Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Mobilní aplikace pro elektronickou evidenci tržeb

Tomáš Vyčítal

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš Vyčítal**
Osobní číslo:	**I14203**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Mobilní aplikace pro elektronickou evidenci tržeb**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Bude vypracován přehled předpisů a zákonů ohledně EET (elektronická evidence tržeb) se zaměřením na technické zpracování a přehled existujících řešení pokladních systémů pro mobilní platformy.

V praktické části bude navržena a implementována mobilní aplikace pro elektronickou evidenci tržeb (pro Android či iOS).

Pro umožnění jednoduchého přechodu živnostníků na elektronickou evidenci tržeb při nastavení spravedlivých podmínek pro všechny je potřeba navrhnout a implementovat aplikaci, kterou bude moci každý zdarma použít. Aplikace by měla fungovat na tabletech s Androidem (příp. iPadech) s minimálními nároky na systém a měla by umožnit tisk účtenek na tiskárně připojené přes USB nebo Bluetooth. Pro aplikaci bude stěžejní jednoduchost instalace i používaní. Aplikace bude uvolněna jako open-source.

Rozsah grafických prací:

Rozsah pracovní zprávy: **cca 30 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

Formát a struktura údajů o evidované tržbě: Popis datového rozhraní pro příjem datových zpráv evidovaných tržeb. *Elektronická evidence tržeb* **[online]. Praha: Finanční správa, 2016 [cit. 2016-10-22]. Dostupné z: http://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf**

MURPHY, Mark L.: *Android 2: Průvodce programováním mobilních aplikací.* **Brno: Computer Press, 2011. ISBN 978-80-251-3194-7**

MEDNIEKS, Zigurd R.: *Programming Android.* **2nd ed. Beijing: O'Reilly, c2012. ISBN 978-1-4493-1664-8**

Vedoucí bakalářské práce: **Mgr. Martin Večeřa**
Red Hat Czech, s.r.o.

Konzultant bakalářské práce: **Mgr. Tomáš Hudec**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Zdeněk Šilar, Ph.D.
pověřený vedením katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 11. 5. 2017

Tomáš Vyčítal

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu Martinu Večeřovi za užitečné rady a připomínky, v neposlední řadě také za zapůjčení pokladní tiskárny.

ANOTACE

Teoretická část popisuje předpisy a zákony týkající se EET zejména s ohledem na technické zpracování a požadavky na aplikaci a knihovnu. Dále pak přehled již existujících řešení a seznámení s použitými technologiemi.

V praktické části je vyvinuta knihovna pro komunikaci s finanční správou a pokladní aplikace pro OS Android s podporou EET.

KLÍČOVÁ SLOVA

EET, Android, Java

TITLE

Mobile application for EET (Electronic Registration of Sales)

ANNOTATION

The theoretical part part describes the laws regarding EET mainly focused on the technical aspect. It also describes the requirements for the library and application and lists existing solutions.

A library communicating with the Financial Administration and Android cash register application with EET support in the practical part .

KEYWORDS

EET, Android, Java

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

EET	Elektronická evidence tržeb
IoT	Internet of Things
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
OS	Operační systém
SOAP	Simple Object Access Protocol
XML	Extensible Markup Language

0 ÚVOD

Elektronická evidence tržeb se dotkne velkého množství podnikatelů a pro mnoho je již povinná. Zejména těm menším může způsobit značné problémy, jelikož pokladní systémy mohou být drahé či vyžadovat periodické platby. Řešení, která jsou poskytnuta zdarma, ale nejedná se o otevřený software, mohou zastarat a nemusí být aktualizována, což je pro prodejce velká komplikace. Z těchto důvodů je velmi důležité vytvořit bezplatné a otevřené řešení, které umožní zejména malým prodejcům pohodlně a bez závislosti na jediném vývojáři či firmě evidovat tržby a tisknout účtenky splňující všechny zákonné náležitosti.

Cílem teoretické části práce je seznámit čtenáře se všemi důležitými informacemi, které se týkají daného tématu. Obsahuje tedy shrnutí zákonů upravujících elektronickou evidenci tržeb a podobu účtenek, včetně způsobu jejich vydávání zákazníkům. Dále také čtenáři přiblíží technologie použité pro komunikaci s finanční správou a pro vývoj knihovny a aplikace. Součástí této části je také analýza konkurenčních řešení a formulace požadavků na výslednou knihovnu a aplikaci.

Praktická část je rozdělena na dvě části, v té první je popisována knihovna a v té druhé aplikace. Knihovna se zaměřuje jen a pouze na samotnou komunikaci s finanční správou a konverzi údajů mezi nativními datovými typy Javy a textovým formátem vyžadovaným finanční správou v XML zprávách. Aplikace tuto knihovnu využívá a přidává grafické uživatelské rozhraní, které umožňuje hlášení tržeb a tisk EET účtenek, na které jsou automaticky umisťovány i všechny další údaje vyžadované zákonem. Aplikace i knihovna jsou probírány jak z hlediska jejich rozhraní, API u knihovny a GUI u aplikace, tak i z hlediska zdrojových kódů.

1 PŘEDPISY A ZÁKONY

Elektronická evidence tržeb stanovuje povinnost neprodleně hlásit tržby centrálnímu serveru finanční správy v elektronické podobě přes Internet. Právní záležitosti, týkající se EET, upravuje zákon č. 112/2016 o evidenci tržeb[1]. Technická dokumentace je pak k dispozici na webu etrzby.cz[2], tentýž web obsahuje také spoustu dalších informací o projektu, pro firmy, podnikatele i zákazníky, včetně odpovědí na často kladené otázky a možnosti otázky pokládat.

1.1 Údaje zasílané státní správě

V rámci elektronické evidence tržeb má obchodník povinnost zasílat státní správě[2]:

- daňové identifikační číslo poplatníka,
- označení provozovny, ve které je tržba uskutečněna, Částka 43 Sbírka zákonů č. 112/2016 Strana 1981
- označení pokladního zařízení, na kterém je tržba evidována,
- pořadové číslo účtenky,
- datum a čas přijetí tržby nebo vystavení účtenky, pokud je vystavena dříve,
- celková částka tržby,
- bezpečnostní kód poplatníka,
- podpisový kód poplatníka,
- údaj, zda je tržba evidována v běžném nebo zjednodušeném režimu
- celková částka plateb určených k následnému čerpání nebo zúčtování,
- celková částka plateb, které jsou následným čerpáním nebo zúčtováním platby,
- daňové identifikační číslo poplatníka, který pověřil evidováním této tržby poplatníka, který tržbu eviduje,
- základ daně z přidané hodnoty a daň podle sazeb daně z přidané hodnoty,

- celková částka v režimu daně z přidané hodnoty pro cestovní službu,
- celková částka v režimu daně z přidané hodnoty pro prodej použitého zboží.

Tyto údaje je nutné zasílat v reálném čase. V případě technického výpadku je možné údaje zaslat později, v běžném režimu nejpozději do 48 hodin od přijetí tržby. Ve zjednodušeném režimu, který může být obchodníkovi schválen úřadem, se údaje zasílají později. V tomto případě je časové omezení 5 dní. V každém případě je nutné vydat účtenku se všemi náležitostmi, ty budou probrány podrobněji v jedné z následujících kapitol.

1.2 Údaje uváděné na EET účtence

EET účtenka musí obsahovat následující údaje[1]:

- fiskální identifikační kód (FIK) nebo podpisový kód poplatníka (PKP),
- své daňové identifikační číslo (DIČ),
- označení provozovny, ve které je tržba uskutečněna,
- označení pokladního zařízení, na kterém je tržba evidována,
- pořadové číslo účtenky,
- datum a čas přijetí tržby nebo vystavení účtenky, pokud je vystavena dříve,
- celkovou částku tržby,
- bezpečnostní kód poplatníka (BKP),
- údaj, zda je tržba evidována v běžném nebo zjednodušeném režimu.

Výše uvedené údaje musejí být uvedeny na EET účtence. Klasickou účtenku nelze EET účtenkou nahradit, jelikož EET účtenky obsahují pouze celkovou částku a nenaplňují tedy zákonné požadavky na ochranu spotřebitele. Stejně tak nelze klasickou účtenkou nahradit EET účtenku, protože ta neobsahuje údaje specifické pro EET. Kromě chybějících EET kódů je ještě požadováno přesnější datum a identifikace prodejny a pokladny či pořadové číslo účtenky. Nicméně nic nebrání umístění všech zákonem vyžadovaných údajů na jednu dlouhou účtenku, která pak slouží jak jako EET účtenka tak i jako klasická účtenka.

FIK je v určitých případech na účtence možné nahradit PKP, který je ale podstatně delší. Rozdíl činí 305 znaků, FIK má 39 znaků a PKP 344. Důvodem k nahrazení může být evidování ve zjednodušeném režimu, kde jsou tržby hlášeny se zpožděním, nikoli okamžitě, nebo výpadek spojení se servery státní správy, kdy tržbu není možné okamžitě nahlásit. FIK je totiž poskytován v odpovědi serveru finanční správy. V případě, že se se serverem nelze spojit, nelze ani získat FIK, a není tedy možné jej umístit na účtenku. [2]

1.3 Údaje uváděné na klasické účtence

Na účtence, která slouží jako daňový doklad nebo třeba pro případ reklamace, je nutné uvést tyto údaje[3]:

- název firmy nebo jméno a příjmení,
- adresa – sídlo (v případě právnické osoby) nebo místo podnikání (v případě fyzické osoby),
- identifikační číslo (IČO) firmy,
- pokud je účtenka s DPH, pak DIČ,
- datum prodeje,
- přesný výčet zakoupeného zboží s následujícími hodnotami:
    - jasný popis koupeného zboží,
    - jeho množství,
    - cena za jednotku,
    - sazba daně za jednotku,
    - celková cena za položku,
    - cena celkem,
    - základní nebo snížená sazba daně;
- podpis, popř. razítko prodávajícího.

1.4 Způsob vydání účtenky

Zákon nijak detailně neupravuje vydávání účtenek, požadavky jsou následující:

- čitelnost (povinnost prodejce),
- bezchybnost (povinnost prodejce),
- vydání ve fyzické podobě (právo zákazníka, nikoli povinnost).

Účtenku je také možné vydat elektronicky (běžné například u internetových obchodů), ale jen za podmínky, že s tím zákazník souhlasí, jelikož má právo dostat fyzickou účtenku.

V případě fyzického vydání neexistuje žádný zákonný požadavek na formu či podobu účtenky. Takže je legální účtenku například ručně napsat zelenou fixou na toaletní papír, bude-li zajištěna čitelnost a bezchybnost. Stejně tak není upravován ani počet účtenek, z hlediska zákona tak není problém účtenky kombinovat nebo vydávat různými způsoby. Legální možností tedy je například vydat EET účtenku elektronicky a klasickou účtenku vytištěnou na termocitlivém papíře, souhlasí-li s tím zákazník a jsou-li dodrženy všechny zákonné požadavky. Není legální účtenku nevydat, nicméně zákazník není povinen vydanou účtenku převzít, musí mu ale být její převzetí umožněno. [4]

2 EXISTUJÍCÍ ŘEŠENÍ

Na trhu existuje velké množství komerčních pokladních systémů s podporou EET. Jejich možnosti jsou však v základních (cenově dostupných) verzích velmi omezené[5–7] v zájmu nalákání zákazníků a následného prodeje drahých licencí. Některé také mají další podmínky používání, jako například účet u banky poskytující dané řešení[7].

Existují také řešení, která tisknou pouze EET účtenky[8]. Ty jsou ovšem značně nepraktické, jelikož je nezbytné mít ještě jednu pokladnu, která bude tisknout normální účtenky, případně je vydávat jiným způsobem.

Aplikací, které jsou skutečně zdarma a naplňují základní požadavky podnikatelů, je pomálu a, až na jednu, se nejedná o otevřený software. Tyto aplikace nicméně plní svůj účel a pro uživatele, které nezajímá co daná aplikace ve skutečnosti dělá, jsou plně dostačující. [9, 10]

Jedinou open source aplikací, která je k dispozici v obchodě Google Play, což je pro drtivou většinu uživatelů Androidu jediný zdroj aplikací, je OpenEET. Zde se ale nejedná o plnohodnotnou pokladnu, která by umožňovala vydání účtenky se všemi zákonnými náležitostmi. Tato aplikace umožňuje jen a pouze hlášení tržeb finanční správě, není možné účtenky ani tisknout, byť je tato možnost již minimálně od 10. srpna 2016, což je také den prvního zveřejnění zdrojových kódů, v plánu. Kvůli tomu je nutné mít ještě jiný způsob, jakým se budou vydávat účtenky, které budou splňovat náležitosti například pro případ reklamace. Prodejce pak musí údaje zadávat do dvou různých pokladen a vydávat každému zákazníkovi dvě účtenky, jednu registrační s náležitostmi EET, kterou musí ručně napsat, z této aplikace a druhou klasickou se seznamem zboží a všemi souvisejícími náležitostmi z druhé pokladny, případně ručně vytvořenou a napsanou. Ruční psaní účtenek je velmi špatný nápad, jelikož je velmi snadné udělat chybu, zejména v číselných údajích a při opisování kódů EET. Kódy EET totiž mají 83 znaků při úspěšném nahlášení nebo 383 při neúspěšném, vzhledem k tomu, že se jedná o z pohledu prodejce nesmyslnou sekvenci až 383 znaků, rozhodně nelze jejich ruční opisování považovat za schůdnou cestu. Z tohoto důvodu se jedná spíše o nouzové řešení, jelikož je to značně komplikované a nepohodlné pro prodejce a zbytečně zdržuje jak zákazníka tak i prodejce. [11]

Další open source řešení EET existují pouze ve formě knihoven[12]. Ty jsou sice připraveny k použití, ale pro vývojáře koncových aplikací, nikoli pro koncové uživatele.

3 ANALÝZA A NÁVRH

3.1 Požadavky na knihovnu

Knihovna bude vytvořena jako samostatně fungující celek a to tak, aby bylo umožněno její využití v rámci dalších aplikací, případně knihoven dále rozšiřujících její funkcionalitu, veřejností. S její dostupností pro veřejnost se velmi úzce pojí licence, musí totiž být nastaveny velmi jasné právní podmínky používání, aby se předešlo jakýmkoli problémům či nejasnostem v budoucnu. Knihovna tedy bude k dispozici jako svobodný software pod licencí Apache ve verzi 2.0, která splňuje všechny dříve zmíněné požadavky.

V rámci knihovny bude zapouzdřena komunikace s finanční správou a to takovým způsobem, aby uživatel nemusel mít detailní znalosti o technické stránce komunikace, postačí mu vědět, jaké údaje se musí v daných situacích hlásit a samozřejmě údaje o nahlašované tržbě. Knihovna samozřejmě musí splňovat všechny požadavky, vyžadované zákonem, a to tak, aby byly všechny nahlášené tržby uznány finanční správou a nehrozilo tedy koncovému uživateli stíhání kvůli špatně nahlášeným tržbám.

Z technické stránky musí knihovna poskytovat rozumné API, které bude využíváno uživateli knihovny a v aplikaci. To by mělo uživateli umožnit co nesnadnější komunikaci se serverem finanční správy, ale zároveň poskytnout dostatečně robustní řešení, jenž poskytne všechny nezbytné údaje a umožní i řešení případných problémů s komunikací. Vzhledem k tomu, že knihovnu musí být také možné využívat z aplikace na operačním systému Android, bude napsána v Javě, v níž jsou na Androidu běžně vytvářeny nativní aplikace, a bude využívat pouze knihovny dostupné pro Android.

3.2 Požadavky na aplikaci

Aplikace musí umožnit tvorbu, hlášení a tisk účtenek podle všech platných předpisů tak, aby bylo možné ji použít v praxi a neriskovat přitom postih. Samotnou komunikaci s finanční správou bude řešit knihovna, která bude v aplikaci využívána. V aplikaci bude k dispozici seznam zboží, ze kterého budou vybírány položky a tvořeny účtenky. Tento seznam bude předem vytvořený uživatelem v rámci aplikace, bude uložen v telefonu a také součástí zálohy. Všechny nahlášené účtenky budou uloženy v telefonu či tabletu a budou i součástí zálohy, pro

případ, že se uživatel bude potřebovat k některé vrátit. Jakoukoli účtenku bude možné obnovit a následně s ní pracovat např: odeslat finanční správě zrušení dané tržby, dodatečně vytisknout účtenku a nebo ji upravit a nahlásit jako novou tržbu. Dále bude umožňovat nastavení údajů daňového poplatníka a další nezbytná nastavení. Posledním požadavkem je licence, aplikace musí být k dispozici stejně jako knihovna pod licencí Apache ve verzi 2.0.

3.3 Elektronická evidence tržeb

Cílem EET je boj s daňovými úniky, tedy s krácením a neplacením daní, jinými slovy donucení všech, kdož jsou povinni, platit daně v plném rozsahu. Tento ambiciózní plán je v praxi realizován evidováním tržeb v reálném čase a vystavením kódu zákazníkovi, který lze použít k ověření nahlášení dané tržby.

EET z technického hlediska spočívá v evidování tržeb vždy ihned v okamžiku, kdy tržba probíhá, byť je umožněno určité zpoždění z technických důvodů, maximálně však dva dny, pět dní při udělení výjimky. V rámci evidence ale nejsou požadovány žádné údaje, které by státní správa neměla k dispozici již před EET, jediné co se liší je způsob a časové omezení jejich doručení. To, co může skutečně přispět k lepší kontrole výběru daní, je FIK. Tento kód je vydáván finanční správou pro každou nahlášenou tržbu a musí být přítomen na účtence vydané prodejcem, nebo, v případě výpadku spojení či výjimky z evidence, jeho o 305 znaků delší náhrada PKP, kterou si prodejce vytváří sám. Pomocí těchto kódů a dalších údajů z účtenky si může zákazník, jejich ručním opsáním do zvláštní webové stránky finanční správy, ověřit, byla-li tržba nahlášena či nikoli. Nicméně ručně opisovat přes sto znaků z každé účtenky je značně nepohodlné, jejich vyhledávání na účtenkách také není zrovna jednoduché, účtenky totiž mohou být velmi dlouhé a každý prodejce umisťuje tyto údaje jinam. Tento postup je také velmi náchylný na chyby, protože se opisují z pohledu zákazníka nesmyslné sekvence znaků, relativně velká čísla a datum ve velmi technickém a uživatelsky nepřívětivém formátu a k tomu musí ještě zvolit režim tržby a jestli obsahuje FIK nebo ne. Jakákoli chyba je pak jen velmi obtížně odhalitelná a může zákazníka vést k mylnému dojmu, že prodejce nenahlásil tržbu. Z těchto důvodu většina lidí pravděpodobně kontroly účtenek neprovádí a provádět nebude nebo jen výjimečně, případně to po určité době vzdá.

K motivaci zákazníků k ověřování účtenek má v budoucnu sloužit účtenková loterie. Byť stát se obvykle snaží hazard spíše potlačovat a zakazovat, v tomto případě se nejedná o typickou hazardní hru, asi není pravděpodobné, že by někdo zběsile nakupoval aby mohl hrát, a vzhledem k tomu, že jejím účelem je zlepšení výběru daní, je z hlediska zákonodárců zjevně obhajitelná. Bez účtenkové loterie či jiného způsobu motivace je totiž větší zapojení zákazníků velmi nepravděpodobné a bez ověřování účtenek zákazníky nemá EET sebemenší smysl. Prodejci totiž tržby nahlašovat nemusí, byť je to porušením zákonu, a pokud zákazníci nebudou účtenky ověřovat, tak je k tomu ani nic nenutí o nic víc před příchodem EET. Mohou také vydávat účtenky s vymyšlenými kódy nebo zcela bez nich, což se velmi obtížně odhaluje bez rozsáhlých a drahých kontrol, jenž by byly prováděny velkým množstvím zaměstnanců. Právě tyto kontroly mohou díky EET provádět všichni zákazníci a finanční správa tak může získat lepší přehled o tom, kdo daně platí a kdo je krátí či neplatí vůbec, aniž by musela zaměstnávat velkou a drahou armádu kontrolorů.

Bezpečnost evidence je zajištěna několika způsoby. Z hlediska odposlouchávání komunikace je použito spojení přes HTTPS, které je šifrované. To sice nezabraňuje odposlechu, ale dešifrování takové komunikace je neúnosně výpočetně náročné a tudíž extrémně nepravděpodobné, byť je technicky možné. Z hlediska autentizace a autenticity se používají certifikáty. Každé potvrzení přijetí tržby finanční správou je podepsané a prodejce si tento podpis může ověřit. Na straně prodejce se podepisuje každá tržba, v níž se podepisují všechny údaje. Tím je zajištěno, že nikdo nemůže hlásit nebo rušit tržby za někoho jiného, alespoň tedy pokud nezíská jeho podpisový certifikát a heslo k němu. [2]

3.3.1 Situace v zahraničí

EET není zcela nová myšlenka, již existuje v Chorvatsku, jehož systém byl hlavní inspirací pro českou verzi. Podobný koncept je zaveden i v dalších státech a ve formě registračních pokladen, které udržují data v pokladně určitým způsobem chráněná proti změnám, je běžný napříč Evropou. [13] Na Slovensku fungují jak registrační pokladny, které jsou používány již delší dobu, tak i on-line registrace podobná naší EET, ta má do budoucna zcela nahradit registrační pokladny. Ověřování účtenek zákazníky je podpořeno účtenkovou loterií, jenž je plánována i u nás. [14] V sousedním Rakousku jsou zavedeny registrační pokladny, nejsou ale

povinné pro malé podnikatele s ročním obratem podle oboru do 15 000 nebo 30 000 eur, na rozdíl od české EET jsou zákazníci povinni přebírat účtenky. [15]

Zkušenosti z ostatních států jsou ale velmi sporné, není totiž možné jednoduše posoudit dopady EET či registračních pokladen kvůli nespočtu ostatních vlivů, které ovlivňují ekonomiku. Například v Chorvatsku došlo po zavedení EET k ukončení činnosti asi 50 000 subjekty, bylo by ale hloupé připisovat tento jev pouze EET, nicméně její zavedení zcela určitě hrálo důležitou roli, pro mnoho podnikatelů to mohla být pomyslná poslední kapka. Na druhé straně částka vybraná na DPH se po dvou letech od zavedení EET zvýšila, což může poukazovat na zefektivnění výběru daní, zejména pokud se vezme v potaz značné snížení počtu podnikatelů. Tento údaj ale není jednoznačným důkazem o prospěšnosti EET. Zajímavým faktem je také to, že po zavedení EET se značně snížilo množství peněz vybraných na DPH a postupně se zvyšovalo. Dále je také potřeba brát v úvahu další vlivy, které mohly tento údaj ovlivnit, a těch existuje nepřeberné množství. Jednou z možností je, že tento vliv je pouze dočasný, protože neplatiči se nestihli na evidenci připravit a nebo se jí zalekli, případně je odradilo velké množství pokut udělených za nehlášení tržeb, postupem času ale opět začnou daně krátit. Ještě je potřeba zmínit, že v Chorvatsku funguje účtenková loterie, jenž motivuje lidi k ověřování účtenek, byť její počáteční pozitivní efekt časem zeslábl, stále se jedná o funkční nástroj k motivaci lidí, který v ČR zatím chybí. Není tedy možné jednoduše říci, je-li EET dobrá či nikoli, přináší jak výhody tak i nevýhody, jejichž důležitost a velikost se liší v očích různých lidí. [13, 16]

3.4 XML

Komunikace mezi podnikatelem a finanční správou probíhá pomocí výměny XML zpráv. Jedná se o velmi starý jazyk, který umožňuje ukládat a sdílet data ve znakové podobě. Výsledná data jsou čitelná jak pro stroj tak i pro člověka, je li obeznámen se syntaxí XML.

Syntaxe spočívá v tazích, které mohou obsahovat atributy a další tagy. Tagy jsou ohraničeny znaky „<“ a „>“, uvnitř těchto znaků se nachází název tagu a za mezerou jeho atributy, více atributů se odděluje mezerami. Atributy se skládají z názvu před znakem rovná se a hodnoty v programátorských uvozovkách za tímto znakem. Pomocí uvozovek se zajišťuje, že text po mezeře v hodnotě jednoho atributu nebude zaměnitelný s názvem následujícího atributu. Po

posledním atributu nebo ihned po názvu tagu, neobsahuje-li tag žádné atributy, je otevírací tag ukončen. K ukončení otevíracího tagu slouží výše zmíněný znak „>“. Po něm následuje tělo, které může obsahovat vnořené tagy, ty se nijak syntakticky neliší od ostatních tagů, nebo může být i prázdné. Po těle tagu následuje uzavírací tag, tedy název, jenž byl použit u otevíracího tagu, obalen znaky „</“ na začátku a „>“ na konci, uzavírací tagy nemají žádné atributy, slouží pouze k ohraničení těla tagu. Je také možné místo uzavíracího tagu použít sekvenci „/>“ na konci tagu otevíracího, která ten uzavírací nahrazuje, takový tag pak ale nemůže obsahovat žádné vnořené tagy. Ještě existuje speciální tag komentáře, ten se uvozuje sekvencí znaků „<!--“ a ukončuje sekvencí „-->“, který umožňuje do XML dokumentu vložit libovolný text, jenž slouží jen pro potřeby člověka a je počítači ignorován. Tagy je možné libovolně vytvářet a vnořovat, neexistují ani žádná další omezení co se týče atributů, jen je zakázáno křížení tagů. To znamená, že není možné otevřít první tag, otevřít druhý a poté uzavřít první tag. V tomto případě by se nejednalo o validní XML, v místě uzavření tagu musí vždy být uzavřeny všechny tagy do něj vnořené. Výše zmíněné je jen stručným popisem základní syntaxe, XML je podstatně komplexnější a mocnější nástroj, který obsahuje mimo jiného schémata, která umožňují definici přesnější struktury dokumentu a jeho validaci podle ní, a jmenné prostory.

```
<tag atribut="hodnota atributu">
    <vnoreny_prazdny_tag/>
    <!-- Komentář -->
</tag>
```

Hlavní výhodou jazyka XML je jeho zavedenost. Za dobu své existence nalezl velké množství různých využití v celé řadě systémů a vzniklo i velké množství jeho variant a jazyků, na měn založených, nejznámějším příkladem je asi HTML. Také existuje spousta nástrojů, které s ním umí pracovat, ať už ve formě knihoven, pro různé jazyky a systémy, tak i finálních aplikací, které usnadňují čtení a nebo vytváření dokumentů v něm. Jako další výhodu lze jmenovat platformní nezávislost. Na druhé straně má i své nevýhody. Jednou z nich je neúspornost, XML přidává k užitečným datům celkem velké množství režie, což má za následek relativně vysoké datové toky při přenosu a nebo nároky na úložiště v případě, že jsou data skladována. Za nevýhodu lze také považovat fakt, že formát dat v XML neodpovídá běžným datovým strukturám programovacích jazyků, například neobsahuje klasická pole, ta se musí vytvářet odlišným způsobem. Další nevýhodou je vysoká náročnost na výpočetní výkon a lze také ještě

zmínit špatnou podporu v Androidu, kde není nativní podpora všech běžných rozšíření XML, je tedy nutné využít knihovnu třetí strany nebo vytvořit vlastní řešení.

3.5 SOAP

SOAP propojuje technologie HTTP a XML a definuje komunikaci mezi dvěma zařízeními. Z hlediska bakalářské práce je důležitá jeho schopnost obalit XML data, podepsat je a odeslat na server, který následně může odpovědět stejným způsobem. Vzhledem k tomu, že SOAP zprávy jsou kompletně v XML je zajištěna jejich nezávislost na platformě i programovacím jazyku. Zachovává se tím také čitelnost pro člověka a zjednodušuje se implementace na platformách, které již podporují XML. [17]

3.6 Android

Android je operační systém momentálně vyvíjený americkou společností Google. Jedná se o nejrozšířenější operační systém pro mobilní telefony a tablety na světě, lze jej najít i v chytrých televizích, autech atp. Z technického hlediska je založen na linuxovém jádře, které je modifikované pro specifické potřeby mobilních ale i dalších zařízení. Pro programátora je k dispozici rozsáhlé API Javy, to umožňuje celkem jednoduše vytvářet aplikace pro tento operační systém. Přesto že na Androidu je hlavním jazykem Java, lze také využívat jiné jazyky, jako je Lua nebo JavaScript[18].

3.7 Java

Tento programovací jazyk byl původně vyvinut společností Sun Microsystems v roce 1995, do vlastnictví Oraclu přešel poté, co Oracle zakoupil Sun. Java je kompilována do tzv. bytecodu, což je mezistupeň mezi zdrojovým kódem a strojovým kódem. Ten pak vykonává JVM, který musí být nainstalován na každém zařízení s podporou Javy. Tento koncept zajišťuje přenositelnost kódu, který je i po zkompilování platformně nezávislý. Bytecode je oproti zdrojovému kódu ve formátu snadněji zpracovatelném počítačem, byť jen velmi obtížně čitelném pro člověka. [19]

Java byla zvolena s ohledem na to, že se jedná o jazyk použitý na Androidu a je tedy relativně jednoduché v něm vytvářet nativní aplikace. To samé samozřejmě platí i pro knihovnu, u té je ještě výhodné, že je možné stejný kód, byť s určitými výjimkami jako je Base64 nebo JAXB,

používat jak na Androidu tak i kdekoli jinde, kde funguje Java, zcela beze změn nebo jen s malými úpravami.

3.8 Bluetooth

Bluetooth je bezdrátová technologie přenosu dat využívající elektromagnetické vlny. Pro vznik spojení mezi zařízeními je potřeba je spárovat, přičemž se zvolí master zařízení, to bude jen jedno a bude řídit komunikaci, a slave zařízení, kterých může být více, poté mezi nimi může probíhat komunikace. Tato technologie je velmi rozšířená v mobilních zařízeních a slouží ke komunikaci mezi nimi, např: mezi telefonem a bezdrátovou tiskárnou. Z důvodu velkého rozšíření v mobilních a zejména zařízeních IoT je kladen velký důraz i na spotřebu energie, jenž je velmi nízká zejména v posledních verzích standardu. [20]

Důvodem pro použití právě Bluetoothu je zejména jeho dostupnost. Drtivá většina pokud ne všechna mobilní zařízení dnes mají Bluetooth a velmi široká je i nabídka tiskáren s podporou komunikace pomocí Bluetoothu. Dalšími pozitivy jsou nativní podpora v Androidu, což značně usnadňuje vývoj, a pohodlnost pro uživatele, jelikož nemusí mít zařízení propojená fyzickými kabely, vše funguje bezdrátově.

3.9 Licence Apache 2.0

Jedná se o licenci pro svobodný software, která umožňuje jak šíření tak i úpravu kódu komukoli, aniž by dotyčný musel žádat o svolení nebo za takové oprávnění platit, licence mu jej uděluje automaticky. Nicméně neuděluje oprávnění používat ochranné známky mimo použití při označování původního autora a díla. [21] Je kompatibilní s GNU GPL ve verzi 3[22] a schválená organizací OSI jako licence svobodného softwaru[23].

4 KNIHOVNA

Knihovna zajišťuje komunikaci s finanční správou a to takovým způsobem, aby se uživatel nemusel zabývat technickou specifikací této komunikace. Jemu postačí knihovně poskytnout všechny hlášené údaje a zbytek, jako je například převod čísel nebo dat do požadovaných formátů či podpis zprávy, již zajistí knihovna.

Uživatelem je v případě knihovny samozřejmě programátor, který vytváří aplikaci pro koncové uživatele nebo využívá tuto knihovnu ve své vlastní, v níž přidává další funkcionalitu, knihovna tedy není určena přímo pro koncového uživatele daňového poplatníka.

4.1 Závislosti

Základní závislostí knihovny je Java, v níž je knihovna napsána, a dostupnost jejího interpretru na cílové platformě. Minimální vyžadovanou verzí je Java 7. Kvůli požadavku finanční správy na použití algoritmu Base64, jehož implementace se liší mezi standardní Javou a implementací Javy na Androidu, je knihovna funkční pouze na operačním systému Android. Nicméně tato závislost je v kódu zapouzdřena do speciální třídy a tak vytvoření verze pro osobní počítače je jen otázkou přepsání jednoho řádku zdrojového kódu. Kromě výše zmíněných již knihovna nemá žádné jiné závislosti, využívá pouze standardní knihovnu Javy, která je dodávána společně s jejím interpretrem.

4.2 Rozhraní

Rozhraní se skládá ze dvou částí.

První částí rozhraní je třída účtenky pojmenovaná EETReceipt. Ta sama o sobě neřeší komunikaci s finanční správou, slouží pouze ke shromáždění údajů od uživatele a finanční správy. Údaje zadané uživatelem jsou přijímány jako standardní datové typy jazyka Java (např: boolean, int…) a následně jsou automaticky převedeny do podoby vyžadované finanční správou. Vzhledem k velkému množství odesílaných údajů, z nichž mnohé jsou nepovinné, se údaje zadávají pomocí zřetězených metod. Tento přístup umožňuje uživateli velmi snadné a přehledné zadávání velkého množství parametrů, bez nutnosti používat desítky nepřehledných pozičních parametrů nebo opakování názvu proměnné, která drží referenci na instanci účtenky. Výhodou tohoto přístupu je také možnost nezadávání

nepovinných parametrů, aniž by bylo nutné používat hodnoty typu null. Dalšími údaji ve třídě účtenky jsou kódy BKP a PKP, které jsou vytvořeny knihovnou na základě údajů poskytnutých uživatelem, a kód FIK poskytnutý finanční správou po úspěšném nahlášení tržby; tyto kódy jsou pak na vyžádání poskytnuty uživateli. Uživatel má ještě možnost si prohlédnout komunikaci v XML, ale v běžném provozu by to nemělo být potřeba.

```
( new EETReceipt() )
    .setCelkTrzba( 3411300 )
    .setCerpZuct( 67900 )
    .setCestSluz( 546000 )
;
```

Druhou částí rozhraní je třída EET, která řeší samotnou komunikaci s finanční správou. Tu je možné používat dvěma různými způsoby. Tím prvním je pomocí statických metod, kdy se pro produkční prostředí používá jedna metoda a pro playground jiná. Druhý způsob spočívá ve vytvoření instance a použití instanční metody této instance, přičemž produkční prostředí nebo playground se určuje při vytváření instance, což usnadňuje programové určení prostředí. Tato třída zajišťuje nahlášení tržby zavoláním jedné metody s účtenkou, zmíněnou výše, jako jediným parametrem. Tato metoda pak vrací takový kód, který zákon vyžaduje uvést na účtence, FIK v případě úspěšného nahlášení nebo PKP v případě, že tržbu nelze nahlásit. Od uživatele tak není vyžadována žádná technická znalost komunikace s finanční správou.

```
// Statická metoda pro produkční prostředí
EET.sendProd( receipt );

// Playground za použití instance
EET eet = new EET( true );
eet.send( receipt );
```

4.3 Kód

4.3.1 Zpracování hodnot

Finanční správa vyžaduje všechny hodnoty v přesně daných, textových formátech, se kterými se ale velmi špatně pracuje. Z tohoto důvodu knihovna zajišťuje převod z nativních datových typů Javy do textových formátů vyžadovaných finanční správou.

Data jsou předávána knihovně ve formě instancí třídy Date, která je nativní součástí Javy. Peněžní údaje jsou předávány jako počet haléřů (datový typ long), použitím celého čísla se zabrání chybám, které by vznikaly při ukládání desetinných čísel. Tyto chyby souvisejí

s použitím binárních mantisy a exponentu s omezenými přesnostmi (datové typy float a double) a jejich následnému převodu do desítkové soustavy, například desítkové číslo 0.1 má v binární soustavě nekonečný počet číslic a není tedy možné jej uložit s absolutní přesností.

```
private String formatDate( Date date ) {
    SimpleDateFormat sdf = new SimpleDateFormat( "yyyy-MM-dd'T'HH:mm:ss'Z'"
);
    sdf.setTimeZone( TimeZone.getTimeZone( "UTC" ) );
    return sdf.format( date );
}

private String halersToString( long halers ) {
    return String.format( halers < 0 ? "%04d" : "%03d", halers
).replaceFirst( "^(.*)(.{2})$", "$1.$2" );
}
```

4.3.2 Sestavení zprávy

Vzhledem k absenci standardní knihovny pro práci s XML a SOAP zprávami na Androidu a také faktu, že formát XML zprávy je stále stejný a mění se pouze údaje v něm, je v knihovně použita šablona, do které se jen doplní potřebné údaje. Tato šablona odpovídá výsledné zprávě, jen nahlašované údaje, kontrolní součty a podpis jsou nahrazeny placeholdery ve formě komentářů a speciálních textových řetězců, které se v šabloně jinak nevyskytují. Ty jsou pak, při přípravě zprávy, nahrazeny skutečnými údaji dané tržby, případně z šablony odstraněny, pokud pro ně v dané tržbě žádný údaj neexistuje, tím vznikne validní tělo SOAP zprávy. Po doplnění kontrolního součtu a jeho podpisu do hlavičky vznikne kompletní a validní SOAP zpráva připravená k odeslání finanční správě.

```
…
<KontrolniKody>
    <pkp cipher="RSA2048"
         digest="SHA256"
         encoding="base64">
        <!--pkp-->
    </pkp>
    <bkp digest="SHA1"
         encoding="base16">
        <!--bkp-->
    </bkp>
</KontrolniKody>
…
```

4.3.3 Podepsání zprávy

Podpis je vytvořen za pomocí standardních nástrojů Javy. Z šablony se vybere tělo SOAP zprávy a z něho je, po úpravě do kanonické podoby, vytvořen kontrolní součet, který je následně doplněn do hlavičky SOAP zprávy. Kanonická podoba je vyžadována specifikací, protože je nutné, aby byla druhá strana, v tomto případě server finanční správy, schopna podpis ověřit. Ověření ale není možné, pokud každá strana pracuje s jiným řetězcem znaků, byť se liší jen o jedinou mezeru nebo znak nového řádku. Část této hlavičky s kontrolním součtem, tak jak je požadováno specifikací, je ve své kanonické formě podepsána a výsledný podpis vložen do původní zprávy. K vytvoření podpisu je použit certifikát poskytnutý koncovým uživatelem, ten jej získá od finanční správy pomocí jejich webové stránky, případně je možné využít libovolný z certifikátů playgroundu, který ale nelze použít ke skutečné evidenci tržeb, slouží pouze k testování komunikace s EET serverem finanční správy. Po vložení podpisu do zprávy je zpráva kompletní a je možné ji odeslat finanční správě.

```
private void sign() throws NoSuchAlgorithmException,
UnsupportedEncodingException, InvalidKeyException, SignatureException {
    // Získání těla z šablony (pomocí regulérního výrazu)
    String body = this.document.replaceFirst(
"[\\d\\D]*(<soap:Body[\\d\\D]*</soap:Body>)[\\d\\D]*", "$1" );
    // Kanonizace těla do podoby vyžadované specifikací pro vytvoření
kontrolního součtu
    body = this.uglifyXML( body );

    // Výpočet kontrolního součtu
    MessageDigest md = MessageDigest.getInstance( "SHA-256" );
    md.update( body.getBytes( "UTF-8" ) );
    // Nahrazení placeholderu kontrolního součtu v šabloně vypočteným
kontrolním součtem
    this.replaceTagPlaceholder( "DigestValue", Compat.base64Enc( md.digest()
) );

    // Získání tagu signedInfo, který bude podepsán
    String signedInfo = this.document.replaceFirst(
"[\\d\\D]*(<ds:SignedInfo[\\d\\D]*</ds:SignedInfo>)[\\d\\D]*", "$1" );
    // Upravení tohoto tagu do kanonické podoby, jež je specifikací
vyžadována pro vytvoření podpisu
    signedInfo = this.uglifyXML( signedInfo );

    // Samotné podepsání tagu
    Signature signature = Signature.getInstance( "SHA256withRSA" );
    signature.initSign( this.receipt.keyChain.getPrivateKey() );
    signature.update( signedInfo.getBytes( "UTF-8" ) );

    // Nahrazení placeholderu podpisu v šabloně vytvořeným podpisem
v kódování Base64
```

```
    this.replaceTagPlaceholder( "SignatureValue", Compat.base64Enc(
signature.sign() ) );
}
```

4.3.4 Komunikace s finanční správou

Komunikace probíhá pomocí standardního HTTPS spojení, v jehož rámci proběhne výměna SOAP zpráv. Finanční správě je odeslána podepsaná zpráva se všemi náležitostmi, připravená v předcházejících krocích, a od finanční správy je přijata odpověď, která se následně zpracuje a uloží pro použití uživatelem. V případě, že nahlášení skončí chybou, je uživateli vyhozena výjimka. V případě chyby na úrovni HTTPS spojení je vyhozena standardní výjimka. Pro případ chyby nahlášené finanční správou je vyhozena speciální výjimka, definovaná třídou EETServerException, která obsahuje číselný kód označující chybu, vrácený finanční správou, a také její slovní popis. Zpráva je ještě před odesláním zkontrolována, jestli obsahuje všechny povinné položky, které vyžaduje finanční správa. Pokud je nalezena chybějící povinná položka, je vyhozena výjimka EETMissingValuesException, která obsahuje název chybějící položky. Tato výjimka umožňuje snadnější opravu chyb, jelikož chyba v odpovědi finanční správy by pouze upozorňovala na nevalidní dokument a neposkytovala by žádné další užitečné informace nápomocné při nápravě chyby.

```
private static String send( EETReceipt receipt, URL url ) throws
NoSuchAlgorithmException, CertificateEncodingException, SignatureException,
InvalidKeyException, IOException, EETException {
    // Příprava SOAP zprávy
    Request request = new Request( receipt );

    // Získání pole bajtů z připravené SOAP zprávy
    receipt.request = request.getString();
    byte[] soapMessage = receipt.request.getBytes( "UTF-8" );

    // Příprava spojení
    HttpsURLConnection connection = (HttpsURLConnection)
url.openConnection();
    connection.setUseCaches( false );
    connection.setDoOutput( true );
    connection.setDoInput( true );
    connection.setInstanceFollowRedirects( false );
    connection.setRequestMethod( "POST" );
    connection.setRequestProperty( "Content-Type", "text/xml" );
    connection.setRequestProperty( "charset", "UTF-8" );
    connection.setRequestProperty( "Content-Length", "" + soapMessage.length
);

    // Odeslání SOAP zprávy serveru finanční správy
    OutputStream os = connection.getOutputStream();
```

```
    os.write( soapMessage );
    os.flush();
    os.close();

    // Vytvoření instance třídy Response, která z přijatých znakových dat
získá vrácené údaje
    Response response = new Response( connection.getInputStream() );

    // Ukončení spojení
    connection.disconnect();
    receipt.setPrvniZaslani( false );
    receipt.response = response.getString();
    receipt.fik = response.getFIK();

    // Vrácení vhodného kódu tak, jak má být uveden na účtence, FIK při
úspěšném nahlášení a PKP při neúspěšném
    return response.getFIK() != null ? response.getFIK() : receipt.pkp;
}
```

5 APLIKACE

V rámci práce byla zpracována nejen knihovna, ale i aplikace pro operační systém Android. Ten je velmi běžný například na mobilních telefonech a tabletech, ale třeba i televizorech nebo automobilech. Aplikace je nicméně přizpůsobena a otestována pouze pro mobilní zařízení, její využití v televizorech a automobilech dává jen malý smysl, byť by někdo mohl prodávat z kabiny automobilu je podstatně běžnější a univerzálnější použití mobilních zařízení jako jsou mobilní telefony a tablety.

Aplikace umožňuje uživateli výběr a vyhledávání zboží ze seznamu, který si může sám vytvořit přímo v aplikaci nebo stáhnout ze serveru, kde je předpřipravený například zaměstnavatelem pro více poboček. Server není součástí této bakalářské práce a proto byl místo něho použit statický soubor zkopírovaný na Pastebin. Ze seznamu zboží je možné sestavovat účtenky, ty pak lze nahlásit finanční správě a tisknout. Mimo to aplikace také obsahuje historii nahlášených účtenek a další funkce, nezbytné pro správné fungování takovéto aplikace.

5.1 Klíčové části kódu

Zde jsou popsány části kódu, které implementují klíčovou funkcionalitu nebo jsou z jiného důvodu zajímavé či důležité. Výjimkou je kód úzce související s uživatelským rozhraním, který bude popsán v samostatné kapitole. U každé popisové části jsou uvedeny ukázky kódu, o kterém se v dané části mluví, ty mohou být zkráceny a vytrženy z kontextu, kód v plném rozsahu je k dispozici v rámci kompletních zdrojových kódů aplikace, které jsou přílohou této práce.

5.1.1 Třída Receipt (účtenka)

Tato třída drží všechny potřebné informace jako například všechny zakoupené položky a kódy BKP, FIK, PKP.

```
private final List<Item> items;
private String bkp;
private String fik;
private String pkp;
```

Vytvoření textové reprezentace účtenky je implementována v metodě getReceiptStr. Ta projde všechno zboží na účtence a vytvoří součty, vypočítá DPH a vše vloží do textové formy. Do ní se

vloží také kódy pro EET, pokud již byly vytvořeny, a další informace požadované zákonem a poskytnuté uživatelem v nastavení. Samozřejmostí je také seznam všeho zakoupeného zboží se všemi zákonem požadovanými údaji. V rámci tvorby se počítá s určitou šířkou papíru nastavené v nastavení uživatelem ve znacích, vše je zarovnáno s ohledem na tuto šířku a to včetně zalamování řádků, aby uživatel již při náhledu účtenky viděl přesně to, co bude vytištěno. Výsledek je pak vrácen jako textový řetězec připravený pro další použití.

```
// Počet kusů každého zboží
HashMap<String, Integer> amounts = new HashMap<>();
// Celková cena
int sum = 0;
// Součet jednotlivých základů DPH
int[] vats = new int[VAT.values().length];
// Seznam položek (pro přehlednost sesumarizován a seřazen)
List<Item> items = new ArrayList<>();
// Obsahuje „-“ při rušení tržby
String negative = this.multiplier == 1 ? "" : "-";
// Řetězec s textem účtenky
String str = "";

// Příprava sesumarizovaného zboží a součtů
for (int i = 0; i < this.items.size(); ++i) {
    // Získání zboží
    Item item = this.items.get(i);
    // Načtení již sesumarizovaného počtu kusů tohoto zboží
    Integer current = amounts.get(item.getName());
    // První výskyt tohoto zboží se přidá do seznamu zboží
    if (current == null) {
        current = 0;
        items.add(item);
    }

    // Inkrementace množství daného zboží
    amounts.put(item.getName(), current + 1);
    // Přičtení ceny zboží k celkové ceně
    sum += item.getPrice();
    // Přičtení ceny zboží k součtu sazby DPH
    vats[item.getVAT().ordinal()] += item.getVATH();
}

…

// Seřazení pro přehlednost
Collections.sort(items);
// Přidání sesumarizovaného zboží
for (int i = 0; i < items.size(); ++i) {
    // Získání zboží
    Item item = items.get(i);
    // Získání množství daného zboží
    int amount = amounts.get(item.getName());
    // Název zboží s celkovou cenou (zarovnáno a zformátováno)
```

```
    str += RSU.align(item.getName(), negative + item.getPriceRawStr(amount) +
" kč");
    // Další údaje na novém řádku: množství, cena za kus, DPH
    str += "  " + amount + " ks, " + negative + item.getPriceRawStr() + "
kč/ks, " + item.getVAT().toString() + " DPH\n";
}

…

// Vypsání součtů (bez DPH, samotné DPH a celkový součet), zarovnáno a čísla
zformátovány pro přehlednost
str += RSU.align("Součet bez DPH", negative + Item.priceFormat.format((sum -
sumVAT) / 100.0) + " kč");
str += RSU.align("DPH", negative + Item.priceFormat.format(sumVAT / 100.0) +
" kč");
str += RSU.align("Součet", negative + Item.priceFormat.format(sum / 100.0) +
" kč");

…

// Zalomení řádků podle šířky nastavené uživatelem
return str.replaceAll("(.{" + Settings.getReceiptWidth() + "})\\n?", "$1\n");
```

5.1.2 Zpětná vazba pro uživatele

Uživatel je informován o událostech, jako jsou chyby nebo dokončení komunikace s finanční správou a její výsledek, pomocí takzvaných snackbarů, jedná se o proužek s textem ve spodní části displeje. Ten byl vybrán z několika důvodů. Tím prvním a nejdůležitějším je, že je součástí Material designu, uživatelé jej tedy již znají z dalších aplikací a také zapadá do designu zbytku aplikace. Jeho další výhodou je jeho pozice ve spodní části displeje, která příliš uživatele příliš neomezuje, byť nevýhodou je snazší přehlédnutelnost než například u klasického upozornění, jenž se zobrazí uprostřed displeje a zablokuje aplikaci dokud jej uživatel nezavře. Poslední výhodou je implementace snackbaru v samotném Androidu, což umožňuje programátorovi zobrazit jej pomocí jednoho řádku kódu.

Zpětná vazba je v aplikaci předávána pomocí handleru, jedná se o součást Androidu, která zajišťuje shromažďování a zpracovávání zpráv. Handler je v rámci aplikace předán různým třídám, jenž jej pak využívají například pro hlášení chyb uživateli nebo pro aktualizace dat po dokončení nějaké operace.

```
// Zpracování zprávy obdržené handlerem
void handleMessage(Message msg) {
    switch (Messages.values()[msg.what]) {
        case exception:
            // Text výjimky je zobrazen uživateli
```

```
            Exception e = (Exception) msg.obj;
            if (e != null) {
                Snackbar.make(this.getWindow().getDecorView().getRootView(),
e.getMessage(), Snackbar.LENGTH_LONG).setAction("Action", null).show();
            }
            this.receiptUpdated();
            break;
        case receiptChanged:
            // Informuje ostatní části aplikace, že je potřeba aktualizovat
pohledy s údaji na účtence
            this.receiptUpdated();
            break;
        …
}
```

5.2 Uživatelské rozhraní

Uživatelské rozhraní je vytvořeno ve stylu Material design, který je prosazován společností Google, vlastníkem Androidu. Tento styl byl zvolen s ohledem na to, že je použit v GUI Androidu a mnoha dalších aplikacích. Je také součástí standardních knihoven Androidu a tudíž je k dispozici všem nativním aplikacím. Jeho použití tedy značně usnadňuje vývoj v tomto stylu a zároveň zajišťuje jednotnost designu napříč aplikacemi. Samotné GUI je pak sestává z jedné aktivity, ta obsahuje menu, záhlaví a kontejner na fragmenty. Různé části GUI jsou pak v samostatných fragmentech, které jsou dle potřeby vyměňovány v kontejneru. Koncept aktivit a fragmentů bude blíže popsán v následujících kapitolách.

5.2.1 Aktivity

Základním prvkem aplikace na operačním systému Android jsou aktivity, každá aplikace má určitou aktivitu jako svůj vstupní bod. Aktivita je třída starající se mimo jiného o životní cyklus aplikace nebo její části a o okno použité k vykreslování GUI, to je správcem oken na Androidu, který je zejména dlaždicový, umístěno přes celou obrazovku nebo její část. Je také možné vytvářet plovoucí okna, která zabírají jen část displeje a lze je po něm přesunovat, byť to není pro Android příliš typické. Další možností jsou aktivity, které nemají žádná okna, ty slouží pro provádění úloh na pozadí. Každá aktivita pak může obsahovat libovolné prvky, ale nemůže do ní být vložena jiná aktivita, k takovémuto členění je možné využít například fragmenty. [24]

5.2.2 Fragmenty

Fragmenty jsou nejhrubšími stavebními kameny GUI v rámci aktivity, zapouzdřují související funkcionalitu uvnitř aplikace, jsou na sobě nezávislé a také jsou znovupoužitelné. Tím je

umožněno je velmi jednoduše umisťovat na různá místa v aplikaci a vyměňovat je na těchto místech. Tyto činnosti jsou v rámci Androidu zajišťovány třídami FragmentManager a FragmentTransaction, jenž poskytují metody pro práci s fragmenty v transakcích. Tyto transakce mohou být uloženy do historie a pak se lze vrátit k předchozím stavům, umožňují také například změny animovat. S výměnou fragmentů úzce souvisí jejich životní cyklus, který je řešen v rámci systému Android a programátor se o něj nemusí nijak starat, nicméně je užitečné jej znát a jsou poskytnuty metody, jenž umožňují reagovat na změny stavu fragmentu. Jako příklad stavu lze uvést stav resumed, v němž se fragment nachází v okamžiku, kde je zobrazen, nebo stav stopped, ve kterém je fragment, jenž není zobrazen. Ke každému stavu se pojí jedna metoda, umožňující reakci na změnu do daného stavu, nebo i více metod, pokud je tato změna komplexnější, jako například při vytváření fragmentu. [25]

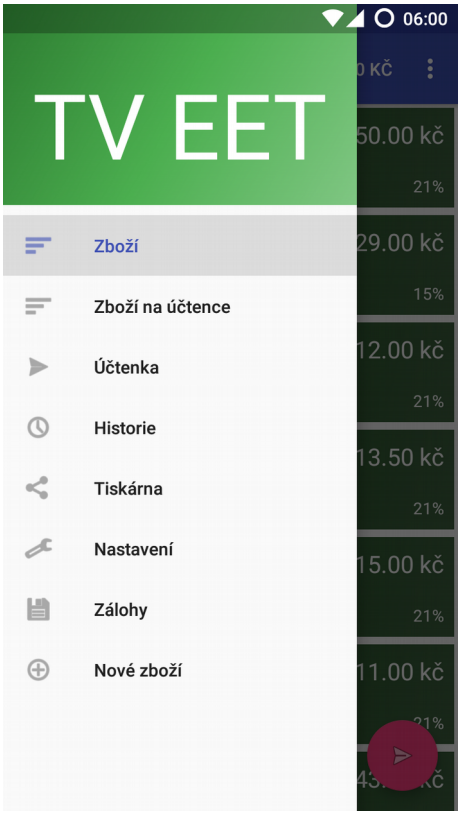
Většina GUI je rozčleněna do fragmentů, z nichž se každý stará o určitou část uživatelského grafického rozhraní. Součástí těchto fragmentů není menu a záhlaví, ty jsou vytvořeny pomocí knihovny Androidu. Například pokud uživatel v nastavení otevře menu a vybere zálohy, bude vyměněn fragment nastavení za fragment záloh. V rámci aplikace jsou fragmenty umístěny v prvku, který vyplňuje většinu obrazovky, jediné, co není součástí těchto fragmentů, je menu a horní panel s cenou (a samozřejmě statusbar, který není vůbec součástí aplikace).

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout …>
    <LinearLayout …>
        <android.support.design.widget.AppBarLayout …>
            <android.support.v7.widget.Toolbar …/>
        </android.support.design.widget.AppBarLayout>
        <FrameLayout android:id="@+id/fragments" …>
            <!-- Sem jsou dynamicky umisťovány fragmenty -->
        </FrameLayout>
    </LinearLayout>
    <android.support.design.widget.FloatingActionButton …/>
</android.support.design.widget.CoordinatorLayout>
```

```
// Zahájení transakce
FragmentTransaction ft = this.getSupportFragmentManager().beginTransaction();

// Nadefinování operací v transakci, v tomto případě nahrazení stávajícího
fragmentu fragmentem f (např. fragment zálohy z předchozího textu).
R.id.fragments identifikuje prvek obsahující fragmenty.
ft.replace(R.id.fragments, f);
```

```
// Provedení transakce
ft.commit();
```

5.2.3 Menu



Obrázek 1

Menu je ve formě tzv. draweru, vysunuje se z levé strany displeje buď tažením prstu nebo pomocí hamburgerového tlačítka (ikona se třemi vodorovnými čárkami nad sebou) v levém horním rohu. V něm se nachází za prvé hlavička s názvem aplikace a za druhé jednorozměrný seznam umožňující navigaci v rámci aplikace. Jedná se o konvenční formu menu velmi běžnou nejen na Androidu, ale i v aplikacích pro jiné systémy.

Z programového hlediska je menu vytvořeno za pomocí standardní knihovny Androidu dle XML definice, která vyjmenovává všechny položky menu, včetně jejich názvů a ikon. To nejen zjednodušuje práci programátorovi tím, že nemusí vytvářet menu ručně z jednotlivých prvků a kontejnerů, ale také automaticky zajišťuje vzhled odpovídající Material designu, což krásně pasuje do zbytku aplikace a je i známé pro většinu uživatelů z jiných aplikací, čímž jim je usnadněna orientace v menu této aplikace. Ve zdrojovém kódu je pak řešeno pouze vyměňování jednotlivých fragmentů, které odpovídají položkám menu, tak, jak uživatel

položky z menu vybírá. Případně při změně fragmentu jiným způsobem je zvýrazněna odpovídající položka menu, aby, i přestože na ni uživatel neklikl, vybraná položka v menu vždy odpovídala zobrazenému fragmentu.

```
<menu
    xmlns:android="http://schemas.android.com/apk/res/android">

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/menu_items"
            android:icon="@android:drawable/ic_menu_sort_by_size"
            android:title="Zboží"
            />
        <item
            android:id="@+id/menu_receipt_items"
            android:icon="@android:drawable/ic_menu_sort_by_size"
            android:title="Zboží na účtence"
            />
…
```

5.2.4 Zboží

Obrázek 2

Seznam prodávaného zboží vytvořeného uživatelem (postup vytváření zboží bude popsán níže). U každé položky je zobrazen název, ten může být i více řádkový, v případě, že se nebude

vejít do položky (cca. více než 2 řádky), bude tato položka zvětšena, aby nemusel být název nikdy zkracován. Na pravé straně se nachází cena s přesností na haléře a DPH vyjádřené v procentech. Barvu každého zboží si uživatel může vybrat z 24 barev rovnoměrně rozložených podle odstínu s 15° odstupem. Zboží je možné filtrovat podle názvu a nebo kategorie, přičemž kategorie jsou napovídány při psaní. Zboží se přidává na účtenku jednoduchým stisknutím požadované položky, zpětnou vazbu zajišťuje šedý pruh s textem ve spodní části s názvem a cenou přidaného zboží. Dlouhým stiskem je možné přejít přímo k úpravě daného zboží, fragment sloužící k úpravě zboží bude popsán později. Součet cen všech přidaných položek je zobrazen v pravém horním rohu. Poslední zajímavou věcí je plovoucí tlačítko, anglicky floating action button, v pravém dolním rohu, to otevírá pohled na účtenku, kde je možné ji, krom jiného, nahlásit finanční správě a vytisknout pro zákazníka.

Vzhled všech fragmentů je nadefinován pomocí XML dokumentů, které obsahují poskládané jednotlivé prvky a kontejnery. Nicméně je možné je přidávat i programově, to je využito právě v tomto fragmentu, kam se zboží přidává až za běhu aplikace na základě seznamu zboží vytvořeného uživatelem. Samozřejmostí je i odebírání prvků za běhu aplikace, v aplikaci například při smazání nějakého zboží. Jako kontejner pro jednotlivé zboží je použit RecyclerView, který zajišťuje efektivní zobrazování prvků včetně rolování, aniž by bylo nutné tyto záležitosti řešit v kódu aplikace. V aplikaci je nicméně potřeba si držet informace o pořadí zboží, jelikož metodě zpracovávající kliknutí, jejíž ukázka je v následující kapitole, je předávána pouze pozice prvku, na který uživatel klikl.

```
<!-- Tento kontejner skládá prvky svisle pod sebe -->
<LinearLayout
    android:orientation="vertical"
    …
    >
    <!-- Tento kontejner skládá prvky vodorovně vedle sebe -->
    <LinearLayout
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:orientation="horizontal"
        …
        >
        <!-- Textové pole pro vyhledávání s napovídáním -->
        <AutoCompleteTextView
            android:id="@+id/search"
            android:layout_weight="1"
            android:inputType="text"
            …
```

```
            />
        <!-- Tlačítko hledat -->
        <Button
            android:id="@+id/search_btn"
            android:layout_weight="0"
            android:text="Hledat"
            …
            />
    </LinearLayout>
    <!-- Seznam zboží -->
    <android.support.v7.widget.RecyclerView
        android:id="@+id/list"
        app:layoutManager="LinearLayoutManager"
        …
        >
    </android.support.v7.widget.RecyclerView>
</LinearLayout>
```

5.2.5 Zboží na účtence

Obrázek 3

Zde je možné vidět seznam zboží, které již bylo přidáno na účtenku. Vzhled a chování seznamu je totožné jako v seznamu všeho dostupného zboží, rozdílem je, že po stisku se zboží, místo přidání na účtenku, z ní odstraní.

```
// Vyvoláno po kliknutí na zboží
public void onClick(View view) {
    // Odstraní zboží z účtenky, které se nachází na pozici, na kterou klikl
uživatel
```

```
    Item item = ReceiptItemsListenerFactory.this.receipt.remove(position);
    // Zpětná vazba pro uživatele
    Snackbar.make(view, "Odebráno: " + item.getBrief(),
Snackbar.LENGTH_LONG).setAction("Action", null).show();
}
```

5.2.6 Účtenka

Obrázek 4

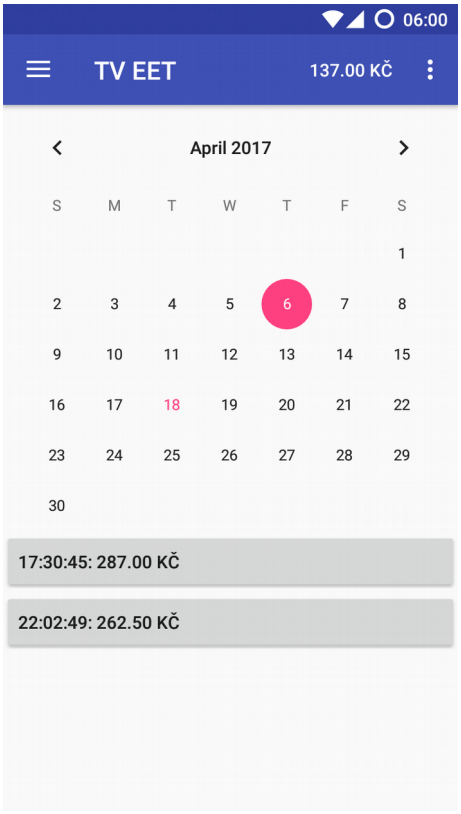
Tento fragment umožňuje uživateli zobrazit si účtenku tak, jak bude vypadat vytištěná a to včetně přesné šířky a zarovnání, pokud to bylo správně nastaveno v nastavení. V případě, že uživatel používá kódování ASCII, které nepodporuje české znaky a proto jsou v něm nahrazovány jejich verzemi bez háčků a čárek, budou všechna diakritická znaménka odstraněna i v tomto náhledu účtenky. Dále umožňuje nahlášení tržby finanční správě, po nahlášení, ať už úspěšném či nikoli, se do účtenky doplní kódy EET, tedy BKP a PKP, v případě neúspěšného nahlášení, nebo BKP a FIK, v případě úspěšného nahlášení. Účtenku je také samozřejmě možné vytisknout a to bez ohledu na to, jestli bylo nahlášení úspěšné, neúspěšné nebo vůbec neproběhlo. Pokud ale nahlášení neproběhlo, nejedná se o platnou EET účtenku, jelikož neobsahuje kódy EET, vydání pouze takové účtenky zákazníkovi by bylo porušením zákona o elektronické evidenci tržeb, byť by účtenka byla platná například při

reklamaci. Tisk je možné provést vícekrát i dodatečně po obnovení z historie, která bude popsána níže. V menu (svislé tři tečky v pravém horním rohu) je možnost zrušení tržby, tedy nahlášení záporné tržby finanční správě, pro případ chybného nahlášení tržby, vrácení peněz a pod.

Z programového hlediska je tento fragment velmi jednoduchý. Využívá třídu Receipt, která udržuje všechny informace a zajišťuje jak nahlašování (pomocí knihovny) tak i tvorbu textové verze účtenky a tisk. Samotný náhled účtenky je zobrazen v textovém pohledu, text v něm je zarovnán k levému okraji a používá se písmo se stejnou šířkou všech znaků. Toto nastavení svým vzhledem velmi dobře odpovídá textu, který by byl vytištěn skutečnou tiskárnou na termocitlivý papír, což je nejběžnější způsob tisku účtenek. Vzhledem k omezené výšce displejů mobilních zařízení je tento textový pohled umístěn v rolovacím kontejneru, který umožňuje rolování tažením prstu, a z designových důvodů je tento kontejner vodorovně vystředěn, takže text není nalepen přímo na levý okraj displeje, pokud je k dispozici dostatek prostoru.

```
// Nahlášení tržby pomocí knihovny
try {
    switch (Settings.getServer()) {
        case play:
            // Vývojové prostředí (tržby nejsou registrovány)
            EET.sendPlay(this.receipt.eetReceipt);
            break;
        case prod:
            // Produkční prostředí (tržby se registrují)
            EET.sendProd(this.receipt.eetReceipt);
            break;
    }
    // Knihovna hází výjimku při jakékoli chybě, pokud se program dostane
sem, znamená to, že tržba byla úspěšně nahlášena

    // Uložení informace o tom, že byla úspěšně nahlášena, do účtenky
    this.receipt.onSubmit();

    // Uložení nahlášené tržby do historie
    Receipts.addReceipt(this.receipt);

    // Zpětná vazba pro uživatele
    this.handler.sendMessage(Messages.generateMessage("Tržba byla úspěšně
nahlášena"));
} catch (Exception e) {
    // Uživateli je zobrazena chybová hláška na základě vyhozené výjimky
    this.handler.sendMessage(Messages.generateMessage(e));
}
```

5.2.7 Historie



Obrázek 5

V tomto fragmentu je možné si prohlížet historii a obnovovat starší účtenky. V horní části se nachází kalendář, který umožňuje výběr dne, a pod ním je seznam účtenek nahlášených v daný den. U každé účtenky je zobrazen čas, s přesností na sekundu, tak jak je byl nahlášen finanční správě, a cena. Po rozkliknutí je možné účtenku znovu vytisknout, zrušit její nahlášení finanční správě a nebo ji použít jako základ pro jinou účtenku, po změně jakéhokoli údaje na ní, bude nahlášena a uložena do historie jako zcela nová účtenka. V historii je také kompletní seznam zrušených účtenek, protože systém finanční správy pro ně nemá zvláštní zacházení, není potřeba žádné zvláštní zacházení ani v aplikaci, jedná se jen a pouze o účtenky se zápornými částkami.

K ukládání účtenek v telefonu je využívána databáze Androidu, ta je součástí systému a k dispozici na všech zařízeních. V rámci databáze jsou ukládány jednotlivé účtenky jako textové řetězce ve formátu JSON, tento způsob byl zvolen ze dvou důvodů. Z hlediska aplikace totiž nemá rozlišování jednotlivých vlastností účtenek v rámci databáze žádný smysl. Uživatel si vybere datum v kalendáři a zobrazí se mu seznam účtenek z daného dne. Z nich si

pak může vybrat tu, kterou potřebuje. Druhým důvodem je, že export do formátu JSON je zároveň použit pro vytváření záloh, což značně zredukovalo množství kódu a času stráveného vývojem.

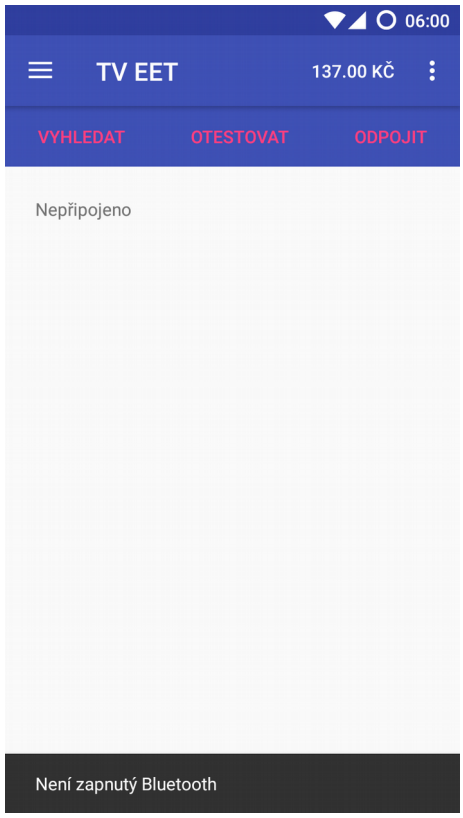
SQL kód vytvářející tabulku s účtenkami:

```
CREATE TABLE receipts_history (
        id INTEGER PRIMARY KEY,
        year INTEGER,
        month INTEGER,
        day INTEGER,
        json TEXT
);
```

Metoda v Javě která ukládá jednu účtenku do historie:

```
void addDay(int year, int month, int day, String json) {
    // Otevření databáze pro zápis
    SQLiteDatabase db = this.getWritableDatabase();

    // Příprava hodnot, které budou vloženy do databáze (rok, měsíc, den
a účtenka v JSON)
    ContentValues values = new ContentValues();
    values.put("year", year);
    values.put("month", month);
    values.put("day", day);
    values.put("json", json);

    // Samotné vložení záznamu do tabulky
    db.insert("receipts_history", null, values);
    db.close(); // Uzavření databáze
}
```

5.2.8 Tiskárna



Obrázek 6

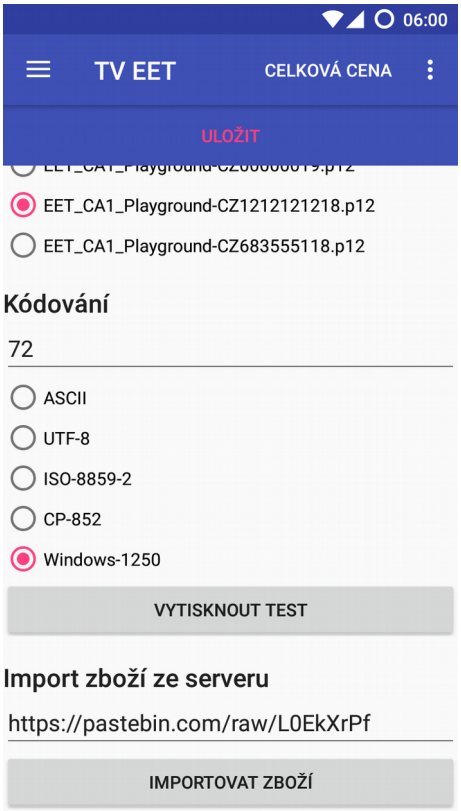
Fragment tiskárna umožňuje uživateli správu jeho tiskáren, od vyhledání, připojení, tisk testovací stránky až po její odpojení. Jakákoli chyba je uživateli zobrazena ve spodní části displeje (tak jako na obrázku kde je chyba, že není zapnutý Bluetooth). Po vyhledání se zobrazí jednoduchý seznam tiskáren a po kliknutí na jednu z nich se aplikace k ní připojí. Tlačítko otestovat umožňuje tisk testovací stránky tzv. selftest, ten je součástí každé tiskárny, ale jeho úspěšné vytištění je důkazem funkčního spojení mezi aplikací a tiskárnou. Odpojit jednoduše přeruší spojení s tiskárnou. Mimo výše uvedené si aplikace ještě automaticky ukládá informace o připojené tiskárně a pokud při startu najde tiskárnu, ke které byla naposledy připojena, tak se k ni připojí automaticky, aniž by ji uživatel musel vyhledávat.

Při práci s tiskárnou jsou využívány nástroje Androidu, které poskytují dobrou podporu pro komunikaci. Pomocí knihoven Androidu je možné velmi jednoduše získat seznam zařízení v dosahu a k některému se připojit (vizte metodu níže). Po připojení je dobré nastavit kódování, aby bylo možné tisknou české znaky. Kódování se nastavuje pomocí sekvence bajtů a čísla označujícího dané kódování, to se bohužel liší na různých zařízeních, není proto možné

jednoznačně spárovat kódování v Javě a na tiskárně, je nutné aby ho zadal uživatel podle dokumentace tiskárny. Samotný tisk probíhá pomocí streamu, do kterého se zapisují bajty a Android zajišťuje přenos mezi dat mezi zařízeními. Tiskárna, která byla použita při vývoji, neposkytuje žádnou zpětnou vazbu, není tedy možné uživatele informovat o tom, jestli byl tisk úspěšný nebo proč úspěšný nebyl. Jediné, co je možné zjistit, je, jestli byla data úspěšně doručena tiskárně, a uživatel samozřejmě vidí, jestli tiskárna účtenku vytiskla či nikoli. Ukončení komunikace pak probíhá jednoduchým uzavřením streamu a socketu.

```
// Připojení k zařízení vybranému uživatelem
void connect(BluetoothDevice device) throws IOException {
    …

    // Získání UUID z tiskárny pro komunikaci
    ParcelUuid[] uuids = device.getUuids();
    UUID uuid = uuids[0].getUuid();

    …

    // Příprava socketu se získaným UUID
    this.socket = device.createRfcommSocketToServiceRecord(uuid);

    // Připojení k tiskárně
    this.socket.connect();

    // Uložení streamu pro pozdější posílání dat
    this.outputStream = this.socket.getOutputStream();

    // Nastavení použitého kódování
    this.setCodepage(Settings.getCodepage());

    …
}
```

5.2.9 Nastavení



Obrázek 7

Nastavení je poměrně sebe vysvětlující fragment, většinu položek jako DIČ nebo adresu stačí jednoduše vyplnit. Pro usnadnění vytváření zboží nebo jeho sdílení mezi více zařízeními je tu možnost jej importovat ze webového serveru. Stačí jednoduše vyplnit adresu, na které se nachází seznam zboží, a stisknout tlačítko importovat zboží. Jedinou záludností nastavení je kódování. Problém s kódování je, že neexistuje žádný respektovaný standard číslování různých znakových sad, stejné kódování je tudíž na různých tiskárnách pod různými čísly, a není se ani možné spolehnout na to, že tiskárna bude podporovat určité kódování. Z těchto důvodů je v nastavení celkem nepraktická možnost zvolit číslo znakové sady, to může být k dispozici v dokumentaci tiskárny nebo na testovací stránce. Pro kontrolu má uživatel možnost zvolenou kombinaci otestovat ještě před uložením, test se skládá ze všech českých znaků s diakritikou na jednom řádku, je-li tiskárna schopna je všechny vytisknou na jeden řádek. Výchozím nastavením je kódování ASCII, které je standardem a základem všech běžně používaných kódování, a tudíž se předpokládat, že bude fungovat na všech tiskárnách. Toto kódování nepodporuje české znaky a proto je při jeho používání odstraňována diakritika.

Třída fragmentu nastavení pojmenovaná SettingsFragment úzce spolupracuje s třídou Settings. SettingsFragment se zabývá interakcí s uživatelem, ukládání jím zadaných hodnot a zobrazováním stávajících hodnot v GUI. Třída Settings obsahuje pouze statické metody a zpřístupňuje uložená nastavení napříč aplikací. K samotnému uložení dat je využívána třída SharedPreferences, jenž je součástí Androidu, ta umožňuje nativní ukládání dat ve formátu klíč hodnota, což je velmi vhodné pro uchovávání nastavení. [26]

Většina hodnot v nastavení je ve formě textových řetězců, zadávaných do textových polí, z nichž jsou jednoduše čteny a ukládány. Stejně jednoduché je zpracování číselných hodnot z číselných polí, jen je nutné převést textový řetězec v poli na číslo. Číselná pole v Androidu automaticky zajišťují, že do nich nebude napsáno nic jiného nežli čísla, byť jsou čísla v nich stále textovými řetězci, a také uživateli automaticky zobrazují číselnou klávesnici, což je uživatelsky přívětivé chování. Ověřovací režim se nastavuje pomocí prvku přepínače, jehož hodnotou je boolean. Ostatní údaje jsou vybírány ze seznamů, ty jsou spárovány buď s předem určenými hodnotami, jako je výčet podporovaných kódování nebo seznam serverů finanční správy, nebo se soubory na disku v případě výběru certifikátu.

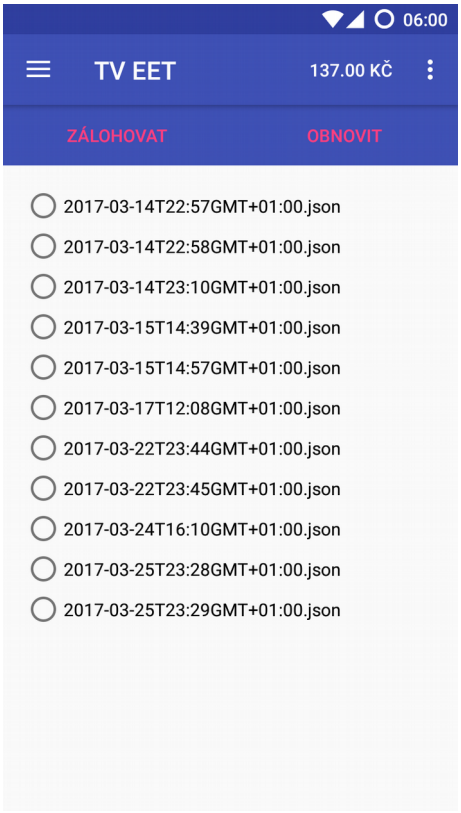
Hodnoty získané od uživatele jsou ukládány až po stisknutí tlačítka uložit, do té doby jsou uloženy pouze dočasně ve svých prvcích a při ukončení aplikace jsou změny ztraceny. Při ukládání jsou přečteny všechny vstupy a zpracované hodnoty postupně uloženy do úložiště nastavení. Změny v úložišti jsou ukládány asynchronně, aby se předešlo zbytečnému blokování GUI při čekání na zapsání dat do úložiště.

Zvláštní pozornost je věnována ukládání hesla certifikátu. V případě, že je daná verze Androidu podporuje bezpečné úložiště klíčů, je vygenerován klíč a heslo je jím zašifrováno a dešifrováno. Klíč je do úložiště uložen Androidem, který se stará o jeho bezpečné uložení. Na starších verzích Androidu bezpečné úložiště neexistuje, vygenerovaný klíč by musel být uložen nezabezpečeně v nastavení nebo by muselo být v aplikaci heslo pro šifrování, které by bylo stejné pro všechny uživatele. Ani jedna z těchto možností není příliš bezpečná, protože je klíč k dešifrování uložen v čitelné podobě spolu se zašifrovanými daty. Z tohoto důvodu je heslo uloženo jako prostý text, to nicméně neznamená, že heslo není nijak zabezpečené. Android nedovoluje číst nastavení jedné aplikace aplikací jinou a ani uživatel nemá k těmto datům přístup, pokud nemá oprávnění uživatele root a to běžně není k dispozici. Aby se k nim

někdo dostal, musel by najít zranitelnost v systému nebo získat oprávnění uživatele root, což na většině zařízení lze pouze při nalezení zranitelnosti, která by to umožnila.

```
public void run() {
    try {
        // Stažení dat z adresy (this.url)
        URL url = new URL(this.url);
        HttpURLConnection con = (HttpURLConnection) url.openConnection();

        // Pokud byl vrácen jiný kód než OK, tak bude uživatel informován
o chybě a nebude se pokračovat
        int responseCode = con.getResponseCode();
        if (responseCode != 200) {
            this.handler.sendMessage(Messages.generateMessage("Chyba
komunikace (" + responseCode + ")"));
            return;
        }

        // Předání stažených dat k importu třídě Settings
        Settings.importItems((new
Scanner(con.getInputStream())).useDelimiter("\\A").next());

        // Zpětná vazba pro uživatele
        this.handler.sendEmptyMessage(Messages.itemsChanged.ordinal());
        this.handler.sendMessage(Messages.generateMessage("Zboží bylo úspěšně
importováno"));
    } …
}

static void importItems(String json) throws JSONException,
UnsupportedImportItemsVersion {
    // Vytvoření objektu z textového řetězce
    JSONObject obj = new JSONObject(json);

    // Zpracování dat podle jejich verze
    int version = obj.getInt("version");
    switch (version) {
        case 1:
            // Vytvoří instanci třídy Items ze stažených dat, tím se ověří
jejich validita, pokud něco není v pořádku, bude vyhozena výjimka
            new Items(json);

            // Uložení staženého zboží do nastavení
            Settings.setItems(json);
            break;
        default:
            throw new UnsupportedImportItemsVersion(version);
    }
}
```

5.2.10 Zálohy



Obrázek 8

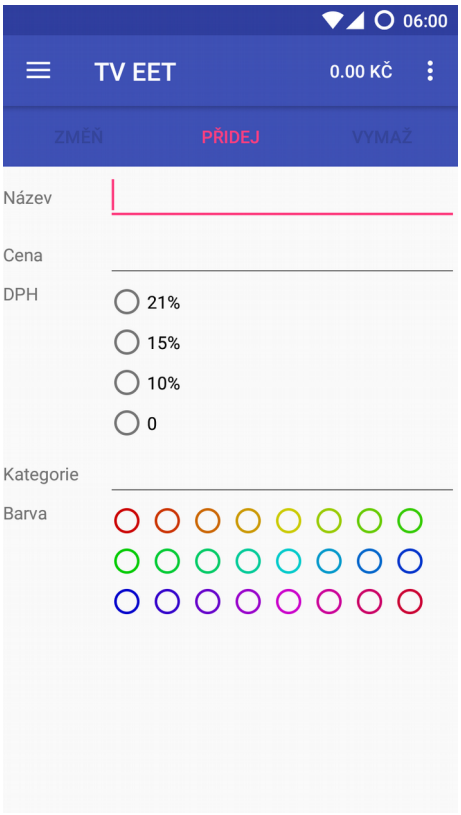
Tento fragment slouží ke tvorbě a obnově záloh. Zálohy obsahují jak nastavení tak všechno zboží i kompletní historii a jsou uloženy ve formátu JSON. Ten byl zvolen s ohledem na to, že je běžně používaný a čitelný pro člověka, pokud dotyčný zná jeho syntaxi. Další výhodou tohoto formátu je, že je podporován přímo v systému Android. Jeho formát také lépe odpovídá způsobu ukládání dat v Javě na rozdíl od XML, obsahuje tedy pole a objekty, což usnadňuje převod dat z Javy do JSON a zpět. Zálohy se ukládají do úložiště zařízení, na kterém aplikace běží, je ale samozřejmě možné je zkopírovat na jiné médiu a zpět nebo na úplně jiné zařízení. Název souboru je ve formátu ISO 8601 data s příponou json, nicméně pokud si uživatel zálohu přejmenuje, bude v seznamu záloh stále k dispozici s novým názvem, jen ji nesmí přesunout mimo složku záloh.

```
{
  "version": 1,
  "settings": {
    "receiptWidth": 32,
    "itemsImportURL": "https://pastebin.com/raw/L0EkXrPf",
    "VAT1": 21,
    "VAT3": 10,
```

```
    "VAT2": 15,
    "address": "Falešná 360, Kolín VII",
    "DIC": "CZ1212121218"
  },
  "history": [
    {
      "multiplier": 1,
      "submitTime": "2017-04-20T20:16:05+0200",
      "bkp": "2EFAE453-3C296FCB-0707875C-19C78100-9981160A",
      "pkp":
"gvjx4QTkwVo7qDQelEPFjT44QYYabgiKm13kkpfcMKh1aWiHePHFetSZvp1iXj5Y8tWhr9fu1XEJ
\nKYBC4+YESrpX3Cd04VU5Obd32rDiDXyQ/FRC/QiDswengrTiv7lH9B0FAk7395OtgBoygo1IErk
1\npTnwqekqOam/J+5m+2PMFn4VEVDJ/e+9R+11ApzXe1j7QSxGr8UbiT1WQRkf9Fw/dQGKKvmtKZ
hS\nficlO+yPtWWm/Zqllq3ev79vyhvJQW8MSdAGlvePcaOXEh2ZhR40KgecgsaqR7QPjgr06i4o+
bM9\n02qEDOZBuNUXAZe1axrACYSnStz09Ag2YzJXxQ==\n",
      "fik": "2d0c8412-4e69-45a0-9978-8c53064e297b-ff",
      "items": [
        {
          "name": "Example 2",
          "price": 3500,
          "VAT": 1,
          "category": "Even"
        },
        {
          "name": "Example 3",
          "price": 5000,
          "VAT": 1,
          "category": "Odd"
        }
      ]
    }
  ]
}
```


Jedná se pouze o ukázku formátu zálohy, kompletní záloha obsahuje podstatně více údajů.

5.2.11 Nové zboží



Obrázek 9

Aplikace umožňuje tvorbu i úpravy zboží. Každé zboží má 5 údajů. Název, který není nijak omezen, může se skládat z libovolného množství jakýchkoli znaků. Cena se zadává jako celé nebo desetinné číslo, jako oddělovač je možné použít jak desetinnou čárku tak i tečku, nemůže ale mít více než 2 desetinná místa, 3. a další jsou ignorována. Sazba DPH se vybírá ze seznamu, kde jsou všechny zákonem určené sazby (uživatel je může změnit v nastavení, ale výchozí sazby odpovídají současnému znění zákona). Kategorie je libovolný textový řetězec, existující kategorie jsou napovídány, ale není potřeba je předem vytvářet. Posledním údajem je barva, uživatel může každé položce přiřadit jednu ze 24 barev. Ty mají odstíny rovnoměrně rozložené po 15 °. Po dlouhém stisku položky v seznamu zboží se zobrazí tento fragment a předvyplní se v něm hodnoty vybraného zboží. Toto zboží je následně možné smazat tlačítkem smazat, upravit a uložit změny tlačítkem změň a nebo na jeho základě vytvořit nové zboží tlačítkem přidej.

V kódu se jedná zejména o prvky nadefinované pomocí XML a načtení, ověření a použití údajů zadaných uživatelem z těchto položek. Název a kategorie jsou pouze textové řetězce, které

mohou být jednoduše použity bez jakýchkoli úprav, stejně jednoduše se pracuje s DPH a barvou, jenž jsou vybírány z předem známých seznamů, jejichž hodnoty jsou spárovány s enumy používanými v rámci aplikace. Složitější je to u ceny, která musí být upravena z formátu čitelného pro uživatele, do formátu používaného v kódu aplikace a API knihovny, kde se pracuje s celým číslem vyjadřujícím počet haléřů. Například údaj 49,90 zadaný uživatelem je potřeba převést na celé číslo 4 990, se kterým se bude dále pracovat v aplikaci. Krom výše zmíněného je ale ještě potřeba vytvořit vyhledávací řetězec, ten obsahuje název, kategorii, cenu a DPH. Řetězec je převeden na malá písmena, aby bylo umožněno vyhledávání bez ohledu na velikost písmen, stejně tak je vždy na malá písmena převeden vyhledávaný text. Účelem vyhledávacího řetězce usnadňuje a do určité míry zefektivňuje vyhledávání položek, jelikož není potřeba procházet více údajů a upravovat je při vyhledávání, ale je to již hotové a stačí jednoduše otestovat jeden řetězec.

Při upravování existujících prvků jsou všechny údaje vyplněny. Název a kategorii stačí opsat, jelikož se jedná o jednoduchý textový řetězec. Do adaptéru pole s kategorií je ještě nastaven seznam kategorií zboží, z tohoto seznamu jsou pak uživateli automaticky doplňovány již existující kategorie. Barva a DPH jsou označeny v odpovídajících seznamech, každá zobrazená hodnota totiž odpovídá určité hodnotě enumu používaného v aplikaci. Pro zobrazení ceny, tedy její převedení z celého čísla haléřů na desetinné číslo korun, je použita stejná metoda, která se využívá při zobrazování ceny ve zbytku aplikace, jen není přidáváno označení měny. Důvodem je fakt, že tento údaj je v GUI označen jako číslo, což znamená že Android automaticky zobrazuje číselnou klávesnici a zároveň kontroluje formát tak, aby nebylo možné zadávat nečíselné hodnoty.

Zboží je v aplikaci udržováno v abecedně seřazeném seznamu, ze kterého ho lze, za použití tlačítek smazat a přidat v tomto fragmentu, smazat a nebo do něho přidat. Při úpravě zboží nejsou úpravy prováděny přímo v instanci zboží, ale je vytvořena zcela nová instance. Původní instance je ze seznamu zboží odstraněna a nová je do něho přidána. Absence možnosti úprav hodnot jednotlivých instancí zajišťuje, že se při úpravách zboží nemůže změnit například zboží, které již bylo přidáno na účtenku, změní se vždy jen zboží v seznamu dostupného zboží.

```
Item(String name, String priceStr, VAT vat, ItemColor color, String category)
{
```

```
    // Rozdělení ceny zadané uživatelem podle desetinné čárky (případně
tečky)
    String[] priceParts = priceStr.replaceAll("[^\\d,.]", "").split("[,.]");
    // První část představuje koruny, cena se ukládá v haléřích
    int price = Integer.valueOf(priceParts[0]) * 100;
    // Druhá část představuje haléře
    if (priceParts.length > 1) {
        switch (priceParts[1].length()) {
            case 0:
                // Pokud bylo zadáno číslo končící desetinnou čárkou, použije
se 0 haléřů
                priceParts[1] = "0";
                break;
            case 1:
                // Pokud bylo zadáno jen jedno desetinné místo, přidá se na
konec 0 aby se číslo vyjádřilo v haléřích
                priceParts[1] += "0";
                break;
            case 2:
                // Pokud byla zadána dvě desetinná místa, není potřeba nic
upravovat
                break;
            default:
                // Pokud bylo zadáno více desetinných míst, třetí a další se
ignorují
                priceParts[1] = priceParts[1].substring(0, 2);
        }
        // Převedení haléřů na číslo a přičtení k ceně
        price += Integer.valueOf(priceParts[1]);
    }

    // Uložení údajů do instance (využíváno z více konstruktorů, vizte
kompletní zdrojové kódy)
    this.setUp(name, price, vat, color, category);
}

private void setUp(String name, int price, VAT vat, ItemColor color, String
category) {
    // Uložení údajů do instance
    this.name = name;
    this.price = price;
    this.vat = vat;
    this.color = color;
    this.category = category;

    // Sestavení vyhledávacího řetězce, údaje jsou odděleny tabulátorem
    this.searchString = ""
            + this.name.toLowerCase() + "\t"
            + this.category.toLowerCase() + "\t"
            + this.vat.toString().toLowerCase() + "\t"
            + this.getPriceStr().toLowerCase()
    ;
}
```

6 ZÁVĚR

V teoretické části byly popsány důležité technické aspekty. Dále bylo popsáno EET včetně stručného přehledu stavu v zahraničí s důrazem na Chorvatsko, jehož systém byl vzorem pro ten český. Byly také zmíněny důležité dopady EET na ekonomiku, ale nebylo dosaženo jednoznačného závěru, jelikož je to téma na celou diplomovou práci v jiném oboru.

Součástí bakalářské práce bylo také naprogramování knihovny pro usnadnění komunikace s finanční správou. Knihovna nebyla součástí zadání, ale byla vypracování po úvodní konzultaci s vedoucím bakalářské práce. Umožňuje programátorovi komunikovat s finanční správou jednoduchým poskytnutím hlášených údajů knihovně a zavoláním jedné metody. Z hlediska budoucího vývoje se nabízí otázka, jestli by nebylo vhodné vytvořit více robustní knihovnu, která by neřešila pouze komunikaci ale byla by si schopna určité údaje, jako jsou například zaplacené daně z jejich základů, sama vypočítat.

Poslední částí práce byla tvorba aplikace umožňující hlášení tržeb a tisk účtenek. Aplikace byla úspěšně vytvořena a splňuje požadavky zadání. V potaz byly vzaty i připomínky vedoucího a aplikace byla na jejich základě upravena. Byť je aplikace funkční a splňuje základní požadavky na takovou aplikaci, v budoucnu by bylo jistě možné ji vylepšit zejména na základě zkušeností z praktického použití a nebo ji doplnit o pokročilé funkce, které jsou běžně nabízeny placenými pokladními systémy.

7 POUŽITÁ LITERATURA

[1] ČESKO. *Zákon č. 112/2016 o evidenci tržeb* [online]. 16. březen 2016 [vid. 2016-03-16]. Dostupné z: http://www.etrzby.cz/assets/cs/prilohy/sb0043-2016(12).pdf

[2] FINANČNÍ SPRÁVA ČESKÉ REPUBLIKY. Technická specifikace. *etrzby.cz* [online]. 2017 [vid. 2017-03-18]. Dostupné z: http://www.etrzby.cz/

[3] HOSPODÁŘSKÁ KOMORA ČR. Na co si dát pozor aneb jak vybírat pokladnu EET. *EET ANO ale* [online]. 2017 [vid. 2017-04-06]. Dostupné z: https://www.eet-ano-ale.cz/si-dat-pozor-aneb-vybirat-pokladnu-eet/

[4] FINANČNÍ SPRÁVA ČESKÉ REPUBLIKY. *etržby - elektronická evidence tržeb* [online]. 2016 [vid. 2017-04-06]. Dostupné z: http://www.etrzby.cz/cs/Elektronicka-uctenka

[5] WWW.FRESHNET.CZ, Studio Fresh Net, s r o. *MARKEETA - Elektronická pokladna pro vaše podnikání* [online]. 2017 [vid. 2017-03-17]. Dostupné z: https://www.markeeta.cz/

[6] VODAFONE CZECH REPUBLIC A.S. *Elektronická evidence tržeb - Vodafone.cz* [online]. 2017 [vid. 2017-03-17]. Dostupné z: https://www.vodafone.cz/podnikatele/specialni-sluzby/epokladna/

[7] ČSOB. *Elektronická evidence tržeb (EET) | ČSOB* [online]. 2017 [vid. 2017-03-17]. Dostupné z: https://www.csob.cz/portal/podnikatele-firmy-a-instituce/produkty/ucty-a-platebni-styk/platebni-karty-a-akceptace-karet/elektronicka-evidence-trzeb

[8] SUNCRAFT CZ S.R.O. *sun EET pokladna* [online]. B.m.: SUNCRAFT CZ s.r.o., 2017 [vid. 2017-03-17]. Dostupné z: https://play.google.com/store/apps/details?id=cz.suncraft.suneet

[9] A.S, CIGLER SOFTWARE. *Profi Účtenka - EET* [online]. B.m.: CIGLER SOFTWARE, a.s., 2017 [vid. 2017-03-17]. Dostupné z: https://play.google.com/store/apps/details?id=cz.csw.Profiuctenka

[10] WWW.EETPLUS.CZ. *EET Plus jednoduše zdarma* [online]. B.m.: www.eetplus.cz, 2017 [vid. 2017-03-17]. Dostupné z: https://play.google.com/store/apps/details?id=cz.eetplus.eet

[11] L-RA. l-ra/openeet-android. *GitHub* [online]. 2016 [vid. 2017-05-17]. Dostupné z: https://github.com/l-ra/openeet-android

[12] L-RA. l-ra/openeet. *GitHub* [online]. 2017 [vid. 2017-03-17]. Dostupné z: https://github.com/l-ra/openeet

[13] FINANČNÍ SPRÁVA ČESKÉ REPUBLIKY. *Zkušenosti ze zahraničí* [online]. 10. březen 2016 [vid. 2017-05-05]. Dostupné z: http://www.etrzby.cz/cs/zajimavosti_zkusenosti-ze-zahranici

[14] ECONOMIA, A.S. Slovensko zavedlo on-line evidenci tržeb. Využívá ji 12 tisíc firem. *Hospodářské noviny* [online]. 10. duben 2015 [vid. 2017-05-05]. Dostupné z: //byznys.ihned.cz/c1-63835440-slovensko-zavedlo-on-line-evidenci-trzeb-vyuziva-ji-12-tisic-firem

[15] CÍGLER SOFTWARE, A.S. EET v Rakousku: podnikatelé si stěžovali u soudu - Elektronická evidence tržeb od A do Z. *Elektronická evidence tržeb od A do Z* [online]. 22. prosinec 2016 [vid. 2017-05-05]. Dostupné z: http://eet.money.cz/blog/eet-v-rakousku-podnikatele-si-stezovali-u-soudu

[16] CÍGLER SOFTWARE, A.S. Chorvaté vybrali víc daní. Trhovci proti EET protestovali marně - Elektronická evidence tržeb od A do Z. *Elektronická evidence tržeb od A do Z* [online]. 10. říjen 2016 [vid. 2017-05-05]. Dostupné z: http://eet.money.cz/blog/chorvate-vybrali-vic-dani-trhovci-proti-eet-protestovali-marne

[17] TUTORIALSPOINT.COM. What is SOAP? *www.tutorialspoint.com* [online]. 2017 [vid. 2017-03-18]. Dostupné z: https://www.tutorialspoint.com/soap/what_is_soap.htm

[18] GARY SIMS. *I want to develop Android Apps - What languages should I learn?* [online]. 2016 [vid. 2017-03-18]. Dostupné z: http://www.androidauthority.com/want-develop-android-apps-languages-learn-391008/

[19] ORACLE. *What is Java and why do I need it?* [online]. 2017 [vid. 2017-03-18]. Dostupné z: https://www.java.com/en/download/faq/whatis_java.xml

[20] BLUETOOTH SIG, INC. *How it works | Bluetooth Technology Website* [online]. 2017 [vid. 2017-03-18]. Dostupné z: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works

[21] THE APACHE SOFTWARE FOUNDATION. *Apache License, Version 2.0* [online]. 2004 [vid. 2017-04-29]. Dostupné z: https://www.apache.org/licenses/LICENSE-2.0.txt

[22] FREE SOFTWARE FOUNDATION, INC. *gnu.org* [online]. 2017 [vid. 2017-04-29]. Dostupné z: https://www.gnu.org/licenses/license-list.html#apache2

[23] OPEN SOURCE INITIATIVE. *Licenses by Name | Open Source Initiative* [online]. 2017 [vid. 2017-04-29]. Dostupné z: https://opensource.org/licenses/alphabetical

[24] ANDROID OPEN SOURCE PROJECT. *Introduction to Activities | Android Developers* [online]. 2017 [vid. 2017-05-03]. Dostupné z: https://developer.android.com/guide/components/activities/intro-activities.html

[25] ANDROID OPEN SOURCE PROJECT. *Fragments | Android Developers* [online]. 2017 [vid. 2017-05-03]. Dostupné z: https://developer.android.com/guide/components/fragments.html

[26] ANDROID OPEN SOURCE PROJECT. *SharedPreferences* | *Android Developers* [online]. 2017 [vid. 2017-05-03]. Dostupné z: https://developer.android.com/reference/android/content/SharedPreferences.html