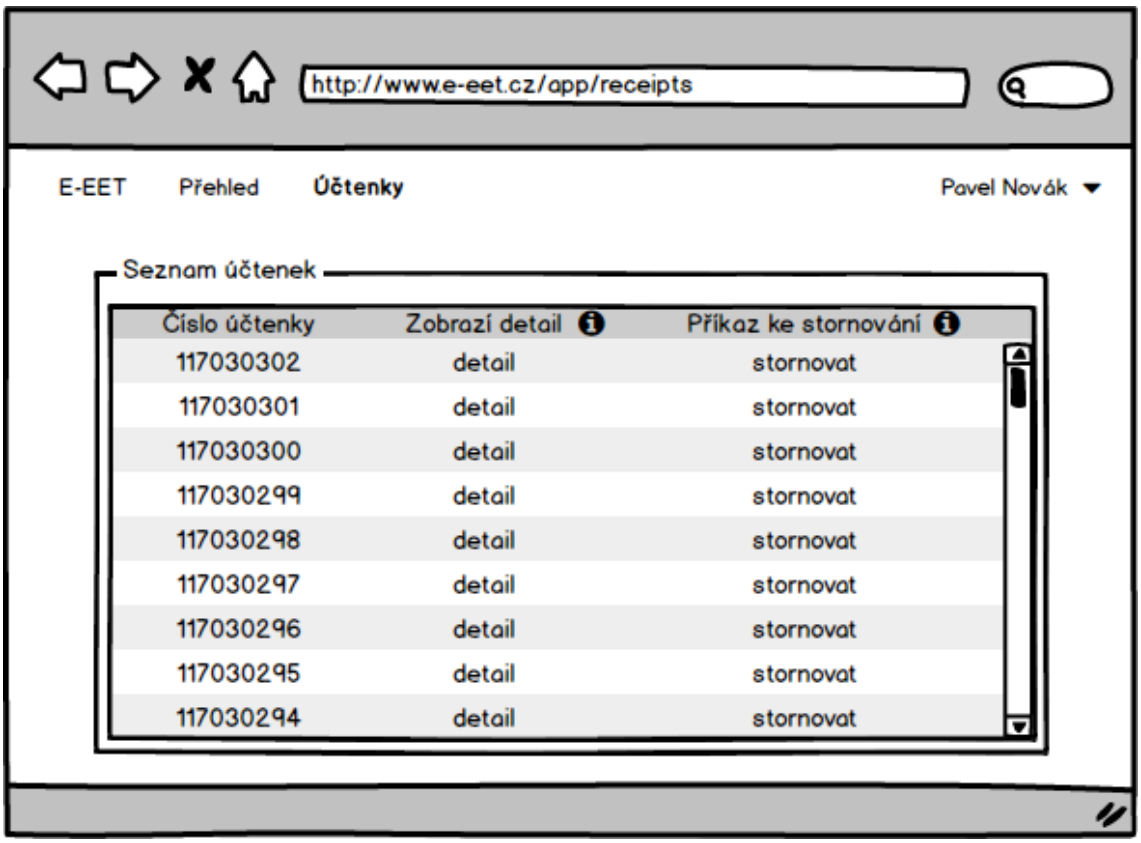
Za zmínku stojí i pravý horní roh, kde jsem naznačil podmenu, které se objeví po kliknutí na přihlášeného uživatele.

5.2.2.4 Účtenky

Kromě *Přehledů*, kde lze vidět graf spíše pro zajímavost, existuje sekce *Účtenky*, kde, jak je vidět na Obrázek 18, vidíme seznam všech evidovaných tržeb. V prvním sloupci se zobrazuje *číslo účtenky* ve formátu, v jakém je posíláno do Finanční správy. V našem případě pro jednodušší identifikaci jsme zvolili číslo objednávky z e-shopu. Další sloupec je tlačítko *detail*, po kliknutí uvidíme výpis všech položek posílaných do evidence. Tlačítko *stornovat* v posledním sloupci slouží pro jednoduché stornování účtenky.

Stornování se totiž provádí tak, že pošleme úplně stejné atributy jako cestou do evidence, jenom částku dáme v mínusové hodnotě.

Obrázek 18 - Seznam účtenek

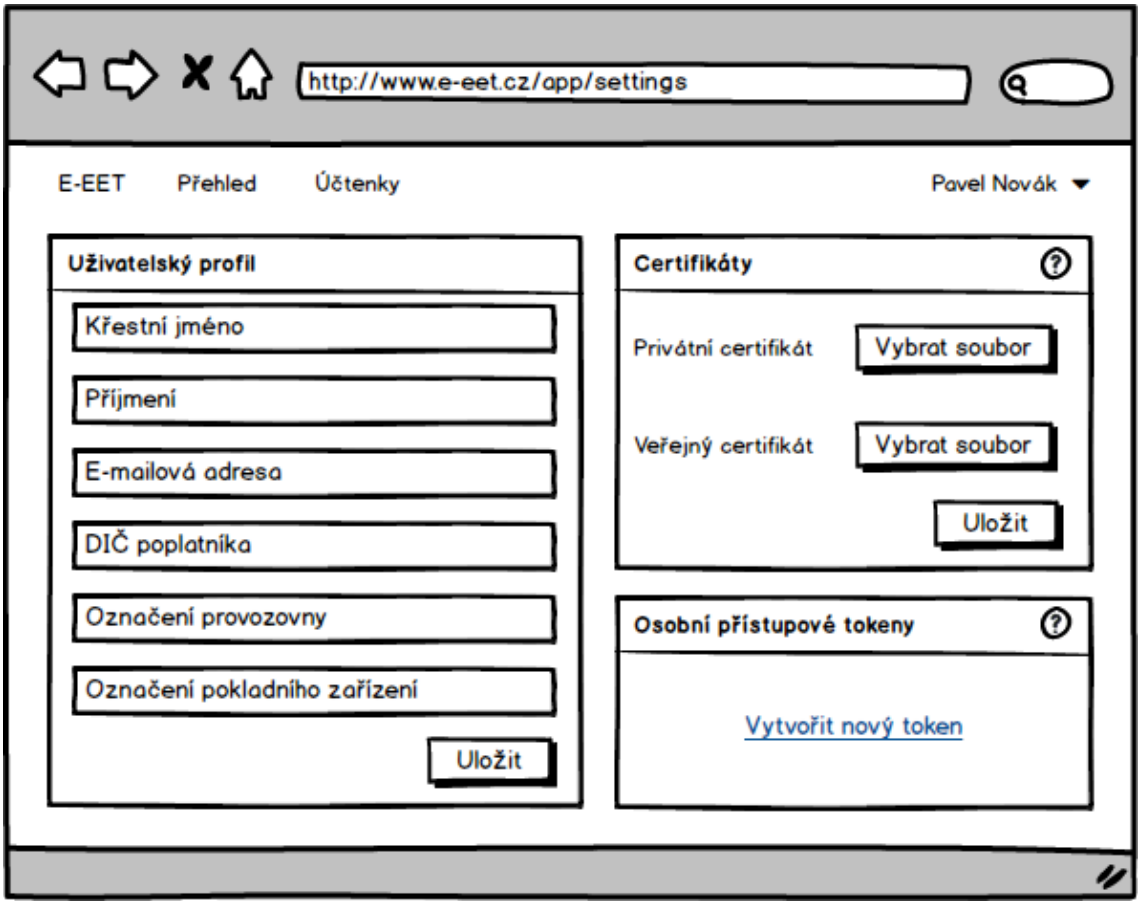


Zdroj: vlastní zpracování

5.2.2.5 Nastavení

Ve wireframu *Přehled* jsem ukazoval podmenu v pravém horním rohu. Po kliknutí na *Nastavení* se dostaneme do Obrázek 19.

Obrázek 19 - Nastavení



Zdroj: vlastní zpracování

Jedná se o nejdůležitější část celé aplikace. V sekci *Uživatelský profil* vlevo je třeba doplnit tři poslední pole, zbytek již známe, tedy *DIČ poplatníka, Označení provozovny a Označení pokladního zařízení*. Poslední dvě zmiňované se vyplňují i na portálu pro evidenci tržeb a je nutné zvolit stejné údaje! V pravé části nahoře se nahrává *Privátní certifikát*, který si každý musí stáhnout z portálu pro evidenci tržeb. A pravá část dole zobrazuje místo pro tokeny, ty slouží pro komunikaci. Nejčastěji tedy majitel musí token vytvořit a předat ho programátorovi, ten s ním poté komunikuje s naší aplikací a je jím se svým profilem na E-EET jedinečně spjat.

5.3 Technické řešení

Provoz produkční verze aplikace je od počátku zajištěn pomocí portálu Heroku, který funguje jako PaaS, jeho hlavní výhodnou je integrovaný Git. Provozovatelé o sobě píší, že vývojář se díky nim může soustředit přímo na vývoj a nemusí se starat o servery, infrastrukturu ani hardware. Vývoj aplikace by tak měl být jednoduší, rychlejší a měl by pomoci dostat aplikaci co nejrychleji na trh. (Heroku, 2017)

Aplikace je vyvíjena pomocí modelu MVC, kde modely zajišťují komunikaci s databází a ovladače hlavní aplikační logiku. Nejzajímavější částí aplikace jsou `controllery`, nejlepší český překlad by byl zřejmě „ovladače“. Konkrétně používá dva a to, jak už bylo ukázáno v diagramu tříd, `UserController` a `ReceiptsController`, postupně nyní vysvětlím jejich nejdůležitější části.

5.3.1 UserController

Tento ovladač obsahuje základní funkce pro práci s uživateli.

Zdrojový kód 1 - Funkce obsažené v UserController

```
public function show(Request $request)
{
      return $request->user();
}
public function update(Request $request)
{
      $user = $request->user();
      $user->first_name = $request->get('firstName');
      $user->last_name = $request->get('lastName');
      $user->email = $request->get('email');
      $user->vat_id = $request->get('vatId');
      $user->premise_id = $request->get('premiseId');
      $user->cash_register_id = $request->get('cashRegisterId');
      $user->save();

      return $user;
}
```

Zdroj: vlastní zpracování

Na řádku číslo jedna vidíme funkci `show`, která složí pro zobrazení daného uživatele. Další funkcí je update, který aktualizuje informace o uživateli. Řádky ve funkci `update` odpovídají polím, která je nutné vyplnit v nastavení profilu uživatele E-EET.

5.3.2 ReceiptsController

Tento již složitější ovladač se stará o logiku spojenou s účtenkami. Je jednou z nejdůležitějších částí systému a stará se o hlavní logiku, nebo funkcionalitu systému.

Zdrojový kód 2 - Funkce zobrazující účtenky

```
public function show(Request $request, string $uuid)
{
      return $request->user()->receipts()->where('id', $id)->get();
}
public function list(Request $request)
{
      return $request->user()->receipts()->get();
}
```

Zdroj: vlastní zpracování

V první funkci, kterou vidíme, používáme pouze jednu konkrétní účtenku. Zatímco funkce následující vypíše list všech účtenek od daného uživatele.

Zdrojový kód 3 - Vytvoření účtenky – validace

```
public function create(Request $request)
{
      // validate the request
      $createReceiptRequest = new CreateReceiptRequest($request-
>all(), $request->user());

      if ($createReceiptRequest->isValid() === false) {
            return $this->response->array([
                  'message' => 'Validation error',
                  'errors' => $createReceiptRequest->getErrors(),
                  'data' => [],
                  'status_code' => 400,
            ])->setStatusCode(400);
      }
}
```

Zdroj: vlastní zpracování

Nejdůležitější funkce vytvoření účtenky `create`. Pro její přehlednost jí rozdělím do několika bloků. Zde v úvodu se provádí validace požadavku. Kdyby byl chybný, vrátí se chyba *„Validation error“* a HTML status kód 400.

Zdrojový kód 4 - Vytvoření účtenky – dispečer

```
// create dispatcher
$certificate = new Certificate(storage_path('eet/certificates/378477
8138.p12'), 'SUPERTAJNEHESLO');
$dispatcher = new Dispatcher($certificate);
$dispatcher->setProductionService();
```

Zdroj: vlastní zpracování

V této fázi se vytvoří dispečer, který obstarává komunikaci s účastněnými stranami. Je vidět, že pro komunikaci používá certifikát, který uživatel nahrál ve svém profilu v sekci nastavení. Také je nutné vyplnit zde heslo, které uživatel spojil s certifikátem. Tak je možné s certifikátem dále pracovat a dostat například kódy, které se vrací při neúspěšné evidenci.

Zdrojový kód 5 - Vytvoření účtenky – účtenka

```
// create EET receipt and data object
$receipt = new EetReceipt;
$data = [];

$receipt->uuid_zpravy = $data['uuid_zpravy'] = UUID::v4();
$receipt->id_provoz = $data['id_provoz'] = $request->user()-
>id_provoz;
$receipt->id_pokl = $data['id_pokl'] = $request->user()->id_pokl;
$receipt->dic_popl = $data['dic_popl'] = $request->user()->dic_popl;
$receipt->porad_cis = $data['porad_cis'] = $request->porad_cis;
$receipt->dat_trzby = $data['dat_trzby'] = new \DateTime($request-
>dat_trzby);
$receipt->celk_trzba = $data['celk_trzba'] = $request->celk_trzba;
$receipt->zakl_nepodl_dph = $data['zakl_nepodl_dph'] = $request-
>zakl_nepodl_dph;
$receipt->zakl_dan1 = $data['zakl_dan1'] = $request->zakl_dan1;
$receipt->dan1 = $data['dan1'] = $request->dan1;
$data['rezim'] = $receipt->rezim;
```

Zdroj: vlastní zpracování

V této fázi se nejdříve vytvoří objekt účtenka a následně se do jeho atributů doplní údaje posílané z e-shopu. Tak vlastně vzniká kompletní účtenka.

Zdrojový kód 6 - Vytvoření účtenky – evidence

```
// send the receipt and get codes
$response = [];

try {
	$data['fik'] = $dispatcher->send($receipt);
	$data['bkp'] = $dispatcher->getBkp();
	$data['pkp'] = $dispatcher->getPkp();

	$response = [
		'message' => 'Success',
		'data' => $data,
		'errors' => [],
		'status_code' => 200,
	];

} catch (\Exception $exception) {
	$data['fik'] = null;
	$data['bkp'] = $dispatcher->getBkp();
	$data['pkp'] = $dispatcher->getPkp();

	$response = [
		'message' => 'Server error',
		'data' => $data,
		'errors' => [
			'message' => $exception->getMessage(),
			'code' => $exception->getCode(),
			'file' => $exception->getFile(),
			'line' => $exception->getLine(),
		],
	'status_code' => 500,
	];
	}
```

Zdroj: vlastní zpracování

Následně se tedy zkompletovaná účtenka posílá na Finanční správu, kde se, jestliže proběhne všechno v pořádku, vrátí kódy FIK, BKP a PKP, i se správou „*Success*“ a kódem 200. V případě, že nastala nějaká chyba, kód vrací pouze BKP a PKP, které zná a oznamuje chybu s kódem 500.

Zdrojový kód 7 - Vytvoření účtenky – uložení

```
 finally {
// store receipt in database
	$request->user()->receipts()->create($data);

// return result
	return $this->response->array($response)-
>setStatusCode($response['status_code']);
}
```

Zdroj: vlastní zpracování

V poslední části se vytvořená účtenka uloží, aby jí měl uživatel k dispozici na svém profilu, a hlavně aby ji systém mohl v případě potřeby uživatele stornovat. Na úplný závěr se nastaví výsledek operace v podobě status kódu.

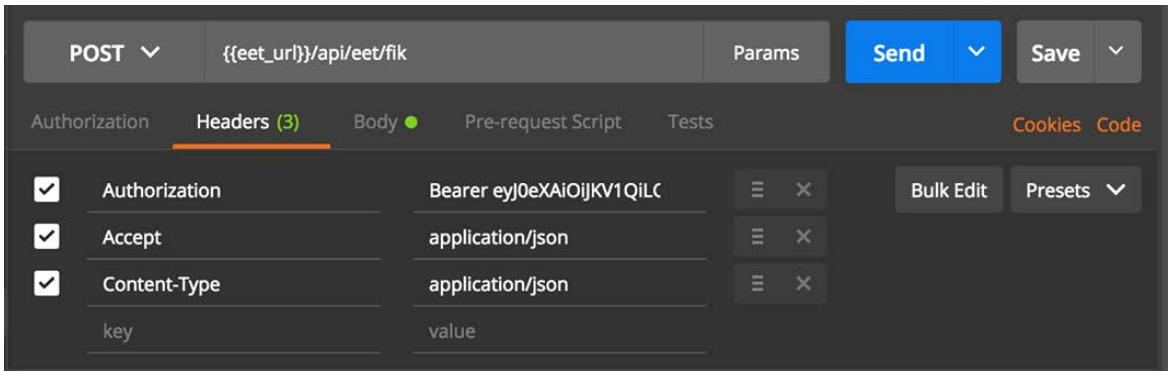
5.3.3 Nasazení na server

Nasazení na server probíhá jednoduchým způsobem pomocí verzovacího systému Git. Odesláním změn do větve master dojde k automatickému sestavení aplikace na portálu Heroku. Po ukončení sestavení je aplikace dostupná online. Jako uživatel si samozřejmě můžete koupit svou doménu, pod kterou svou aplikaci najdete. Aplikace běží na adrese www.e-eet.cz.

5.4 Implementace do e-shopu

Nyní se dostáváme ke zmiňované jednoduché fázi implementace. Jediné, co je třeba do e-shopu vložit, je POST request. V něm jsou důležité dvě části – hlavičky (headers) a tělo (body). Adresou, na kterou se dotazujeme, je *www.e-eet.cz/api/receipt*, to je vidět v prvním řádku obou obrázků níže.

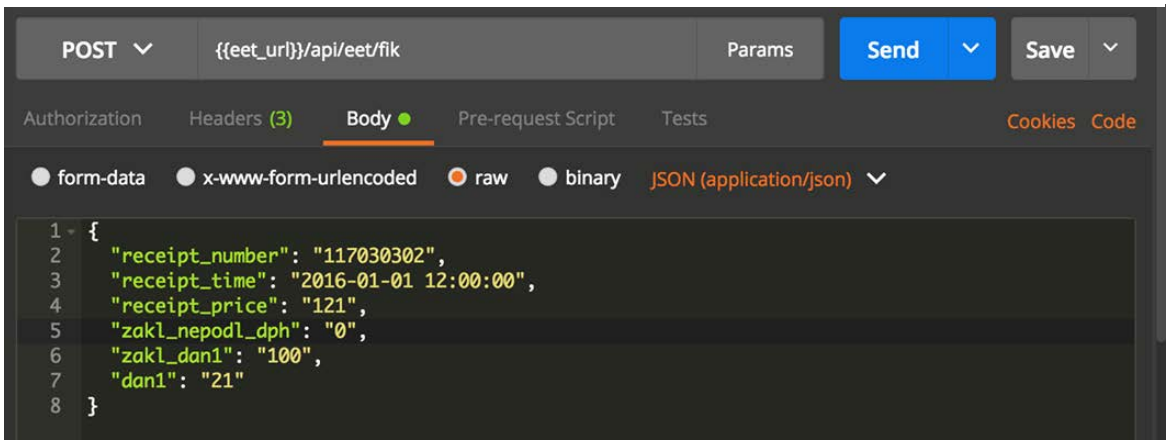
Obrázek 20 - Hlavičky metody POST



Zdroj: vlastní zpracování

V hlavičce je důležitý údaj *Authorization*, kam se vkládá vygenerovaný token, z administrace majitele e-shopů se slovem *Bearer* na začátku, to značí autorizační token a je součástí metody OAuth, o které jsem již psal. Další dva údaje nám stručně říkají, s jakým typem internetového média pracujeme, stačí vyplnit `application/json`.

Obrázek 21 - Tělo metody POST



Zdroj: vlastní zpracování

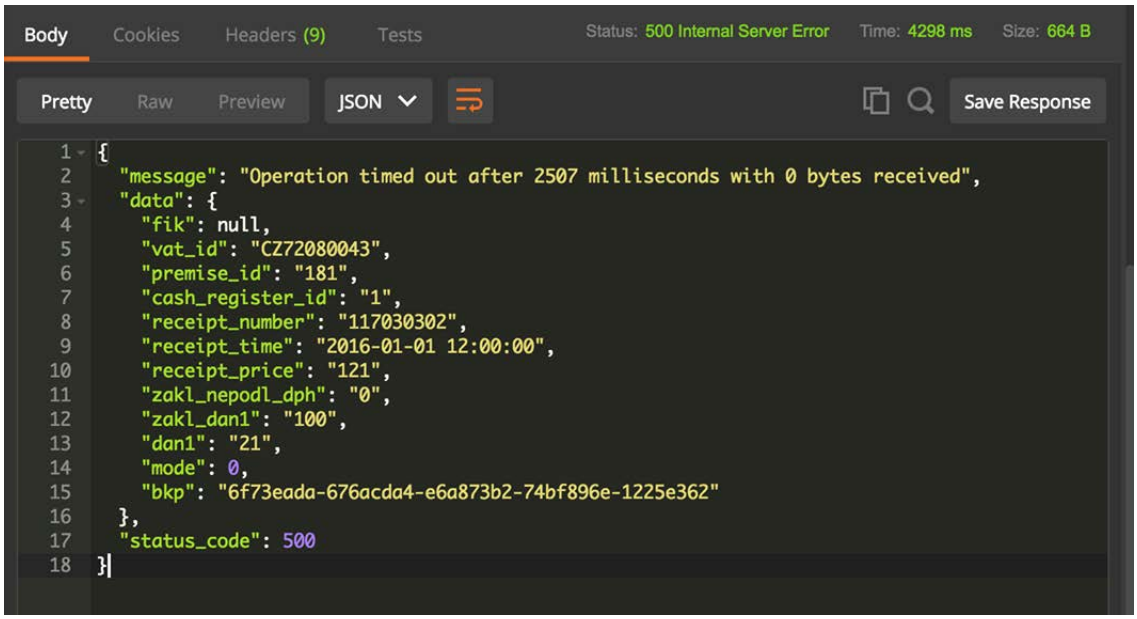
Z obrázku výše můžeme vidět, jaké hodnoty e-shop posílá. V našem případě se tedy jedná o:

- receipt_number = pořadové číslo účtenky
- receipt_time = datum a čas přijetí tržby
- receipt_price = celková částky tržby
- zakl_nepodl_dph = celková částka plnění osvobozených od DPH, ostatní plnění
- zakl_dan1 = celkový základ daně se základní sazbou DPH
- dan1 = celkové DPH se základní sazbou

Vše již bylo v práci popisováno. Za připomenutí stojí jen hodnota nula v sekci *zakl_nepodl_dph*, která se vyplňuje a posílá pouze v případě fakturace na Slovensko pro plátce DPH. Její hodnotou by byla částka bez DPH. Zbylé dvě hodnoty by poté naopak byly nulové.

Zde je důležité zmínit, že kdyby e-shop chtěl použít naši doplňkovou službu pro posílání účtenek z naší strany, je třeba z e-shopu předávat ještě e-email zákazníka.

Obrázek 22 - Odpověď dotazu POST



Zdroj: vlastní zpracování

Pro ukázku odpovědi jsem zvolil chybovou zprávu, která nám v sekci *message* oznámí chybu. Rozdíl, oproti zprávě potvrzovací, je ten, že v atributu *fik*, by byl vyplněn fiskální identifikační kód, s kterým poté může programátor dále pracovat. Nejčastěji ho tedy použije do účtenky, kterou musí vystavit.

Kromě toho se nám vrací všechny posílané údaje ze strany e-shopu, rozšířené o údaje, které jsme přidali z profilu uživatele (majitele e-shopu). Všechny údaje byly již opět v práci popsány.

6 Ekonomická hlediska

Pod tímto názvem bych rád v této kapitole ukázal výhodnost pro majitele e-shopu a zároveň udržitelnost aplikace E-EET.

6.1 Výhodnost

Aby bylo možné docílit ekonomické výhodnosti, musí náklady na implementaci a paušál za používání E-EET dlouhodobě nepřevyšovat cenu naprogramování a implementace řešení vlastního.

Pro relevantnost jsem požádal dva kamarády na volné noze[3], kteří se EET zabývali. Ti bez předchozí dohody odhadli, pro kompletní naprogramování do e-shopu, stejně 7 MD[4]. Ještě jsem poprosil projektového managera z jedné IT firmy, která EET také svým klientům implementovala. Obecně platí, že firmy jsou více opatrné, dávají si rezervy a snaží se do problematiky opravdu proniknout, nepodceňují zabezpečení a spuštění až opravdu ostré verze, což se příliš nedodržuje. I z toho důvodu byl jeho odhad dokonce 20 MD! Když vytvořím z jejich sazeb průměr, dostávám se poté na 5000 Kč za MD. Tuto sazbu použiji v Tabulka 5.

Programátor e-shopu tisknisi.cz implementoval řešení E-EET za osm hodin! Na rychlosti mu ubíralo to, že byl první, a tak jsme spolu ladili vše osobně a bez finální specifikace a dokumentace. Troufám si říci, že po těchto zkušenostech a sepsání návodů, které umístím na webové stránky, se dokáže čas stáhnout i na polovinu! Nicméně pro reálnost počítám v tabulce s oněmi osmi hodinami s jeho hodinovou sazbou 650 Kč.

Tabulka 5 – Cenové náročnosti implementace

Freelancer 1	Freelancer 2	IT firma	E-EET
35 000 Kč	35 000 Kč	100 000 Kč	5 200 Kč

Zdroj: vlastní zpracování

Z tabulky je patrné, že jsme dosáhli jednoho z hlavních cílů, tedy toho, že cenová zátěž pro majitele e-shopů je s řešením E-EET nejnižší.

6.2 Udržitelnost

Zde samozřejmě platí také pravidlo, že náklady nesmí převyšovat výnosy.

[3] Člověk na volné noze anglicky freelancer.
[4] MD = ManDay = „člověko den“, tedy osm hodin práce jednoho programátora.

Náklady

Náklady jsou zde hlavně poplatky za umístění na serveru neboli hosting a cena domény. V současné době běží aplikace na portálu Heroku, který umožňuje účtování pouze za dobu aktivního používání, tzn. za dobu, kdy přišli zákazníci na váš web/aplikaci. Velikou výhodou je účtování po vteřině! Jelikož je internet plný robotů, i při neaktivním používání přibývají minuty celkem rychle. Ne všichni totiž zachytí filtry, které má Heroku nastavené. A tak i při téměř žádné zátěži, přišly ze začátku nájmu ceny na zhruba 170 Kč s DPH. V nejbližší době tedy zvolím přesun hostingu na českého poskytovatele Wedos, který pronajímá virtuální servery za fixních 121 Kč s DPH na měsíc.

Dále je třeba počítat se zmíněnou doménou. Ty jsou v současné době opravdu levné. Také s případným programátorem pro zprávu E-EET. Budeme-li předpokládat, že vše funguje a v systému evidence nebudou změny, půjde jenom o kosmetické úpravy v řádu hodin, pro náš příklad uvažujme celý den práce tedy jeden MD. Předpokládejme hodinovou sazbu programátora 650 Kč.

Aby se o aplikaci dozvěděla široká veřejnost, je třeba počítat s náklady na marketing. V současné době pracuje spousta specialistů na principu optimalizací, tedy investují jenom tolik, aby vždy byli v zisku. Nenutí vás například investovat do PPC reklam nebo do sponzorovaných příspěvků na sociálních sítích deset tisíc měsíčně, když vyděláte jen dva. I tak budu v nákladech počítat s určitou částkou, kterou budu brát jako investici, i kdyby měla být v počátcích ztrátová.

Výnosy

Po analýze konkurence, jejich ceníků, cenových politik a s přihlédnutím k tomu, aby E-EET bylo dlouhodobě výhodnější, navrhuji poplatek za transakci ve výši **0,5 %** z objednávky, a při vyšším tarifu, který zprostředkovává i poslání elektronické účtenky e-mailem, **1 %**.

Co se týká výnosů, e-shop tisknisi.cz má v současné době průměrný počet objednávek měsíčně 368, při průměrné hodnotě 1 053 Kč s DPH. Když budeme pro simulaci předpokládat, že bychom tisknisi.cz rozdělili na dva e-shopy, kde jeden by chtěl zajistit posílání účtenek z naší strany, tak dostaneme 184*1 053 = 193 752*0,01 = **1 937,52 Kč** při procentním poplatku a **968,76 Kč** při půl procentním.

Výsledky a součet je vidět v Tabulce 6.

Tabulka 6 - Náklady a výnosy pro udržitelnost

Název	**Náklady/měsíc**	**Výnosy/měsíc**
Doména	13 Kč	-
Virtuální server	121 Kč	-
Programátor	5 200 Kč	-
Marketing	2 000 Kč	-
Zprostředkovaná evidence	-	2 906,28 Kč
Součet	**7 334 Kč**	**2 906,28 Kč**

Zdroj: vlastní zpracování

Z tabulky je vidět, že pouze e-shop tisknisi.cz aplikaci E-EET neuživí. Pozitivní ale je, že stačí pouze další dva e-shopy o stejné velikosti, aby aplikace začala být zisková! Navíc reálnou potřebu programátora počítám daleko nižší, pravdu ale ukáže až čas.

Všiml jsem si, že se hodně služeb v čase přizpůsobuje trhu a zpětné vazbě, jelikož by tento systém nebyl pro velké e-shopy, které jsou často právě na zelené louce vytvořené. Tak bych navrhoval transformaci ceníku tak, že maximální paušál za využívání E-EET bude na částce 1 999 Kč s DPH. Nechal bych to obchodně otevřené a přizpůsobil jednáním s velkými klienty. V tomto případě by na udržitelnost systému stačily 4 velké e-shopy.

7 Závěr

Důležitým faktorem pro pochopení dané problematiky bylo důkladné nastudování rozsáhlé dokumentace, týkající se elektronické evidence tržeb. Proto značnou část tvoří její základní popis ihned v úvodu práce. Následuje popis technologií a nástrojů, které jsem se rozhodl použít pro vývoj aplikace, všechny patří k nejmodernějším a nejpoužívanějším v současné době. V závěru teoretické části jsem čtenáře seznámil s e-shopem Tisknisi.cz.

V praktické části práce jsem nejprve popsal myšlenku ohledně finální aplikace a její účel, který pomáhá podnikatelům snižovat náklady implementace elektronické evidence tržeb do jejich e-shopů. Když byl definován cíl, mohl jsem se pustit do architektury aplikace. Pomocí nástrojů UML, konkrétně dvou zřejmě nepoužívanějších diagramů současnosti, jsem aplikaci namodeloval a definoval zapojené subjekty a funkce, které aplikace potřebovala. Použil jsem digram případů užití a digram tříd. Po definování pozadí aplikace bylo načase navrhnout, jak by aplikace měla vypadat z pohledu uživatele. Proto jsem pomocí drátěných modelů navrhl všechny části aplikace.

V nejtechničtější části práce jsem popsal ovladače, které řeší hlavní aplikační logiku. Prvním ovladačem je `UserController`, který obsahuje základní funkce pro práci s uživateli. Druhým je `ReceiptsContoller`, který se stará o logiku spojenou s účtenkami. Ukázky kódů doplňují popisy jejich řádků, funkcí a účelů. Následovala část, ve které jsem vysvětlil, jak jednoduchá je implementace řešení E-EET. Ta probíhá tím způsobem, že do e-shopu přidáme tzv. „POST request“ se specifickou hlavičkou a tělem zprávy.

V samotném závěru práce, konkrétně z kapitol Výhodnost a Udržitelnost, vyplývá, že náklad na implementaci do e-shopu se bude pohybovat kolem 5 200 Kč. Počítáme-li k tomu zmíněný maximální paušál pro větší e-shopy 1 999 Kč měsíčně, tak ceny pro čistou implementaci, kterou navrhovali lidé na volné noze, dosáhneme za 15 měsíců. V případě firmy dokonce za 47 měsíců! Toto jsou dvě čísla, která jednoznačně ukazují, že použití řešení E-EET je dlouhodobě výhodnější. Tomu přispívá i fakt, že z hlediska cash-flow nemá firma často takové jednorázové prostředky, které odhadovali mnou dotazovaní programátoři a projekt manager. Z těchto faktů a čísel si troufám konstatovat, že cíle práce byly dosaženy a řešení je efektivní.

Výslednou aplikaci si lze prohlédnout na stránce www.e-eet.cz.

I. Summary and keywords

My diploma thesis is focused on creating an application for Electronic Register of Sales. The main sense of the application is to facilitate an implementation to e-shops, which were created on so-called “Greenfield”. This application is supposed to help programmers in their work, because the most complex logic will be done by the app. Another big advantage of it is saving money to the e-shop owners. The implementation of my solution is more time saving than the implementation which is provided by Financial administration of the Czech Republic.

The first part of my thesis is concentrated on Electronic Register of Sales issue. It’s following by a description of technology and tools which I used for the app development. The selected technology and tools are the most modern and widely used nowadays. The end of the theoretical part is about introducing e-shop Tisknisi.cz

In the practical part of the thesis I described the “E-EET” application and compiled its structure in terms of frontend and backend. Then I explained its technical solution. Code samples are accompanied by explanations. A phase “implementation solution” is described by POST request method. I’ve designed a price policy to ensure the profitability of “E-EET” application by myself.

Keywords:

Electronic Register of Sales, e-shop, implementation, UML, Wireframe, Laravel, PostgreSQL, PHP, HTML, CSS, E-EET

II. Seznam použitých zdrojů

Adaptic. (2016). *Backend*. Načteno z Adaptic: http://www.adaptic.cz/znalosti/slovnicek/backend/

Adaptic. (2016). *Frontend*. Načteno z Adaptic: http://www.adaptic.cz/znalosti/slovnicek/frontend/

Asociace malých a středních podniků a živnostníků ČR (AMSP ČR). (2016). *O co jde?* Načteno z Elektronická evidence tržeb (EET): http://www.eltrzby.cz/cz/o-co-jde

Bean, M. (2015). *Laravel 5 Essentials.* Birmingham: Packt Publishing.

Bell, D. (15. Červen 2003). *An introduction to the Unified Modeling Language*. Načteno z IBM: https://www.ibm.com/developerworks/rational/library/769.html

Coggeshall, J. (3. Květen 2001). *Embedding PHP in HTML*. Načteno z OnLamp: http://www.onlamp.com/pub/a/php/2001/05/03/php_foundations.html

Compute Staff. (11. Únor 2011). *Man Who Invented the World Wide Web Gives it New Definition*. Načteno z Compute Magazine: http://computemagazine.com/man-who-invented-world-wide-web-gives-new-definition/

Čápka, D. (2013). *1. díl - Úvod do UML*. Načteno z ITnetwork: http://www.itnetwork.cz/navrhove-vzory/uml/uml-uvod-historie-vyznam-a-diagramy

Česko. (16. Březen 2016). *Zákon č. 112/2016 o evidenci tržeb.* Načteno z etržby: http://www.etrzby.cz/assets/cs/prilohy/sb0043-2016(12).pdf

Ferguson, N. (3. Srpen 2016). *What's The Difference Between Frontend And Backend?* Načteno z CareerFoundry: http://blog.careerfoundry.com/web-development/whats-the-difference-between-frontend-and-backend

Finanční správa ČR. (2016). *Co když něco selže?* Načteno z etržby: http://www.etrzby.cz/cs/co-kdyz-neco-selze

Finanční správa ČR. (10. Říjen 2016). *Formát a struktura údajů o evidované tržbě a popis datového rozhraní pro příjem datových zpráv evidovaných tržeb.* Načteno z etržby: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf

Finanční správa ČR. (2016). *Jak to funguje?* Načteno z etržby: http://www.etrzby.cz/cs/jak-to-funguje

Finanční správa ČR. (10. Říjen 2016). *Popis položek datové zprávy a příklady situací při evidenci tržeb.* Načteno z etržby: http://www.etrzby.cz/assets/cs/prilohy/Popis_polozek_datove_zpravy_v3.1.pdf

Heroku. (2017). *What is Heroku?* Načteno z Heroku: https://www.heroku.com/about

Jackson, J. (31. Červenec 2014). *PHP gets a formal specification, at last*. Načteno z IT WORLD: http://www.itworld.com/article/2697195/enterprise-software/php-gets-a-formal-specification--at-last.html

Laravel. (2017). *The Laravel PHP Framework.* Načteno z GitHub: https://github.com/laravel/framework

Lim, W. (18. Červen 2012). *A Beginner's Guide to Wireframing*. Načteno z envatotuts: https://webdesign.tutsplus.com/articles/a-beginners-guide-to-wireframing--webdesign-7399

Matzner, J. (29. Červen 2015). *Dali jsme zákon o #EET k nastudování právníkovi. Ten ho smetl ze stolu.* Načteno z Podnikatel.cz: http://www.podnikatel.cz/clanky/dali-jsme-zakon-o-eet-k-nastudovani-pravnikovi-ten-ho-smetl-ze-stolu/

Mills, D. C. (28. Listopad 2016). *So what is HTML, really?* Načteno z Mozzila Developer Network: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

Mozilla Developer Network. (18. Únor 2017). *CSS*. Načteno z Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/Web/CSS

OAuth. (2012). *OAuth 2.0*. Načteno z OAuth: https://oauth.net/2/

php.net. (2017). *History of PHP*. Načteno z PHP: http://php.net/manual/en/history.php.php

php.net. (2017). *What can PHP do?* Načteno z PHP: http://php.net/manual/en/intro-whatcando.php

PostgreSQL. (2017). *About*. Načteno z PostgreSQL: https://www.postgresql.org/about/

Silva, N. (2. Únor 2012). *The Complete Guide to UML Diagram Types with Examples.* Načteno z creately: http://creately.com/blog/diagrams/uml-diagram-types-examples/

Šedivý, F. (2017). *Pojmenování tříd*. Načteno z GitHub: https://github.com/filipsedivy/PHP-EET/wiki/Pojmenování-tříd#receipt

w3schools. (2015). *Styling HTML with CSS*. Načteno z w3schools: https://www.w3schools.com/html/html_css.asp

w3Techs. (10. Březen 2017). *Usage statistics and market share of PHP for websites.* Načteno z w3Techs: https://w3techs.com/technologies/details/pl-php/all/all

III. Seznam obrázků a tabulek

IV. Seznam příloh

V. Přílohy

Příloha 1 - UserController

```php
<?php
namespace App\Http\Controllers\Api\V2;
use App\Http\Controllers\Api\Controller;
use Illuminate\Http\Request;
/**
 * @Resource("User", uri="/user")
 */
class UserController extends Controller
{
    /**
     * Get user
     *
     * Get complete information about the authorized user who made
the request.
     *
     * @Get("/")
     * @Versions({"v1"})
     * @Response(200, body={"id": 10, "username": "foo"})
     *
     * @param  Request $request
     * @return mixed
     */
    public function show(Request $request)
    {
        return $request->user();
    }
    /**
     * Update information of the authorized user who made the
request.
     *
     * @Put("/")
     * @Versions({"v2"})
     * @Response(200, body={"id": 10, "username": "foo"})
     *
     * @param  Request $request
     * @return mixed
     */
    public function update(Request $request)
    {
        $user = $request->user();
        $user->first_name = $request->get('firstName');
        $user->last_name = $request->get('lastName');
        $user->email = $request->get('email');
        $user->vat_id = $request->get('vatId');
        $user->premise_id = $request->get('premiseId');
        $user->cash_register_id = $request->get('cashRegisterId');
        $user->save();

        return $user;
    }
}
```

Zdroj: vlastní zpracování

```php
<?php

namespace App\Http\Controllers\Api\V2;

use App\Http\Controllers\Api\Controller;
use App\Http\Requests\Api\CreateReceiptRequest;
use App\Models\Receipt;
use FilipSedivy\EET\Certificate;
use FilipSedivy\EET\Dispatcher;
use FilipSedivy\EET\Receipt as EetReceipt;
use FilipSedivy\EET\Utils\UUID;
use Illuminate\Http\Request;

/**
 * @Resource("Receipts", uri="/receipts")
 */
class ReceiptsController extends Controller
{
    /**
     * Get one particular receipt.
     *
     * @Get("/{uuid}")
     * @Versions({"v2"})
     * @Parameters({
     *      @Parameter("uuid", type="string", required=true,
description="UUID účtenky.")
     * })
     * @Response(200, body={"id": 10, "username": "foo"})
     *
     * @param  Request $request
     * @param  int $id
     * @return mixed
     */
    public function show(Request $request, int $id)
    {
        return $request->user()->receipts()->where('id', $id)-
>get();
    }

    /**
     * Get all receipts.
     *
     * @Get("/")
     * @Versions({"v2"})
     * @Response(200, body={"id": 10, "username": "foo"})
     *
     * @param  Request $request
     * @return mixed
     */
    public function list(Request $request)
    {
        return $request->user()->receipts()->get();
    }
```

```php
public function create(Request $request)
    {
        // validate the request
        $createReceiptRequest = new CreateReceiptRequest($request->all(), $request->user());

        if ($createReceiptRequest->isValid() === false) {
            return $this->response->array([
                'message' => 'Validation error',
                'errors' => $createReceiptRequest->getErrors(),
                'data' => [],
                'status_code' => 400,
            ])->setStatusCode(400);
        }
        // create dispatcher
        $certificate = new Certificate(storage_path('eet/certificates/3784778138.p12'), '26Tisknisi_67o');
        $dispatcher = new Dispatcher($certificate);
        $dispatcher->setProductionService();

        // create EET receipt and data object
        $receipt = new EetReceipt;
        $data = [];

        $receipt->uuid_zpravy = $data['uuid_zpravy'] = UUID::v4();
        $receipt->id_provoz = $data['id_provoz'] = $request->user()->id_provoz;
        $receipt->id_pokl = $data['id_pokl'] = $request->user()->id_pokl;
        $receipt->dic_popl = $data['dic_popl'] = $request->user()->dic_popl;
        $receipt->porad_cis = $data['porad_cis'] = $request->porad_cis;
        $receipt->dat_trzby = $data['dat_trzby'] = new \DateTime($request->dat_trzby);
        $receipt->celk_trzba = $data['celk_trzba'] = $request->celk_trzba;
        $receipt->zakl_nepodl_dph = $data['zakl_nepodl_dph'] = $request->zakl_nepodl_dph;
        $receipt->zakl_dan1 = $data['zakl_dan1'] = $request->zakl_dan1;
        $receipt->dan1 = $data['dan1'] = $request->dan1;
        $data['rezim'] = $receipt->rezim;

        // send the receipt and get codes
        $response = [];

        try {
            $data['fik'] = $dispatcher->send($receipt);
            $data['bkp'] = $dispatcher->getBkp();
            $data['pkp'] = $dispatcher->getPkp();

            $response = [
                'message' => 'Success',
                'data' => $data,
                'errors' => [],
                'status_code' => 200,
            ];
```

```
        } catch (\Exception $exception) {
            $data['fik'] = null;
            $data['bkp'] = $dispatcher->getBkp();
            $data['pkp'] = $dispatcher->getPkp();

            $response = [
                'message' => 'Server error',
                'data' => $data,
                'errors' => [
                    'message' => $exception->getMessage(),
                    'code' => $exception->getCode(),
                    'file' => $exception->getFile(),
                    'line' => $exception->getLine(),
                ],
                'status_code' => 500,
            ];

        } finally {
            // store receipt in database
            $request->user()->receipts()->create($data);

            // return result
            return $this->response->array($response)-
>setStatusCode($response['status_code']);
        }
    }
}
```

Zdroj: vlastní zpracování

```php
<?php

namespace App\Models;

use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Laravel\Passport\HasApiTokens;
use Spatie\MediaLibrary\HasMedia\HasMediaTrait;
use Spatie\MediaLibrary\HasMedia\Interfaces\HasMedia;

class User extends Authenticatable implements HasMedia
{
    use HasApiTokens, Notifiable, HasMediaTrait;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'first_name', 'last_name', 'email', 'password', 'dic_popl',
 'id_provoz', 'id_pokl'
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    /**
     * The accessors to append to the model's array form.
     *
     * @var array
     */
    protected $appends = ['full_name', 'is_eet_ready'];

    /**
     * Always capitalize the first name when it's saved to the
database.
     *
     * @param string $value
     */
    public function setFirstNameAttribute(string $value) : void
    {
        $this->attributes['first_name'] = ucfirst($value);
    }
```

```
    /**
     * Always capitalize the last name when it's saved to the
database.
     *
     * @param string $value
     */
    public function setLastNameAttribute(string $value) : void
    {
        $this->attributes['last_name'] = ucfirst($value);
    }

    /**
     * Get the EET ready flag for the user.
     *
     * @return bool
     */
    public function getIsEetReadyAttribute() : bool
    {
        return $this->attributes['is_eet_ready'] =
            !empty($this->dic_popl) &&
            !empty($this->id_provoz) &&
            !empty($this->id_pokl);
    }

    /**
     * Get the user's full name.
     *
     * @return string
     */
    public function getFullNameAttribute() : string
    {
        return $this->attributes['full_name'] = "{$this-
>first_name} {$this->last_name}";
    }

    /**
     * Get the receipts for the user.
     */
    public function receipts()
    {
        return $this->hasMany('App\Models\Receipt');
    }
}
```

Zdroj: vlastní zpracování

Příloha 4 - Receipt

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Receipt extends Model
{
//    protected $primaryKey = null;

    public $incrementing = false;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'uuid_zpravy',
        'id_provoz',
        'id_pokl',
        'dic_popl',
        'porad_cis',
        'dat_trzby',
        'celk_trzba',
        'zakl_nepodl_dph',
        'zakl_dan1',
        'dan1',
        'rezim',
        'fik',
        'bkp',
        'pkp',
    ];

    /**
     * Get the user that owns the receipt.
     */
    public function user()
    {
        return $this->belongsTo('App\Models\User');
    }
}
```

Zdroj: vlastní zpracování

Příloha 5 - CD s elektronickými přílohami