Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Mobilní aplikace pro podporu živnostníků

Plzeň, 2019 Ondřej Váně

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 2. května 2019

Ondřej Váně

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé práce Ing. Ladislavu Pešičkovi za cenné rady, připomínky a především za čas, který strávil vedením mé práce.

Abstract

The aim of this work is to create a system consisting of a mobile application for the Android platform and the server. The resulting application is primarily intended for sole traders and small entrepreneurs who could use it to manage income, expenses, traders and stock management. Next function will be evaluation of individual traders and displaying basic statistics in clear graphs. The theoretical part is devoted to the analysis of existing applications, the analysis of libraries that will be used in the resulting application and technologies that are used for the Android platform. In the practical part there is designed the overall system and subsequently described its implementation. In conclusion, the implemented system is tested and suggestions for further expansion are proposed.

Abstrakt

Cílem této práce je vytvoření systému skládajícího se z mobilní aplikace pro platformu Android a serveru. Výsledná aplikace je určena primárně pro živnostníky a malé podnikatele, kteří by ji mohli využít ke správě příjmů, výdajů, obchodníků a skladového hospodářství. Další funkcí bude hodnocení jednotlivých obchodníků a zobrazování základních statistik do přehledných grafů. Teoretická část práce je věnována analýze existujících aplikací, analýze knihoven, které budou ve výsledné aplikaci využity, a technologiím, jež jsou používány pro platformu Android. V praktické části je navrhnut celkový systém a následně popsána jeho implementace. V závěru je implementovaný systém otestován a jsou zde navrhnuty náměty pro další rozšíření.

Obsah

1 Úvod

Cílem této práce je vytvořit mobilní aplikaci na platformě Android, která bude primárně určena pro živnostníky a malé podnikatele, kteří by ji mohli využít k základnímu přehledu o jejich podnikatelské činnosti. V aplikaci budou moci spravovat seznam jednotlivých příjmů, výdajů, obchodníků a skladových položek. Součástí této mobilní aplikace bude serverová část pro přihlášení, registraci a zejména synchronizaci dat jednotlivých uživatelů.

První část této práce je věnována teoretickému rozboru, kde jsou analyzovány některé vybrané aplikace na platformě Android, které by mohly výše popsané funkcionality splňovat. Jsou zde zmíněné jejich základní funkce, výhody a nevýhody. Následně je zde uvedeno shrnutí poznatků z analýzy a popis některých základních funkcí, které by mohla vytvářená aplikace obsahovat. Dále jsou zde analyzovány knihovny, které budou využity k důležitým funkcím aplikace. Na závěr této části jsou rozebrány technologie pro platformu Android, jež jsou v dnešní době nejpoužívanější.

Ve druhé části jsou využity poznatky z teoretického rozboru a jsou zde navrhnuty všechny vhodné funkcionality, které by měl výsledný systém obsahovat. Následně je v této části navržena celková struktura mobilní aplikace a serveru. Dále také obsahuje návrh struktury pro ukládání dat v systému a návrh komunikace mezi serverem a aplikací. Závěr této kapitoly je věnován zabezpečení výsledného systému.

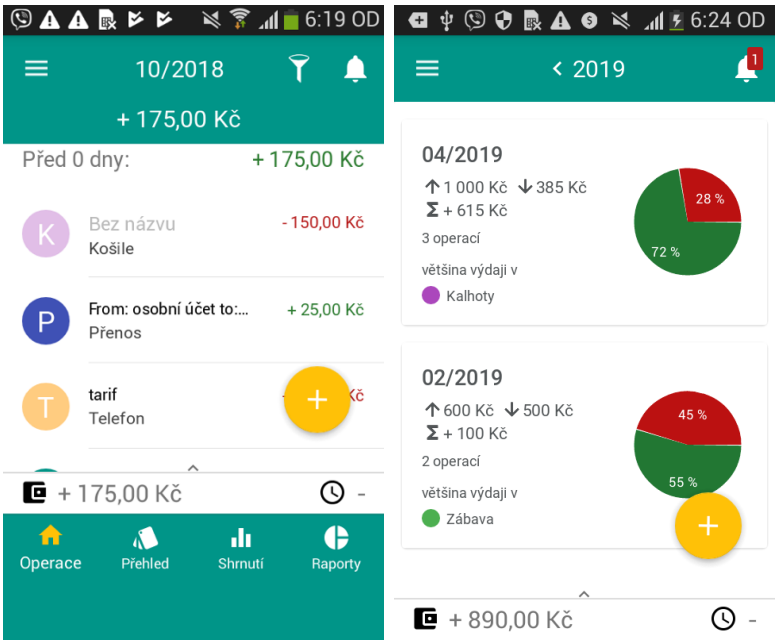
V poslední části je popsána implementace navrhnutého systému, kde je uvedena struktura výsledného projektu a funkce jednotlivých částí. Poté je zde popsán postup testování navrhnutého systému pomocí uživatelských scénářů, které kontrolují správné chování nejdůležitějších funkcí systému. V závěru této části jsou uvedena některá možná rozšíření, která by mohla být námětem pro další práci.

2 Analýza existujících aplikací

V této kapitole budou analyzovány čtyři vybrané aplikace pro platformu Android, které by mohl živnostník využít k podpoře podnikání. Tou je zde myšlena taková aplikace, která usnadní živnostníkovi správu příjmů, výdajů, skladového hospodářství a jednotlivých obchodníků. Aplikace, které budou v této kapitole analyzovány jsou: Moje finance, iDoklad, Smart Receipts a Fakturace, účetnictví a nabídky. Všechny aplikace jsou dostupné ve veřejném repozitáři Google Play. Celkové shrnutí aplikací bude uvedeno v závěru této kapitoly.

2.1 Aplikace Moje finance

Základní funkčnost aplikace Moje finance je členění příjmů a výdajů do různých kategorií viz obrázek 2.1a. Příjmy a výdaje jsou zobrazeny v přehledných grafech za jednotlivé měsíce viz obrázek 2.1b. Základní kategorie jsou v aplikaci přednastavené, lze však přidávat své vlastní. V aplikaci je možné vytvořit několik na sobě nezávislých účtů, mezi kterými lze libovolně přepínat a sledovat jejich bilance.



(a) Hlavní obrazovka (b) Graf příjmů a výdajů

Obrázek 2.1: Aplikace Moje finance

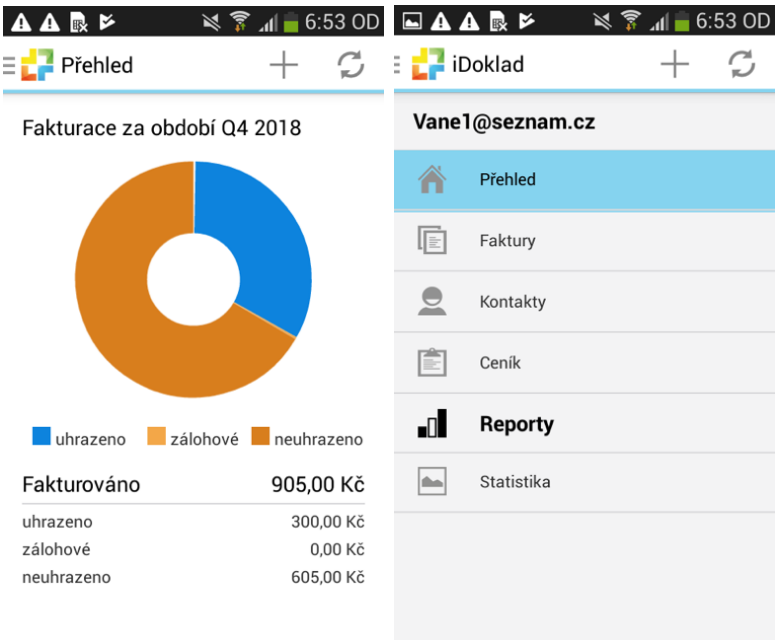
Mezi jednotlivými účty lze snadno převádět peněžní prostředky. Další významná funkce, k ulehčení práce se správou příjmů a výdajů, je existence předpřipravených šablon, které si může uživatel vytvořit a využít je pro pravidelně se opakující příjmy nebo výdaje. Aplikace je opatřena synchronizační funkcí umožňující zálohu dat na server.

Aplikace Moje finance je jednoduchá, přehledná a snadno ovladatelná. Uživatelské rozhraní je přívětivé.

Mezi nedostatky aplikace by bylo možné zařadit nevyužití potenciálu mobilního zařízení např. pořizování fotografií faktur a účtenek. Další nevýhodou je absence skladového hospodářství, které by mohlo být vhodné pro živnostníka. Tato aplikace je vhodná spíše pro správu domácích financí.

2.2 Aplikace iDoklad

Aplikace iDoklad je vhodná pro jednoduché vytváření faktur, správu fakturačních údajů a sortimentu. Jednotlivé faktury jsou rozlišovány podle typu platby (uhrazeno, zálohové a neuhrazeno) viz obrázek 2.2a. Na obrázku 2.2b je zobrazeno menu s hlavními funkcemi. Ve správě fakturačních údajů lze přidávat a evidovat fakturační údaje jednotlivých obchodníků.



(a) Typy faktur (b) Menu

Obrázek 2.2: Aplikace Moje finance

Další funkce je ceník, kde je možno spravovat sortiment zboží nebo služby, které podnikatel nabízí. U každého sortimentu je uveden údaj o ceně, jednotkách a další informace. Následně je možno položky vkládat na fakturu. Přehled vydaných faktur je zobrazen v přehledném koláčovém grafu za jednotlivé kvartály účetního období.

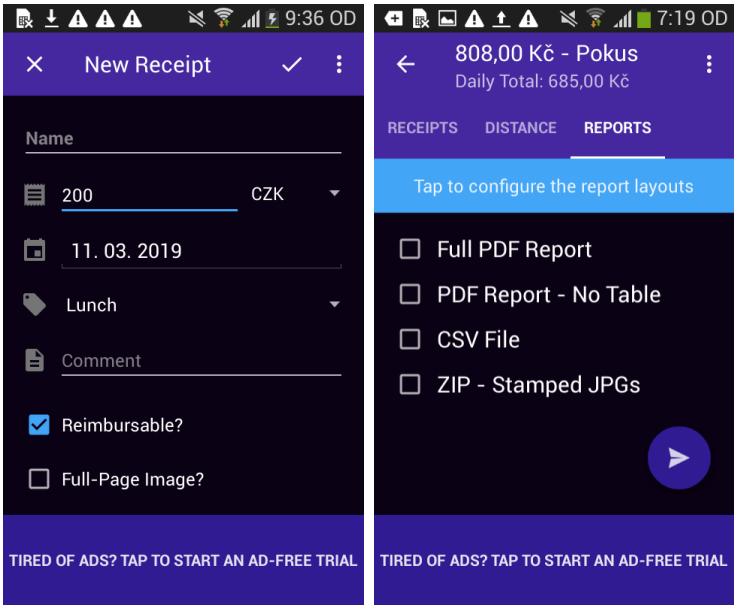
Tato mobilní aplikace je součástí webového portálu iDoklad.cz nabízející mnoho dalších nástrojů pro podnikatele. Tento webový portál je plnohodnotným účetním systémem v cloudu.

Mezi přednosti aplikace iDoklad patří snadná tvorba faktur, evidence nabízeného sortimentu a přehledné zobrazení částek za vydané faktury.

Mezi nevýhody patří absence jakýchkoliv výdajů, tudíž si uživatel nemůže zobrazit statistiky o jeho výdělcích. Další nedostatek aplikace je chybějící množství u sortimentu zboží. Aplikace iDoklad je vhodná pro správu a tvorbu faktur.

2.3 Aplikace Smart Receipts

Smart Receipts je aplikace založená na uschovávání stvrzenek, účtenek, faktur ve formě fotografií. Ke každé stvrzence je možno přidat částku, datum uskutečnění výdaje a libovolnou poznámku viz obrázek 2.3a. Dále lze generovat přehled do různých formátů například PDF, CSV nebo ZIP viz obrázek 2.3b.



(a) Vytváření výdaje (b) Generování přehledu

Obrázek 2.3: Aplikace Smart Receipts

Výdaje lze členit do různých období, které si uživatel nastaví (týden, měsíc, rok). Další funkce je přidávání výdajů za cesty, které uživatel absolvoval. K tomuto výdaji lze přidat ujetou vzdálenost, cenu za kilometr a cíl cesty. Aplikace umožňuje vedení výdajů v několika národních měnách.

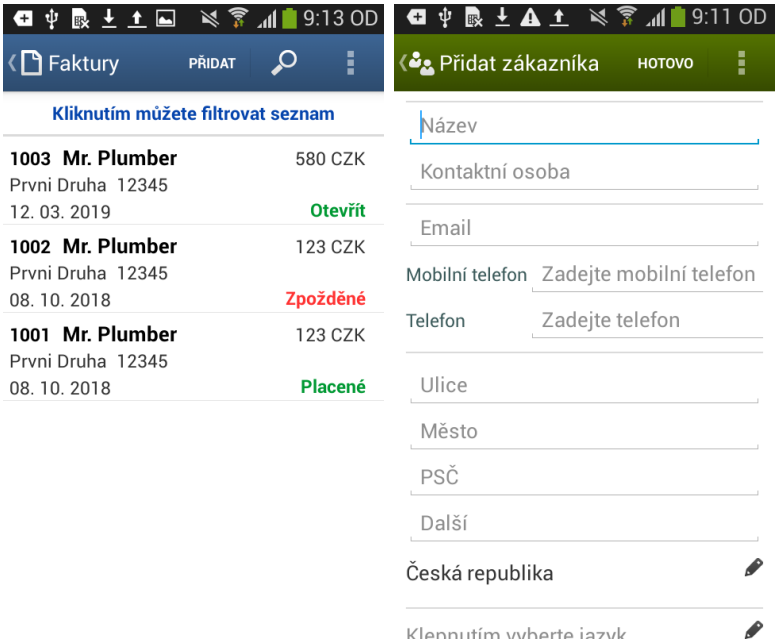
Mezi největší přednosti této aplikace patří využití fotoaparátu k uschování jednotlivých účtenek a faktur, které jsou snadno dohledatelné. Další užitečná vlastnost je generování dat z aplikace do různých formátů, jež mohou být využity v jiných systémech pro správu výdajů.

Hlavní nevýhoda je absence přehledného grafu, který by uživatele informoval o výdajích za určitá období.

2.4 Aplikace Fakturace, účetnictví a nabídky

Aplikace Fakturace, účetnictví a nabídky pochází od společnosti Speed Invoice, která se zabývá vývojem účetního softwaru pro mobilní zařízení, tablety a osobní počítače.

Základní dovedností aplikace je vytváření nabídek a faktur. Ke každé faktuře lze přiřadit datum fakturace a splatnosti. V hlavním náhledu je následně zobrazen stav faktury(placené, otevřené, zpožděné) viz obrázek 2.4a.



(a) Hlavní přehled faktur (b) Vytváření zákazníka

Obrázek 2.4: Aplikace Faktura, účetnictví a nabídky

Fakturační údaje jsou vybírány z předem vyplněných zákazníků, jež si může uživatel přidávat. Na obrázku 2.4b je zobrazeno přidání nového zákazníka. Dále je možno vkládat jednotlivé položky na fakturu, které si uživatel vytvoří.

Funkce nabídky je v podstatě vytvoření nové faktury, u které nebude nastavené datum splatnosti. Pokud se služba nebo prodej zboží uskuteční, uživatel má možnost nabídku převést mezi faktury a doplnit datum splatnosti.

Tato aplikace má velice podobné funkce jako iDoklad, který byl zmíněn výše. Aplikace bohužel postrádá přehledné zobrazení vydaných faktur například pomocí grafu. Oproti iDokladu však obsahuje funkci nabídky, která může být pro podnikatele velice vhodná.

2.5 Shrnutí

První velkou část dostupných aplikací, které by mohl živnostník využít, jsou aplikace pro správu osobních financí. Z této oblasti byla vybrána aplikace pod názvem Moje finance popsána v kapitole č. 2.1. Další zástupci jsou například Domácí finance jednoduše, Wallet, Domácí rozpočet a 1Money. Všechny tyto aplikace umožňují základní správu příjmů a výdajů členěných do různých tříd. Následně zobrazí všechny příjmy a výdaje v grafu a tím si mohou uživatelé udělat představu o jejich hospodaření s finančními prostředky. Tyto aplikace jsou založeny na tom, že budou sloužit zejména pro správu osobních a domácích financí. Proto tyto aplikace postrádají skladové hospodářství a vedení seznamu obchodníků.

Druhou část tvoří aplikace pro vytváření a správu faktur. Zástupci těchto aplikací jsou popsány v kapitolách č. 2.2 a č. 2.4. Umožňují vedení seznamu obchodníků, kterým může být vystavena faktura. Další funkce je vedení sortimentu zboží, které je možno přidávat na fakturu. Tyto aplikace jsou určeny výhradně k tvorbě faktur, tudíž neobsahují jakoukoli správu výdajů.

Po analýze vyplynulo, že by bylo vhodné vytvořit aplikaci, která bude spravovat příjmy a výdaje, podobně jako v aplikacích pro osobní finance. Možnost vedení seznamu obchodníků a hodnocení k nim a správu skladových položek s aktuálním dostupným množstvím. Vše by mělo být zobrazené v přehledných grafech.

3 Použité knihovny

V této kapitole budou analyzovány knihovny, které bude aplikace využívat pro své hlavní funkce. Mezi tyto funkce patří, zobrazování statistik do přehledných grafů a komunikace s webovým serverem (registrace, přihlašování a synchronizace dat).

3.1 Analýza grafových knihoven

Ve výsledné aplikaci budou zapotřebí grafy, které zobrazí statistiku příjmů, výdajů a hodnocení obchodníků. Příjmy a výdaje budou zobrazovány v koláčovém a liniovém grafu za vybrané období. Hodnocení obchodníků bude ve sloupcovém grafu.

Nezbytné vlastnosti, které by měla knihovna pro vykreslování grafů splňovat, jsou:

- vykreslování koláčových, liniových a sloupcových grafů,
- animaci jednotlivých grafů,
- jednoduchou implementaci,
- uživatelsky přívětivé zobrazování.

Pro analýzu byly vybrány následující knihovny: GraphView, MPAndroid chart a EazeGraph.

3.1.1 Knihovna GraphView

GraphView je open source knihovna pro vykreslování grafů na platformě Android. V této knihovně jsou dostupné liniové, sloupcové a bodové grafy. Je možné vykreslení všech možných kombinací těchto grafů. Umožňuje snadné vytváření datových řad, které jsou zobrazené v grafu. Dále je možné snadno měnit barvu, styl a rozsah zobrazovaných dat. Knihovna je schopna přidat animace do jednotlivých grafů.

Tato knihovna je dostupná s licencí MIT (tyto licenční podmínky bývají někdy označovány spíše jako X11 License). MIT licence umožňuje komukoliv bez omezení používat a šířit program, pokud z něj neodstraní kopii licence a jméno autora [16].

3.1.2 Knihovna MPAndroid Chart

MPAdroid chart je open source knihovna dostupná zdarma. Oproti předchozí knihovně GraphView obsahuje větší množství grafů jako například koláčový, svícnový, radarový a bublinový. U všech grafů je možné nastavit animaci s různou délkou trvání efektu. Implementace dat do grafů je velice snadná a obsahuje množství funkcionalit, které zpřehledňují výsledný graf. Pro barevné schéma grafu lze využít předdefinovaných barevných palet. Pokud je schéma nevyhovující lze nadefinovat své vlastní. Většina dostupných grafů je interaktivní. Umožňují přibližování, oddalování a posouvání.

Knihovna je dostupná s licencí Apache License 2.0. Tato licence vyžaduje po uživateli zachování autorství a tzv. disclaimer. Disclaimer obecně znamená zřeknutí odpovědnosti. Apache umožňuje uživateli svobodně software využívat, distribuovat a upravovat v souladu s licenčními podmínkami [3].

3.1.3 Knihovna EazeGraph

EazeGraph je open source knihovna, která nabízí čtyři základní grafy (sloupcový, sloupcový vrstvený, koláčový a liniový) a u všech lze nastavit animaci, která je spuštěna při zobrazení grafu. Interakce na dotyk je dostupná pouze pro koláčový a liniový graf. Zobrazení dat do grafu je velice jednoduché a intuitivní.

Knihovna EazeGraph je dostupná s licencí Apache License 2.0, jejíž vlastnosti byly popsány v kapitole 3.1.2.

3.1.4 Shrnutí

První knihovna postrádá koláčový graf, který je nezbytný pro přehledné zobrazení dat ve výsledné aplikaci. Proto není knihovna GraphView vhodná pro toto využití. Druhá knihovna obsahuje všechny potřebné grafy a umožňuje velikou variabilitu a úpravu grafů podle využití. Vizuálně působí velice přívětivě. Poslední analyzovaná knihovna dominuje nejjednodušší implementací dat do grafu, ale neumožňuje tolik variability jako předchozí knihovna.

Na základě požadavků byla vybrána jako nejvhodnější knihovna MPAndroid chart, která splňuje všechny potřebné vlastnosti.

3.2 Analýza knihoven pro komunikaci se serverem

Další funkcí vytvářené aplikace bude komunikace s webovým serverem. Komunikace se serverem bude využita k registraci a přihlášení uživatele. Dále také k záloze a obnově uživatelských dat.

Bude zapotřebí knihovna se schopností zasílat a přijímat zprávy pomocí metody POST ve formátu JSON. Jedná se o jeden z nejpoužívanějších formátů pro přenos dat. Pro analýzu byly vybrány dvě knihovny: Volley a Retrofit.

3.2.1 Knihovna Volley

Volley je síťová knihovna vyvinutá společností Google, která byla vydána v roce 2013. Jedná se o velice užitečnou knihovnu nabízející mnoho funkcí jako synchronní a asynchronní požadavky, stanovení priorit, vytváření více požadavků najednou a také ukládání dat do mezipaměti. K největším nedostatkům Volley patří nedostatečně podrobná dokumentace k jednotlivým funkcím [2].

Pomocí této knihovny je možno zasílat a přijímat data ve třech různých formátech (text, JSON a pole JSON). Pokud je poslán požadavek s jedním vybraným formátem, může být přijat pouze ve stejném formátu (jestliže jsou odeslána data ve formátu JSON, tak mohou být přijaty pouze ve formátu JSON). Tento nedostatek lze odstranit přepsáním několika metod v knihovně. Další funkcí je stahování obrázků z URL adresy. Ten je v aplikaci automaticky převeden na bitmapu, se kterou lze dále pracovat. Své požadavky je schopna knihovna vykonávat na pozadí, což je v aplikacích Android vyžadováno.

Jedna z největších předností Volley je opakování požadavku při vypršení časového limitu. Při vytváření požadavku lze nastavit metodu s názvem *setRetryPolicy*, která opakuje zasílání podle nastavení. Ve výchozím nastavení je doba opakování stanovena na 5 vteřin, tu však lze snadno změnit. Pokud dojde k opakovanému zasílání požadavku, tak je zpráva načtena z mezipaměti, kde je dočasně uložena [2].

Knihovna Volley je zdarma, open source a dostupná s licencí Apache License 2.0.

3.2.2 Knihovna Retrofit

Retrofit je REST klient pro Android, který vytvořila společnost Square. Díky této knihovně lze relativně snadno číst, nahrávat, editovat a mazat data ze serveru pomocí webové služby.

Tato knihovna dokáže zpracovávat data ve stejných formátech jako Volley. Navíc však dokáže pracovat s několika dalšími formáty jako jsou například datum, číslo, objekt a kolekce [2]. Pokud je nutná komunikace pomocí formátu JSON, tak je nezbytné využití další knihovny, která konvertuje přijatá data do objektu JSON. Není tedy automaticky přeložen do požadovaného formátu.

Retrofit umožňuje také opakované odesílání požadavků po daném časovém intervalu. Bohužel však neobsahuje funkci ukládající vytvořené zprávy do mezipaměti, proto může být ve výsledku o něco pomalejší. Tuto funkci je možno svépomocí doplnit.

Knihovna Retrofit je zdarma, open source a dostupná s licencí Apache License 2.0.

3.2.3 Shrnutí

Obě analyzované knihovny splňují všechny vybrané vlastnosti, které byly výše definovány.

Retrofit je velice jednoduchá knihovna, avšak oproti Volley neobsahuje takové množství nastavení a úprav dle požadavků programátora. Výhoda knihovny Volley je automatická transformace požadavků do potřebného formátu a nemusí být k tomu využita další knihovna. Volley dále obsahuje ukládání vytvořených požadavků do mezipaměti, což Retrofit v základní verzi neobsahuje. Tím může být Volley v některých případech rychlejší.

Pro komunikaci v rámci vytvářené aplikace byla vybrána Knihovna Volley.

4 Technologie pro Android

Android je nejrozšířenějším operačním systémem pro mobilní zařízení. Ve třetím kvartálu roku 2018 měl Android 86,8% podíl na trhu chytrých mobilních telefonů následovaný operačním systémem iOS s 13,2% [18].

Programovacích jazyků pro operační systém Android je mnoho, výčet základních je uveden v následujícím seznamu. Avšak mezi nejpoužívanější patří Java a Kotlin:

- *Java* – Je oficiálním jazykem pro vývoj Android aplikací s plnou podporou IDE Android Studio.
- *Kotlin* – Společnost Google začíná vývoj v Kotlinu prosazovat a mnoho vývojářů na něj přechází. Umožňuje kombinovat třídy napsaná v Javě a Kotlinu.
- *C#* – Podporuje některé užitečné nástroje jako je například Unity nebo Xamarin, které jsou dobré pro vývoj her nebo multiplatformních aplikací.
- *C / C++* – Android Studio podporuje C++ s využitím Java NDK, které umožňuje nativní kódování. Tyto programovací jazyky mohou být využity pro tvorbu mobilních her [14].

4.1 Java

Java je objektově orientovaný programovací jazyk, který v roce 1995 vyvinula firma Sun Microsystems (od roku 2010 součástí společnosti Oracle). Podle TIOBE indexu z března roku 2019 je momentálně nejpopulárnějším programovacím jazykem na světě. TIOBE index je indikátorem popularity programovacích jazyků [19].

Java je interpretovaným programovacím jazykem, který je překládán do tzv. bajtkódu, který je nezávislý na architektuře počítače nebo ostatních zařízeních. Program je spouštěn pomocí interpreta JVM, jenž se skládá z části zajišťující vazbu na hardware a z části interpretující bajtkód. Java obsahuje generační správu paměti, jenž je realizována pomocí garbage collectoru, který vyhledává již nepoužívané části paměti a uvolní je pro další použití. Java využívá silnou typovou kontrolu vyžadující definování datových typů všech používaných proměnných [7].

Android však nevyužívá k běhu svých aplikací, napsaných v Javě, JVM. Využívá virtuální stroj Dalvik, který byl vytvořen společností Google. Aplikace je kompilována do bajtkódu s příponou `.dex` a `.odex` pro běh na virtuálním stroji Dalvik. Koncovka `.dex` je *Dalvik Executable*, jenž je kompaktním formátem pro systémy, které jsou omezeny paměťovou nebo výkonovou kapacitou. Soubor s koncovkou `.odex` znamená *Optimized Dalvik Executable* a obsahuje optimalizovanou formu souboru `.dex` [10].

Od verze Android 5.0 „Lollipop“ však plně nahradilo Dalvik nové běhové prostředí pro aplikace s názvem ART. Hlavní výhodou je, že ART překládá aplikace z bajtkódu na strojové instrukce přímo při instalaci (Dalvik tento krok realizoval až při spuštění aplikace). Výhodou je menší náročnost při spouštění aplikací, větší výdrž baterie a celkově větší plynulost prostředí.

4.2 Kotlin

Kotlin byl vydán v roce 2011 společností JetBrains. Jde o objektově orientovaný jazyk běžící nad JVM. Kotlin byl navržen tak, aby nahrazoval nedostatky jazyka Java a byl s ní zároveň interoperabilní.

Kotlin je staticky typovaný jazyk, všechny proměnné mají pevně daný datový typ. Avšak není vždy nutné datový typ explicitně udávat, kompilátor odpovídající typ doplní.

Kotlin obsahuje oproti Javě jednodušší lambda výrazy, automatickou implementaci getterů a setterů, primární konstruktory a mnoho dalších funkcionalit [11]. Aplikace jsou opět kompilovány pro běh na virtuálním stroji Dalvik jako v Javě, tudíž je možno část aplikace vyvíjet v Kotlinu a část v Javě. Je běžné, že aplikace je psána v Javě a jsou využity knihovny, které jsou napsané v Kotlinu nebo naopak. Dále je možno v Kotlinu vyvíjet multiplatformní aplikace pro Android a iOS.

4.3 Shrnutí

Momentálně je Java stále primárním oficiálním jazykem pro vývoj aplikací na platformě Android, ale spoustu vývojářů přechází na Kotlin, který eliminuje některé nedostatky Javy. Jelikož Kotlin vychází z Javy, tak přechod mezi nimi je velice snadný a rychlý. Dokumentace pro oba jazyky jsou rozsáhlé a přehledné, avšak pro Javu existuje větší množství materiálů, jako jsou například návody, řešené problémy a postřehy ostatních vývojářů.

Pro vývoj aplikace byl vybrán programovací jazyk Java, se kterým má autor více zkušeností a také z důvodu většího množství materiálů.

5 Návrh systému

V této kapitole budou popsány funkcionality vytvářené aplikace pro podporu živnostníků. V další části bude popsán návrh aplikace a její základní struktura. Dále zde bude představena funkce serveru a jeho návrh.

Název vytvářené aplikace bude Živnostníček. Tento název byl zvolen z důvodu, aby na první pohled bylo jasné, že se jedná o aplikaci pro živnostníky a malé podnikatele. Tato cílová skupina by mohla využít aplikaci k přehledu o jejich podnikatelské činnosti, profitu, dostupném materiálu a jednotlivých obchodnících.

5.1 Vybrané funkcionality aplikace

Aplikace bude obsahovat následující vybrané funkcionality, které byly vybrány mimo jiné na základě analýzy existujících aplikací. Některé již byly zmíněny v podkapitole 2.5.

5.1.1 Zobrazení statistik do grafů

Jedna ze základních funkcionalit aplikace bude zobrazování jednotlivých statistik do přehledných grafů. Budou zde použity tři základní typy grafů: koláčový, liniový a sloupcový. Data zobrazená v grafech budou závislá na vybraném období, které si bude moci uživatel zvolit, aby bylo možné odlišit jednotlivá účetní období či měsíce. V aplikaci budou zobrazeny statistiky příjmů a výdajů (kumulativní nebo v čase), průměrné hodnocení jednotlivých obchodníků, příjmy a výdaje členěné podle vytvořených druhů.

5.1.2 Vedení obchodníků

Tato funkcionalita bude schopna přidat obchodníka a další informace k němu jako například kontaktní osobu, telefonní číslo, IČO, DIČ a sídlo obchodníka. Sídlo si bude moci uživatel zobrazit na mapě. Dále bude možnost všechny informace u obchodníka editovat či smazat celý záznam.

5.1.3 Hodnocení obchodníků

Tato funkcionalita je určena pro hodnocení služeb, komunikace nebo kvality zboží daného obchodníka. Ke každému obchodníkovi bude moci uživatel při-

dávat hodnocení, které bude obsahovat počet hvězdiček (od 0 do 5 hvězdiček) a také vlastní poznámku. U náhledu obchodníka bude zobrazena průměrná hodnota vypočtená ze všech dostupných hodnocení.

5.1.4 Základní skladové hospodářství

Další funkcionalitou je vedení skladového hospodářství živnostníka. Bude možnost vytvořit skladovou položku s určitým názvem, množstvím, jednotkou a poznámkou. V náhledu skladu bude zobrazen název skladové položky a její dostupné množství ve skladu. Jednotlivé skladové položky bude možno přidávat na fakturu, a tím se bude měnit její dostupné množství.

5.1.5 Vedení příjmů a výdajů

Základní funkcionalitou bude vedení příjmů a výdajů, dále označovány jen jako faktury, které budou zobrazovány v přehledných grafech. Ke každé faktuře bude uživateli umožněno přidat základní informace jako je název, částka, datum a sazba DPH. Dále bude možno přiřadit fotografii faktury, kterou bude moci uživatel pořídit nebo vybrat již existující fotografii z galerie. Tato funkcionalita také bude umožňovat přiřadit k faktuře existujícího obchodníka a zařadit fakturu podle typu příjmu nebo výdaje.

Ke každé faktuře také bude moci uživatel přidat jednotlivé položky faktury a jejich množství. Pokud se bude jednat o příjem, tak bude moci přidávat pouze položky, které jsou dostupné v zadaném množství ve skladu. Jestliže půjde o výdaj, bude moci uživatel vytvořit novou skladovou položku, která bude následně vytvořena s odpovídajícím množstvím ve skladu.

5.1.6 Základní informace pro živnostníky

V této funkcionalitě budou zobrazeny základní informace, které by mohl živnostník využít. Mezi tyto informace patří aktuální měnové kurzy základních měn, důležité termíny a užitečné odkazy na webové servery pro živnostníky a podnikatele. Pro načítání aktuálních kurzů bude využit server ČNB. Data o důležitých termínech budou uložena na serveru, odkud se prostřednictvím aplikace stáhnou.

5.1.7 Nastavení

V nastavení si bude moci uživatel vybrat pro jaké období budou data v aplikaci zobrazována. Tím je umožněno, aby mohl uživatel pracovat pouze s daty za vybrané účetní období nebo za určitý měsíc v roce. Tímto nastavením

budou ovlivněny pouze příjmy a výdaje. Další volbou v nastavení bude umožnění vkládat zahraniční IČO a DIČ, jež mohou mít rozdílný formát od českých identifikačních údajů, které jsou validovány.

5.2 Funkce serveru

Součástí vytvářené aplikace bude také server, který bude využit k registraci, přihlášení a záloze dat uživatele. Jednotlivé funkce serveru jsou podrobněji popsány níže.

5.2.1 Registrace uživatele

Jedna z funkcí serveru je registrace nového uživatele. Uživatel, jenž bude chtít využívat aplikaci Živnostníček se musí nejprve zaregistrovat. Registrace bude probíhat přímo v jedné z aktivit aplikace. Údaje, které budou vyžadovány po uživateli k registraci jsou: jméno, příjmení, e-mail a heslo. Minimální vyžadovaná délka hesla bude 8 znaků, což je v dnešní době považováno za osvědčenou minimální délku hesla [15]. Server bude kontrolovat, zda již vytvářený uživatel neexistuje podle zadané e-mailové adresy. Následně oznámí uživateli, zda registrace proběhla úspěšně či nikoli. Pokud došlo k nějaké chybě, oznámí tuto chybu uživateli.

5.2.2 Přihlášení uživatele

Další funkcí serveru je autentizace uživatele, který je již zaregistrovaný. Autentizace uživatele bude probíhat pomocí uživatelského jména a hesla. Jako uživatelské jméno bude využita e-mailová adresa, která se snáze pamatuje. Uživatel si bude moci zvolit, zda zůstane přihlášen nebo po každém spuštění aplikace bude znovu zadávat své přihlašovací údaje. Jestliže autentizace uživatele proběhne v pořádku, bude uživatel přesměrován do hlavní aktivity aplikace.

5.2.3 Synchronizace dat

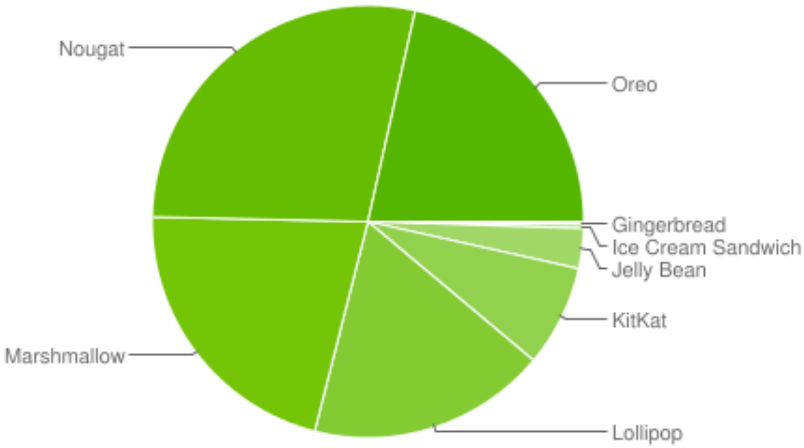
Poslední funkcí serveru je synchronizace dat se serverem. Tím je zde myšlena záloha dat na server a obnova dat ze serveru. Primárně je tato funkce určena k ochraně aplikačních dat při ztrátě nebo poškození zařízení. Zálohu dat bude moci uživatel provádět ručně a nebo nastavit automatickou zálohu dat při každé změně záznamu. Dále zde bude umožněno nastavit zálohování dat pouze s připojením k Wi-Fi, aby se uživatel nemusel obávat o vyčerpání

datového limitu mobilních dat. Při ztrátě nebo poškození zařízení bude data možno obnovit ze serveru.

5.3 Minimální úroveň API aplikace

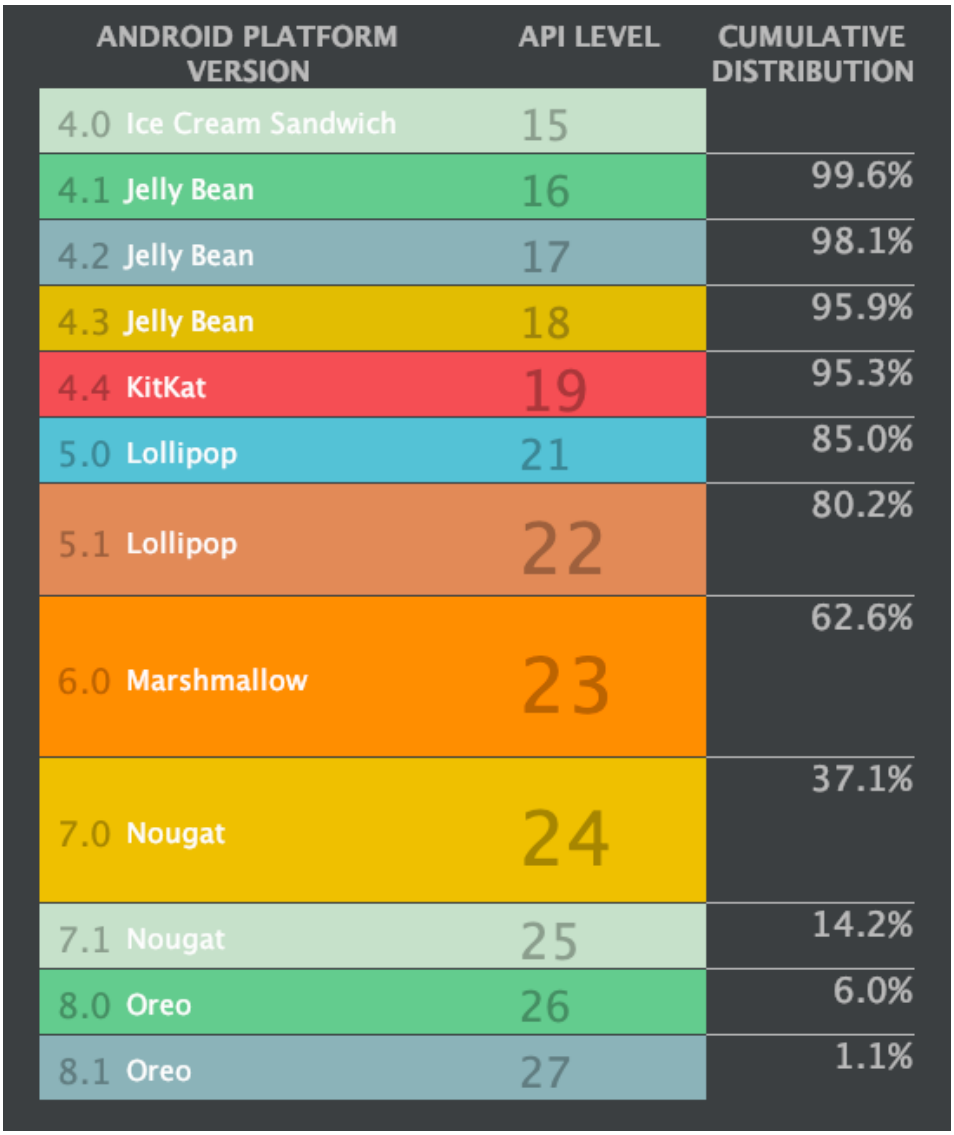
Před začátkem vývoje aplikace je nutné zvolit minimální úroveň API tzv. *API level*, jenž bude aplikací podporován. To znamená, že tuto aplikaci bude možno spustit pouze na zařízení s minimální verzí API nebo vyšší. Každá úroveň API se váže k jedné verzi Androidu. Minimální úroveň je dobré vybírat v závislosti na uživatelské základně tak, aby aplikaci mohlo využívat co největší spektrum uživatelů. Dalším důležitým faktorem pro výběr minimální verze API jsou také podporované funkcionality. Například v API úrovně 23 se poprvé objevila autentizace pomocí otisku prstů. Tato funkce nebyla v předchozích verzích podporována [1].

Jelikož vytvářená aplikace nebude využívat žádné speciální nebo sofistikované funkce, které jsou dostupné ve vyšších úrovních API, bude nejdůležitějším faktorem výběru uživatelská základna. Pro výběr minimální API úrovně lze využít aktuální statistiky od společnosti Google, které obsahují zastoupení jednotlivých verzí Androidu na trhu. Tato statistika je zobrazena na obrázku 5.1, kde tmavší odstín odpovídá novější verzi.



Obrázek 5.1: Distribuce verze Androidu pro říjen 2018 [6]

Dále je zde také možné využít statistiku, která je dostupná ve vývojovém prostředí Android Studia, která je zobrazena na obrázku 5.2. Tato statistika zobrazuje, kolik procent uživatelů bude schopno tuto aplikaci používat při výběru konkrétní úrovně API.



Obrázek 5.2: Statistika pro výběr úrovně API v Android Studiu

Pro vytvářenou aplikaci byla zvolena minimální úroveň API 16. Podle statistik z výše uvedeného obrázku může aplikaci s touto verzí API využívat 99,6% uživatelů.

5.4 Oprávnění aplikace

Základním účelem oprávnění je chránit soukromí uživatele Androidu. V závislosti na funkci může systém přidělit oprávnění automaticky nebo se musí tázat uživatele, který jej musí schválit. Jedná se například o přidělení práva zápisu do externí paměti systému. To by mohlo ovlivnit bezpečnost sytému

nebo ostatních aplikací. Z tohoto důvodu oprávnění vyžaduje schválení od uživatele [17]. Oprávnění lze přidělit v nastavení aplikace nebo pomocí dialogových oken, které se táží uživatele, zda chce oprávnění povolit.

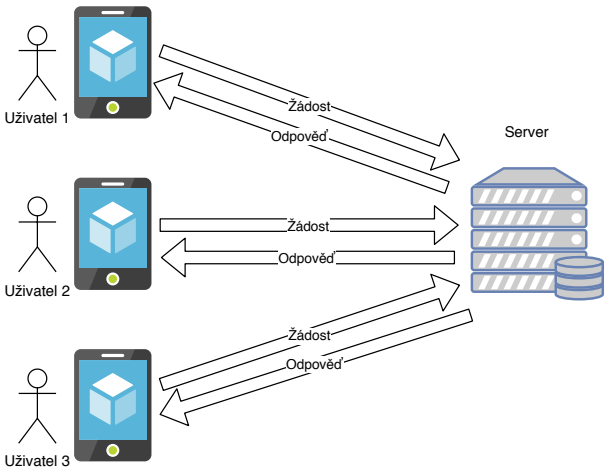
Jednotlivá oprávnění aplikace jsou obsaženy v souboru *AndroidManifest.xml*. V následujícím seznamu jsou uvedena všechna oprávnění, které aplikace vyžaduje:

- *android.permission.INTERNET* - Oprávnění, které povolí aplikaci otevřít síťový soket [17]. Toto oprávnění je využito pro komunikaci aplikace se serverem.
- *android.permission.ACCESS_NETWORK_STATE* - Oprávnění, které umožní aplikaci přístup k informacím o síťovém připojení [17]. Toto oprávnění je opět využito ke komunikaci se serverem.
- *android.permission.CAMERA* - Oprávnění, které je vyžadováno při práci s fotoaparátem a pořizování snímků. Práce s fotoaparátem je vyžadována při snímání obrázku faktur. Toto oprávnění není přiděleno automaticky, tudíž je musí uživatel potvrdit.
- *android.permission.READ_EXTERNAL_STORAGE* - Oprávnění, které povolí aplikaci přistupovat a číst data z externího úložiště mobilního zařízení. Toto oprávnění je vyžadováno ke čtení pořízených fotografií a také ke čtení pomocných souborů.
- *android.permission.WRITE_EXTERNAL_STORAGE* - Oprávnění, které povolí aplikaci zapisovat data do externího úložiště mobilního zařízení. Toto oprávnění je vyžadováno k zápisu pořízených fotografií a také k zápisu pomocných souborů.
- *android.permission.WRITE_CALENDAR* - Oprávnění, které umožňuje aplikaci zapisovat data do kalendáře uživatele. Toto oprávnění je vyžadováno ve funkci základních informací pro živnostníky, kde jsou zobrazeny důležité termíny. Tyto termíny si mohou uživatelé vložit do svého kalendáře.

Oprávnění, která jsou označena jako nebezpečná, musejí být explicitně povoleny uživatelem: přístup k fotoaparátu, čtení a zápis do externího úložiště a zápis do kalendáře. Pro přidělení těchto oprávnění je využita knihovna s názvem *EasyPermissions*, jež zobrazuje uživateli dialogová okna pro potvrzení jednotlivých oprávnění. Toto potvrzení probíhá při startu aktivity, která vyžaduje příslušná oprávnění.

5.5 Celková struktura systému

Celkovou strukturu systému lze rozdělit do dvou základních částí: mobilní aplikace a server. Mobilní aplikace komunikuje se serverem pomocí žádosti a příslušné odpovědi přes internet. Před každou žádostí se pokusí aplikace navázat spojení se serverem. Pokud je spojení navázáno úspěšně, je odeslána zpráva, kterou server zpracuje a odešle zpět příslušnou odpověď. Jestliže je uživatel v aplikaci trvale přihlášen, může využívat většinu funkcí aplikace v offline režimu. Základní model celkové struktury je zobrazen na obrázku 5.3.



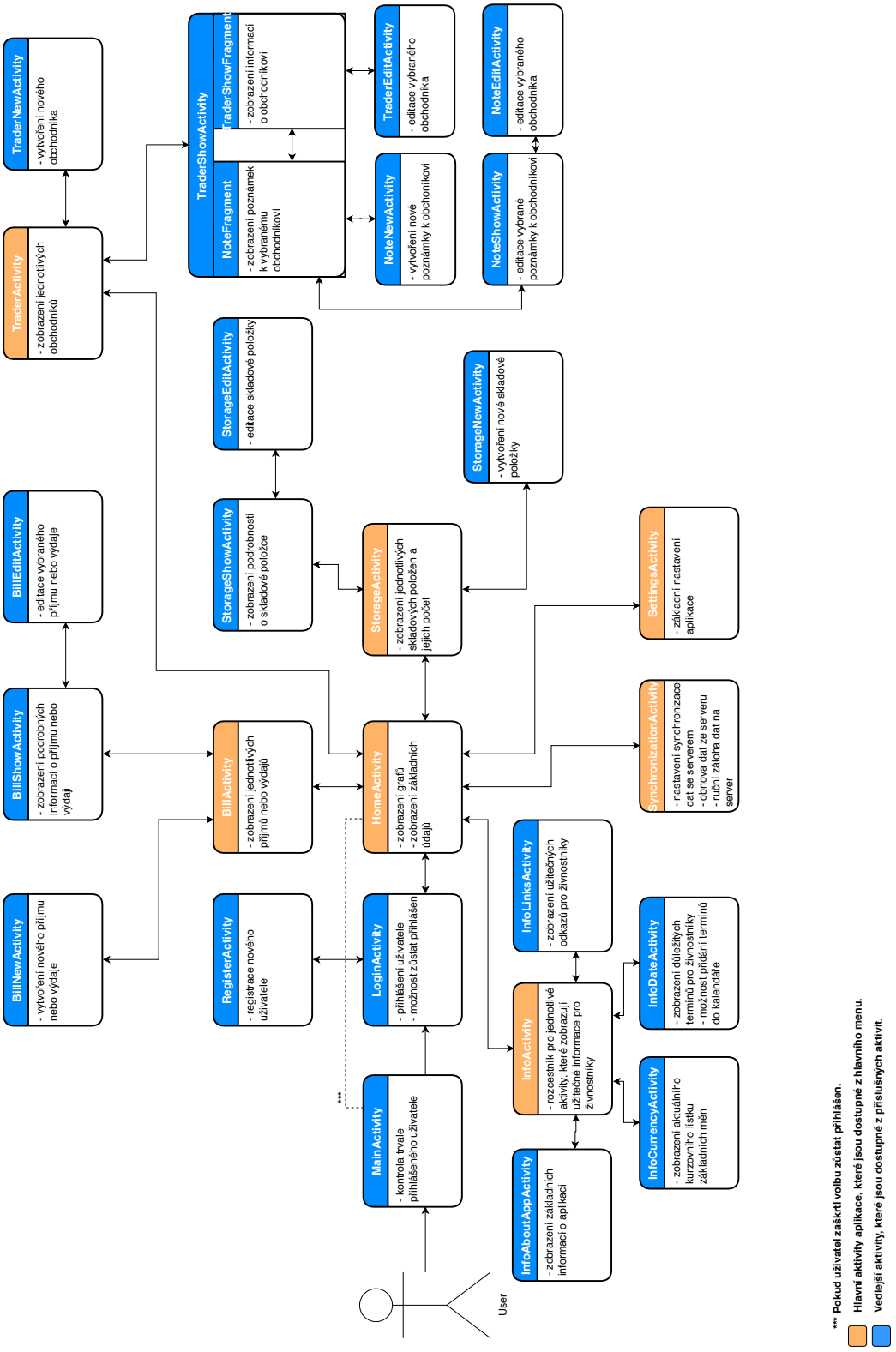
Obrázek 5.3: Celková struktura projektu

5.5.1 Struktura aplikace

Základním stavebním kamenem aplikace jsou tzv. aktivity a fragmenty, které obstarávají jednotlivé funkce. Celá aplikace obsahuje 32 aktivit a 2 fragmenty.

Při každém spuštění aplikace je zavolána aktivita s názvem `MainActivity`, která zjistí, zda je uživatel trvale přihlášen. Pokud je uživatel přihlášen, je spuštěna aktivita pro zobrazení základních statistik o příjmech a výdajích s názvem `HomeActivity`. Pokud tak není, je spuštěna aktivita pro přihlášení uživatele s názvem `LoginActivity`, ze které se lze přepnout do aktivity pro registraci nového uživatele s názvem `RegisterActivity`.

Přihlášený uživatel může využít boční navigační menu, ze kterého jsou dostupné všechny základní aktivity. Toto menu je hlavním navigačním prvkem celé aplikace. Struktura aktivit a jejich základní popis je zobrazen na obrázku 5.4. Pro přehlednost jsou zobrazeny nejdůležitější aktivity aplikace.



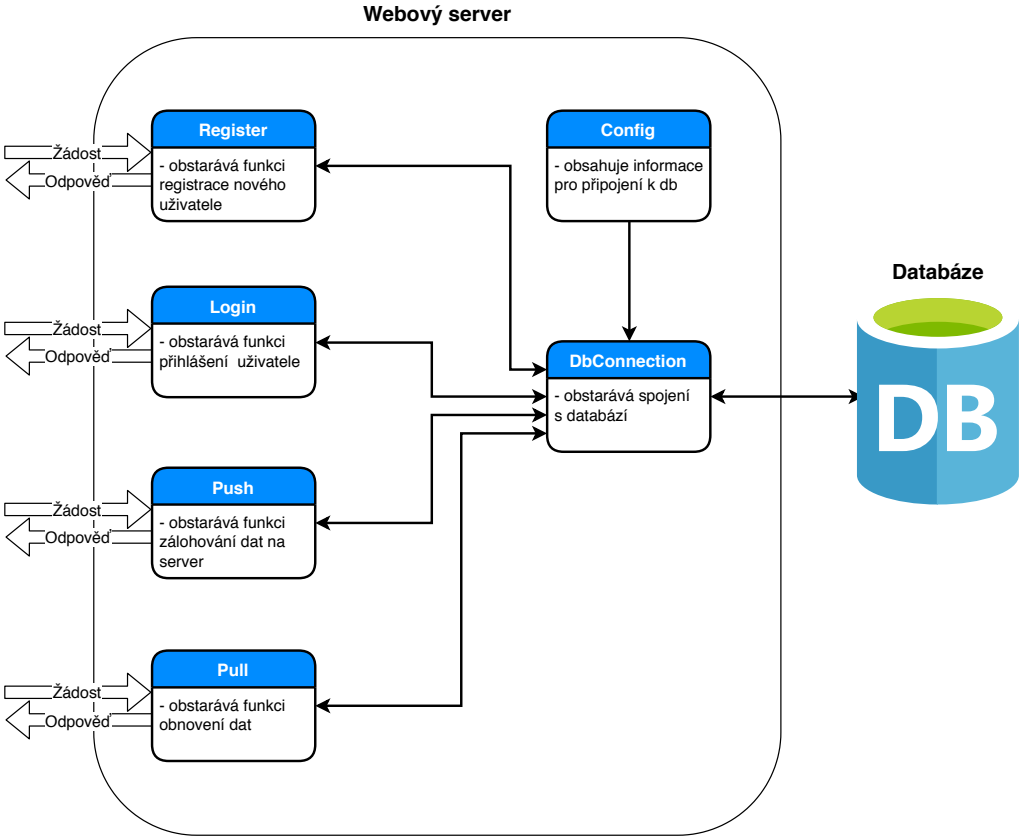
Obrázek 5.4: Struktura aplikace

5.5.2 Struktura serveru

Jelikož aplikace nemusí být se serverem neustále ve spojení a komunikuje s ním pouze na základě žádostí a odpovědí, byl pro tento účel vybrán webový server. Tento server bude přijímat jednotlivé požadavky od klientů, dále pak komunikovat s databází a následně odesílat odpovědi.

Server je tvořen devíti moduly, které mezi sebou navzájem komunikují a obstarávají jednotlivé funkce serveru. Při komunikaci jsou nejprve zavolány moduly s odpovídajícím názvem funkcionality (`login, register, push, pull`). Po té jsou využity moduly pro komunikaci s databází pod názvem `DbConnect` a `Config`, které obsahují potřebné funkce pro navázaní spojení s databází. Další moduly, které server zahrnuje, jsou moduly obsahující pomocné funkce. Tyto moduly se nazývají `basic_functions`, `pull_functions` a `push_functions`.

Po vyhodnocení požadavku je odpověď odeslána k uživateli. Celková struktura serveru je zobrazena na obrázku 5.5. Pro přehlednost jsou na obrázku zobrazeny pouze nejdůležitější moduly.



Obrázek 5.5: Struktura serveru

5.6 Ukládání dat v systému

V tomto projektu je zapotřebí ukládat velké množství nejrůznějších informací. První velkou část informací tvoří uživatelská data, která obsahují informace o příjmech, výdajích, obchodnících a podobně. Další informace, které je zapotřebí ukládat, je nastavení uživatele a pořízené fotografie.

Jednotlivé postupy ukládání dat budou popsány v následujících podkapitolách.

5.6.1 Návrh databázového modelu

V celém projektu jsou využity dvě databáze. První je databáze SQLite na mobilní zařízení a druhá je serverová databáze MariaDB pro zálohu uživatelských dat, registraci a přihlášení. Datové modely obou databází si jsou velice podobné, tudíž zde bude popsán pouze návrh modelu, který je použit na straně mobilní aplikace. Následně budou uvedeny rozdíly mezi databázovým modelem na serveru a na mobilním zařízení.

Datový model mobilní aplikace obsahuje osm tabulek, které jsou mezi sebou propojeny relačními vztahy. V následujícím seznamu jsou popsány jednotlivé tabulky, důležité atributy a jejich využití:

- *user* – Základní tabulka, která obsahuje informace o uživateli a je navázána na ostatní tabulky. Atributy, jež tabulka obsahuje jsou jméno, příjmení, hash hesla uživatele, datum registrace a datum posledního přihlášení. Primárním klíčem této tabulky je identifikační číslo, které je vygenerováno serverem při registraci uživatele a je převzato do databáze v mobilu.

- *identifiers* – Tato tabulka je využita ke generování jednoznačných identifikátorů pro většinu tabulek v tomto modelu. Databáze SQLite má omezené funkce a nelze u ní zapínat a vypínat automatickou inkrementaci jednoznačného identifikátoru záznamu po vytvoření databáze. Tato funkce bude nezbytně nutná při obnově záznamů ze serverové databáze, které již mají svůj vygenerovaný identifikátor. Proto je využita tabulka s názvem `identifiers`, která tuto funkci zastupuje. Tabulka je navázána vztahem `1:1` na tabulku `user` pomocí identifikátoru uživatele. Tento vztah je využit, aby byly tyto informace tematicky odděleny od základních informací o uživateli. Tabulka obsahuje momentálně volné identifikátory k jednotlivým tabulkám. Pokud je vkládán nový záznam, je z této tabulky vybrán volný identifikátor, který je přiřazen k vytvářenému záznamu. Následně je odpovídající identifikátor zvět-

šen o jedničku, aby bylo zařízeno, že je identifikátor v každé tabulce jednoznačný. Primárním klíčem této tabulky je automaticky vytvořený identifikátor.

- *trader* – Tabulka, která obsahuje informace o obchodníkovi a s tabulku `user` je v relačním vztahu `1:N`, tudíž každý uživatel může vytvořit několik obchodníků. Tito obchodníci jsou přístupní pouze uživateli, který je vytvořil. Atributy, které tabulka obsahuje jsou: název, kontaktní osoba, telefonní číslo, IČO, DIČ a adresa obchodníka.

- *note* – Tabulka, která obsahuje informace o poznámce ke každému obchodníkovi. Je navázána na tabulkou `trader` relačním vztahem `1:N`, který umožňuje ke každému obchodníkovi přidat více poznámek. V této tabulce je uložen nadpis poznámky, text poznámky, datum vytvoření a hodnocení obchodníka, které je v rozsahu 0 - 5. Z atributu hodnocení obchodníka je následně vypočítáno průměrné hodnocení, které je zobrazeno v grafu o obchodnících.

- *type* – Tabulka obsahující jednotlivé typy příjmů nebo výdajů, které si může uživatel vytvořit. Tabulka je navázána na tabulku `user` relačním vztahem `1:N`, a dále může být navázána na tabulku s názvem `bill` opět vztahem `1:N`. Avšak vazba na tabulky `bill` není povinná. Tabulka `type` obsahuje atributy s názvem typu a barvou, již si uživatel vybral při vytváření. Barva typu je následně využita pro přehlednější zobrazení statistik.

- *storage_item* – Tabulka, která obsahuje základní informace o skladových položkách. Tato tabulka je navázána relačním vztahem `1:N` na tabulku `user` a také na tabulku `item_quantity` obsahující informace o množství skladové položky. Tabulka obsahuje informace o názvu, jednotkách a poznámce skladové položky.

- *item_quantitiy* – Tabulka, která uchovává informaci o množství skladové položky. Je navázána na tabulku `storage_item` vztahem `1:N`, tudíž každá skladová položka může mít N záznamů v této tabulce. Dále obsahuje nepovinný entitní vztah na tabulku `bill`, který umožňuje přidávat položky s množstvím na fakturu. Základním atributem této tabulky je atribut s názvem `quantity`, který je datového typu `float`. Pokud je hodnota záporná, jedná se o vyskladnění, a pokud je kladná, tak jde o naskladnění.

- *bill* – Tabulka, která obsahuje informace o přijmu nebo výdaji. Zda se jedná o příjem nebo výdaj určuje atribut s názvem `is_expense`,

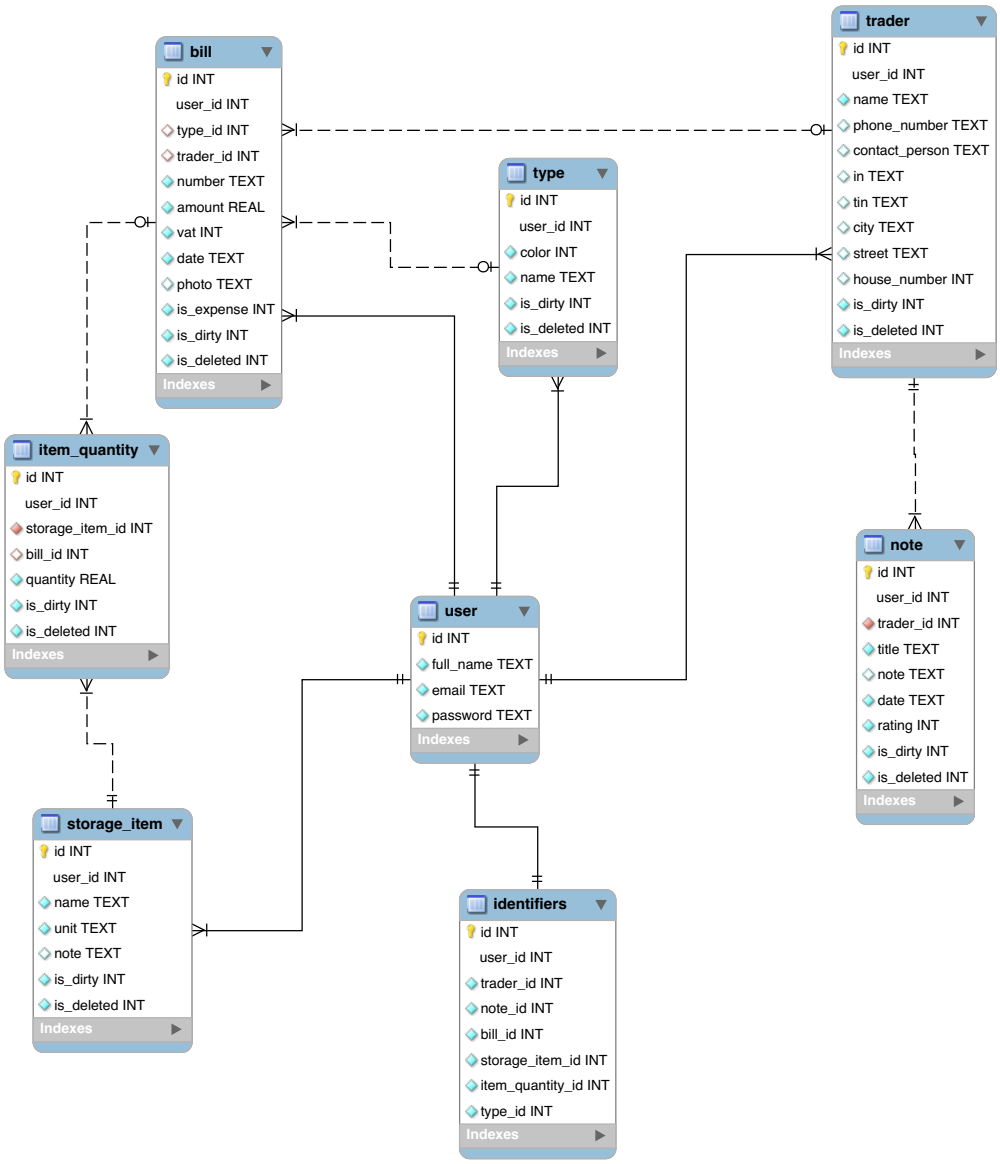
jenž nabývá hodnoty 0 a 1. Tato tabulka je navázána vztahem `1:N` na tabulky `user`. Dále obsahuje nepovinné vazby na tabulky `type`, `trader` a `item_quantity`. Atributy, které obsahuje tato tabulka jsou: název faktury, částka, datum a DPH v procentech.

Primárním klíčem v tabulkách `trader`, `note`, `bill`, `storage_item`, `item_quantity` a `type` byla zvolena dvojice klíčů, jenž je tvořena identifikátorem uživatele a identifikátorem získaným z tabulky `identifiers`. Tato dvojice byla zvolena z důvodu, aby nedocházelo ke kolizím v primárních klíčích při synchronizaci dat se serverem.

Dále jsou ve všech tabulkách, kromě tabulky `user`, obsaženy dva atributy, jež jsou využity při synchronizaci dat se serverem. První ze dvou atributů se nazývá `is_dirty`, který nabývá hodnoty 0 a 1. Záznamy, jež mají atribut `is_dirty` nastaven na jedničku, je nutno zálohovat na server. Tento atribut je nastaven na jedničku pokaždé, když dojde k editaci nebo vytvoření nového záznamu. Jakmile dojde k záloze záznamu na server, je tento atribut nastaven na nulu. Druhým atributem je atribut s názvem `is_deleted`, který nabývá opět hodnot 0 a 1. Tento atribut označuje, zda je záznam smazán či nikoli. Jestliže dojde ke smazání záznamu v aplikaci, nastaví se atributy `is_deleted` a `is_dirty` na jedničku. Tím je zajištěno, že se tento záznam odešle na server jako smazaný. Záznamy, které mají atribut `is_deleted` nastavený na jedničku jsou na serveru následně smazány [12].

Hlavním rozdílem databázového modelu na serveru a na mobilní zařízení je absence tabulky s názvem `identifiers`, která na serveru není potřeba. Primární klíče jsou již vytvořeny na straně mobilní aplikace, a proto není nutné tuto tabulku udržovat na serveru. Dalším rozdílem je absence atributu `is_dirty`, který je využit k záloze dat ze strany mobilní aplikace. Posledním rozdílem jsou dva atributy v tabulce `user`, jež obsahují datum registrace a datum posledního přihlášení uživatele, které jsou uloženy na serveru.

Výše popsaný databázový model, který je použit v mobilní aplikaci, je zobrazen na obrázku 5.6.



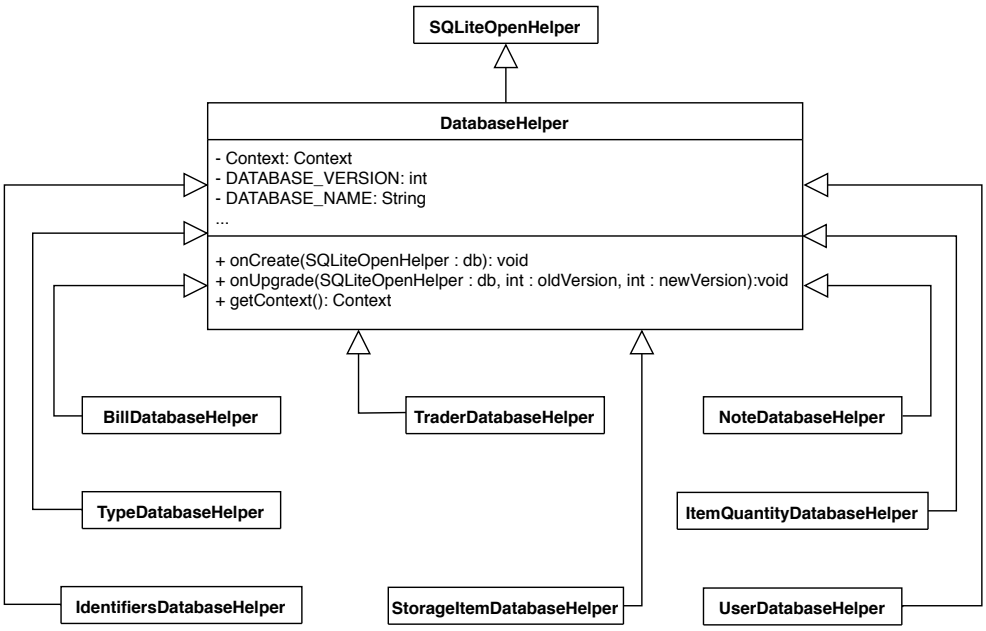
Obrázek 5.6: Návrh databázového modelu

5.6.2 Práce s SQLite databází

SQLite je malá, nenáročná, ale velice výkonná relační embedded databáze, která je součástí operačního systému Android. Databáze SQLite je možno použít na nejrůznějších operačních systémech jako je například Windows 8, iOS, Linux nebo třeba Solaris. Veškerá data jsou uložena v jednom souboru, se kterým databázový engine pracuje. Z funkcionalit klasických databázových serverů, jako je MySQL, SQL Server nebo Oracle podporuje SQLite více tabulek, indexování, podporu triggerů a také vytváření pohledů [21].

Ke komunikaci aplikace s databází SQLite slouží třída s názvem `DatabaseHelper`, která je odvozena od třídy `SQLiteOpenHelper`. Ve třídě `DatabaseHelper` jsou uvedeny názvy tabulek a atributů, SQL příkazy pro vytvoření a smazání jednotlivých tabulek, název a aktuální verze databáze. Třída `DatabaseHelper` přepisuje dvě základní metody, pro řízení životního cyklu databáze, s názvem `onCreate` a `onUpgrade`. Metoda `onCreate` se stará o vytvoření nové databáze za pomocí nadefinovaných SQL příkazů. Pokud dojde ke zvýšení aktuální verze databáze, je zavolána metoda `onUpgrade`, pomocí které lze upravit nebo smazat aktuální databázi, a následně je opět zavolána metoda `onCreate`.

Ke každé tabulce v databázovém modelu existuje třída, která je odvozena od třídy `DatabaseHelper`, jež se stará o práci se záznamy v této tabulce. Každá tato třída obsahuje základní metody pro práci se záznamy jako je například vytvoření, výběr, editace nebo mazání záznamu. Na obrázku 5.7 je znázorněn UML diagram tříd pro práci s databází.



Obrázek 5.7: UML diagram tříd pro práci s databází

5.6.3 Ukládání uživatelského nastavení

V aplikaci je nutno ukládat uživatelské nastavení, které obsahuje například aktuálně vybrané období pro práci se záznamy, možnost vkládání zahraničního IČO a DIČ nebo nastavení automatické zálohy. K ukládání dat o uživatelském nastavení je vhodné využít tzv. Shared preferences.

SharedPreferences umožňují ukládat a spravovat malé množství strukturovaných dat. Pro ukládání je využit externí *xml* soubor, ve kterém jsou uložena data ve formě klíč-hodnota. Jelikož jsou data uložena v externím souboru, tak jsou dostupná i po restartu aplikace [21].

Pro práci s uživatelským nastavením je využita třída `Settings`, která je jedináčkem. Tato třída obsahuje metody pro práci s *SharedPreferences*, mezi nejzákladnější metody patří `saveSettingsToSharedPreferences` a `readSettingsFromSharedPreferences`. Při spuštění aplikace v aktivitě `MainActivity` je zavolána funkce `readSettingsFromSharedPreferences`, která se pokusí načíst nastavení přihlášeného uživatele. Pokud nastavení pro přihlášeného uživatele není dostupné, je použito výchozí nastavení. Pro ukládání nastavení je využita metoda `saveSettingsToSharedPreferences`, která ukládá nastavení do souboru s názvem `settings.xml`. Pro odlišení nastavení uživatelů na jednom zařízení je klíč ukládán v kombinaci s uživatelským identifikátorem.

5.6.4 Ukládání fotografií

Ke každému příjmu nebo výdaji je možno přiřadit fotografii. Lze přiřadit existující fotografii z galerie nebo pořídit novou. V obou případech je v databázi ukládána cesta k cílové fotografii v textovém formátu.

Při volbě fotografie z galerie je spuštěna aktivita pomocí metody `startActivityForResult`, jež po skončení vrátí cestu k vybrané fotografii, která je následně uložena v databázi.

Při pořizování fotografie je nejprve vytvořen soubor, do kterého bude nově pořízená fotografie ukládána. Pro vytvoření jednoznačného názvu souboru, je název tvořen aktuálním datem a časem (formát názvu souboru je `jpg_yyyyMMdd_HHmmss.jpg`). Pokud dojde k vytvoření souboru, je spuštěna aktivita pro pořízení fotografie. Tato aktivita je spuštěna s atributem `EXTRA_OUTPUT`, který umožní uložení pořízené fotografie do předem vytvořeného souboru [5]. Cesta k pořízené fotografii je opět uložena do databáze v textovém formátu.

Při načítání fotografie z cesty uložené v databázi může dojít k načítání velkého množství dat na haldu. Tím by mohlo dojít ke zpomalení celé aplikace, při překročení maximální velikosti haldy i k pádu aplikace. Proto je využita knihovna s názvem `Glide`, která využívá kompresní algoritmy a princip cashování do souboru pro hladké zobrazení fotografie v uspokojivé kvalitě [8].

5.7 Návrh komunikace se serverem

Jak již bylo zmíněno, aplikace komunikuje se serverem formou žádosti a odpovědi. Pro komunikaci se serverem byl zvolen protokol `HTTPS` s využitím funkce `POST`. Data jsou na server zasílána ve formátu určeném pro přenos dat s názvem `JSON`.

Při komunikaci dojde k zavolání URL adresy serveru s názvem odpovídající funkce a předání dat pomocí metody `POST`. Data jsou na serveru příslušnou funkcí zpracována a odpověď je zaslána zpět k mobilní aplikaci. Pro komunikaci se serverem je využita třída s názvem `Server`, jež definuje název serveru, protokol a URL adresy k jednotlivým funkcím serveru.

5.7.1 Registrace uživatele

Pro registraci nového uživatele je využita aktivita `RegisterActivity`, která od uživatele načte potřebná data a odešle je na server. Následně čeká na odpověď. Závazná URL adresa pro registraci uživatele je zobrazena v následujícím rámečku:

```
adresa_serveru/api/register.php
```

Zpráva, která je odesílána na server obsahuje: celé jméno, e-mailovou adresu a hash hesla. Struktura zprávy pro registraci, ve formátu JSON, je:

```
{"password":"heslo","email":"e-mail","full_name":"celé_jméno"}
```

Po zpracování požadavku je serverem odeslána odpověď, jež obsahuje kód výsledku. Struktura zprávy je zobrazena v následujícím rámečku:

```
{"status":"kód_výsledku"}
```

V následujícím seznamu jsou popsány jednotlivé kódy, které zde mohou nastat:

- *0* – Uživatel byl úspěšně zaregistrován.
- *1* – Uživatel se zadanou e-mailovou adresou již existuje.
- *2* – Nebyly vyplněny všechny potřebné informace.

Podle přijatého kódu aplikace informuje uživatele o výsledku registrace a provede odpovídající akci.

5.7.2 Přihlášení uživatele

Pro přihlášení uživatele je využita aktivita `LoginActivity`, která od uživatele načte e-mailovou adresu a heslo. Závazná URL adresa pro přihlášení uživatele je:

```
adresa_serveru/api/login.php
```

Zpráva, odesílaná aplikací na server, obsahuje e-mailovou adresu uživatele a hash hesla. Struktura zprávy je zobrazena v následujícím rámečku:

```
{"password":"heslo","email":"e-mail"}
```

Po zpracování požadavku server odešle odpověď, která obsahuje kód výsledku a identifikátor uživatele, který byl vygenerován serverem. Následně jsou informace o uživateli uloženy do databáze na mobilním zařízení. Struktura odpovědi je v následujícím formátu:

```
{"status":"kód_výsledku","id":"identifikátor_uživatele"}
```

V následujícím seznamu jsou popsány jednotlivé kódy, které zde mohou nastat:

- *0* – Uživatel byl úspěšně autentizován.
- *1* – Nesprávná kombinace hesla a e-mailové adresy.
- *2* – Nebyly vyplněny všechny potřebné informace.

Pokud byl uživatel úspěšně autentizován, je spuštěna aktivita `HomeActivity`, kde jsou dostupné všechny funkce aplikace.

5.7.3 Synchronizace dat

Synchronizaci dat lze rozdělit do dvou částí. První částí je záloha dat na server tzv. *push* a druhou částí je obnova dat ze serveru tzv. *pull*. V obou těchto případech je v hlavičce zprávy uvedena e-mailová adresa a hash hesla pro autentizaci uživatele při synchronizaci dat. Závazná URL adresa pro zálohování a obnovu dat uživatele je:

```
adresa_serveru/api/<push | pull>.php
```

Při zálohování dat na server je využita třída `Push`, která z databáze načte všechny záznamy, jež mají identifikátor `is_dirty` nastaven na jedničku. Dále vytvoří zprávu k odeslání těchto dat na server. Struktura zprávy pro zálohování dat na server je zobrazena v následujícím rámečku:

```
[{"password":"heslo","email":"e-mail"},{"traders":[data]},
  {"notes":[data]}, {"types":[data]}, {"bills":[data]},
  {"storage_items":[data]}, {"item_quantities":[data]}]
```

Jako odpověď je opět přijata zpráva s kódem výsledku, který udává informaci o tom, zda záloha dat proběhla v pořádku. Struktura zprávy je stejná jako odpověď při registraci. Kód výsledku může nabývat dvou následujících hodnot:

- *0* – Záloha dat proběhla úspěšně a data byla uložena na server.
- *1* – Při záloze došlo k chybě a data nebyla uložena na server.

Pokud jsou data na serveru uložena úspěšně, je u všech záznamů nastaven identifikátor `is_dirty` na nulu, a tím již nemusí docházet k opětovnému zálohování.

Zpráva pro obnovu dat obsahuje údaje pro autentizaci uživatele (e-mailová adresa a heslo). Struktura zprávy je stejná jako při přihlášení. Jako odpověď je od serveru přijata zpráva s kódem výsledku a případně s uživatelskými daty. Formát zprávy je následující:

```
    [{"status":"kód_výsledku"},{"traders":[data]},
{"notes":[data]}, {"types":[data]}, {"bills":[data]},
{"storage_items":[data]}, {"item_quantities":[data]}]
```

Kód výsledku může nabývat dvou následujících hodnot:

- *0* – Autorizace uživatele proběhla v pořádku a data ze serveru byla odeslána.
- *1* – Autorizace uživatele neproběhla v pořádku a data nebyla odeslána ze serveru.

Po úspěšném přijetí dat ze serveru dojde ke smazání všech aktuálních záznamů přihlášeného uživatele, aby nedošlo ke kolizi mezi primárními klíči v jednotlivých tabulkách. Následně je využita třída `Pull`, která zpracuje přijatou zprávu a vloží všechny přijaté záznamy do databáze.

5.8 Zabezpečení

V aplikaci mohou být ukládány nejrůznější informace o podnikání uživatele, které mohou být důvěrné, proto je nezbytně nutné, aby byla data zabezpečena. Zabezpečení je zde myšleno proti úniku a následnému zneužití

dat. V následujících podkapitolách jsou uvedeny některé bezpečnostní prvky v různých částech systému.

5.8.1 Zabezpečení aplikace

Pro přístup do aplikace je nutno znát uživatelské jméno a heslo. Heslo je na straně mobilní aplikace uloženo v databázi. Aby nedošlo k úniku a zneužití hesla (například připojením zařízení k pc a přečtením souboru s databází) je uloženo ve formě *hashe*, který je dále zašifrován. Při práci s heslem je heslo nejprve vytvořen hash hesla, a poté se již pracuje pouze s *hashem* hesla. Pro hashování hesla byla zvolena funkce s názvem *SHA-512*, která patří mezi bezpečné hashovací funkce [9]. Pro zašifrování hesla byla zvolena symetrická bloková šifra *AES*.

Dalším bezpečnostním prvkem v aplikaci je úprava vstupních hodnot, které by mohly obsahovat nějaký škodlivý kód. Při ukládání záznamu do databáze je použita funkce pro odstranění speciálních znaků, která by měla tomuto zabránit.

5.8.2 Zabezpečení serveru

Na server je možno zasílat požadavky i bez použití mobilní aplikace, proto je nezbytně nutné zabezpečit vstupní body serveru tak, aby nemohlo dojít k úniku informací. Při komunikaci se serverem jsou všechny vstupní hodnoty ošetřeny funkcí s názvem `removeSpecialCharsInput`, která chrání databázi proti tzv. *SQL injection*.

Dalším zabezpečením je přikládání autentizačních údajů při každé komunikaci se serverem kromě registrace. Na serveru nejprve dojde k ověření kombinace e-mailové adresy a hashe hesla, a až následně je zpracováván požadavek.

5.8.3 Zabezpečení komunikace

Pro zabezpečení komunikace mezi klientskou aplikací a serverem je využito protokolu `HTTPS`, který využívá protokol `HTTP` a `SSL` nebo novější verzi `TLS`. Protokol `HTTPS` zajišťuje autentizaci, ověření certifikátu, šifrování přenášených dat a také integritu [20].

Pro využití protokolu `HTPPS` je nutné mít na serveru nainstalován certifikát podepsaný certifikační autoritou, které důvěřuje operační systém Android. Většina těchto certifikačních autorit má zpoplatněné služby, proto je využita třída s názvem `HttpsTrustManager`, jež povolí všechny certifikační autority.

5.9 Získávání informací

Ve výsledné aplikaci budou zobrazeny informace o aktuálních kurzech základních měn a důležité termíny pro aktuální účetní období. Aby byla zaručena aktuálnost těchto dat, nebudou tyto informace uloženy ve zdrojovém kódu aplikace. Jednotlivé informace budou získávány ze zdrojů, které jsou aktuální.

5.9.1 Načítání aktuálních kurzů

Pro načítání aktuálních kurzů bude využito serveru České národní banky, která zveřejňuje aktuální devizové kurzy. Kurzy na serveru ČNB jsou aktualizovány jedenkrát denně a to vždy ve 14:30. Tyto kurzy jsou dostupné na adrese:

```
http://www.cnb.cz/cs/financni_trhy/devizovy_trh/
kurzy_devizoveho_trhu/denni_kurz.txt
```

Data jsou uložena v textovém souboru, kde první řádka obsahuje hlavičku souboru. Hlavička obsahuje datum, pro který byl kurz vyhlášen ve formátu `DD.MM.RRRR`. Za datem následuje mezera a znak # spolu s pořadovým číslem zveřejňovaných kurzů v rámci roku. Na další řádce jsou popisky jednotlivých hodnot, uložených v souboru, odděleny znakem |. Po této řádce již následují samotná data seřazená abecedně podle názvu země [13]. Příklad části souboru je zobrazen v následujícím rámečku:

```
12.04.2019 #73
země|měna|množství|kód|kurz
Austrálie|dolar|1|AUD|16,246
Brazílie|real|1|BRL|5,840
Bulharsko|lev|1|BGN|13,097
...
```

Pro zobrazování aktuálních kurzů je navržena aktivita `InfoCurrencyActivity`, která se při spuštění pokusí stáhnout soubor s aktuálními kurzy pomocí metody `downloadFileFromServer`. Tato metoda využívá třídu z knihovny Volley s názvem `StringRequest`, jež pomocí metody `GET` získá text z URL adresy předané v parametru. Pokud jsou data úspěšně stažena, dojde k vytvoření lokálního souboru s názvem `denni_kurz.txt`, kde jsou data uložena pro pozdější použití. Následně jsou data s aktuálními kurzy zobrazeny uživateli pomocí metody `setTextToActivity`.

5.9.2 Načítání důležitých termínů

Důležité termíny se mohou v jednotlivých účetních obdobích měnit v závislosti na pracovních a nepracovních dnech. Informace o jednotlivých termínech jsou proto uloženy na serveru, kde je možné tato data jednoduše měnit. Adresa, na které jsou dostupná data, je zobrazena v následujícím rámečku:

```
adresa_serveru/api/files/dates.txt
```

Na této adrese je dostupný textový soubor, který obsahuje informace o důležitých termínech pro aktuální kalendářní rok. Na první řádce je obsažen rok, pro který jsou data aktuální. Na každé další řádce je uveden jeden záznam, který je oddělen středníkem. Záznam obsahuje datum ve formátu `DD.MM.RRRR`, který může být uvozen znakem *. Datum uvozené touto značkou označuje termín pro podnikatele, kteří využívají služeb daňového poradce. Dále je zde uveden význam termínu zakončený středníkem. Záznamy jsou chronologicky seřazeny podle data. Příklad části souboru je zobrazen v následujícím rámečku:

```
2019;
01.01.2019;začátek účetního období roku 2019;
*01.07.2019;termín pro podání daňového přiznání za rok 2018;
31.12.2019;konec účetního období roku 2019;
...
```

Pro zobrazování důležitých termínů v aplikaci je určena aktivita `InfoDateActivity`, jež pracuje s výše popsaným souborem podobně jako v aktivitě pro zobrazování aktuálních kurzů.

6 Popis implementace

V této kapitole bude popsána implementace celkového systému, který se skládá z mobilní aplikace a serveru, navrženého v předchozí kapitole.

6.1 Aplikace

Aplikace Živnostníček byla naprogramována pomocí programovacího jazyka Java, jenž byl vybrán v kapitole 4. Vývoj aplikace probíhal ve vývojovém prostředí s názvem *Android Studio v3.1.2*. Dále bylo využito softwaru *DB Browser for SQLite v3.10.1* pro prohlížení a editaci záznamů v databázi SQLite.

6.1.1 Struktura projektu

Pro mobilní aplikaci je použita struktura projektu vygenerovaná použitým vývojovým prostřením. Tato struktura je závazná pro bezproblémové přeložení aplikace. Struktura obsahuje mnoho souborů a složek. Popis nejzákladnější složek a souborů je popsán v tabulce 6.1.

Tabulka 6.1: Struktura projektu aplikace

Složka/Soubor	*Obsah*
`gradle`	Složka se soubory pro překlad aplikace,
`res`	kořenová složka pro zdroje aplikace,
`res/drawable`	loga a použité ikony,
`res/layout`	rozvržení grafického rozhraní aktivit,
`res/values`	definice řetězců, barev a stylů,
`res/menu`	definice rozvržení menu,
`res/mipmap`	různé verze ikon aplikace,
`res/xml`	konfigurační soubory (např. fotoaparátu),
`libs`	použité knihovny,
`java`	zdrojové kódy aplikace,
`AndroidManifest.xml`	názvy aktivity, verze API, oprávnění apod.

6.1.2 Architektura aplikace

Aplikace je implementována architekturou *Model-view-controller* nebo-li *MVC*. Tato architektura je rozdělena do tří základních komponentů. Komponent *model* se stará o práci s daty a logiku aplikace. *View* se stará o grafické rozhraní aplikace a zobrazování dat. Poslední komponent *controller* se stará o předávání dat mezi modelem a view [4].

V aplikaci zastávají funkci view a controlleru jednotlivé aktivity, které se starají o fungování grafického rozhraní a zobrazování příslušných dat uživateli. Model implementuje funkce pro načítání dat z databáze a ukládání dat do databáze.

6.1.3 Balíky tříd

Jednotlivé třídy jsou děleny do tzv. *balíků*. Balíky tříd jsou umístěny ve složce `java` v balíku `com.ondrejvane.zivnostnicek`. Třídy jsou do jednotlivých balíků rozděleny podle funkce dané třídy. Celkem je projekt tvořen osmi balíky, které zde budou popsány.

`activities`
V tomto balíku jsou obsaženy veškeré aktivity, které aplikace obsahuje. Tento balík je dále dělen do dalších balíků, které jednotlivé aktivity rozděluje do tematických celků pro lepší přehlednost. Jeden z nich je například balík `bill` obsahující aktivity pro práci s fakturami. Obsahuje aktivity pro zobrazení, editaci a vytváření nové faktury.

`adapters`
Balík obsahující jednotlivé adaptéry, které jsou využity k zobrazování záznamů do tzv. *ListView*. Každé třídě jsou v konstruktoru předány data získaná z databáze. Následně jsou data pomocí metody `getView` nastaveny do grafického rozhraní. Všechny adaptéry jsou odvozeny od třídy `BaseAdapter`.

`model`
Balík, který tvoří jeden architektonický komponent s názvem *model*. Tento balík se stará o aplikační logiku a je rozdělen na další dva balíky. První je balík `database`, který obsahuje třídy pro práci s *SQLite* databází. Druhým balíkem je `model_helpers`, jenž zahrnuje tzv. *přepravky*, které jsou využity pro snazší předávání dat mezi modelem a aktivitami. Jména těchto tříd kopírují názvy databázových tabulek, jež s nimi souvisejí.

server
V tomto balíku se vyskytují třídy pro komunikaci se serverem. Obsahuje například třídu `Server`, která je jedináčkem a udržuje informace o serveru (adresu serveru, protokol a adresy jednotlivých funkcí). Dále obsahuje třídy pro synchronizaci dat s názvem `Push` a `Pull`.

session
Balík obsahující třídy pro práci s informacemi o přihlášeném uživateli. Je zde například třída `UserInformation`, která za pomocí třídy `SessionHandler` zjistí informace o aktuálně přihlášeném uživateli. Třída `UserInformation` je jedináčkem a zahrnuje všechny potřebné informace o uživateli. Tato třída zaručuje dostupnost všech těchto informací skrze celou aplikaci. Dále také obsahuje dvě třídy pro ukončení celé aplikace a odhlášení aktuálně přihlášeného uživatele.

utilities
Balík, který obsahuje třídy starající se o nejrůznější vedlejší funkce v aplikaci. Je zde například třída s názvem `WifiCheckerUtility`, která je využita při kontrole připojení k internetu pomocí Wi-Fi. Další třídou obsaženou v tomto balíku je třída `PictureUtility`, která zahrnuje metody pro práci s fotografiemi.

menu
Balík obsahující dvě třídy pro práci s menu v aplikaci. První třídou je třída s názvem `Menu`, která je využita ve všech aktivitách aplikace. Tato třída je využita k přechodu mezi jednotlivými aktivitami skrze celou aplikaci, které jsou dostupné v hlavním bočním menu. Druhou třídou je třída `HomeOptionMenu`, která je využita k přechodu mezi aktivitami pro zobrazení jednotlivých grafů v aplikaci.

helpers
Poslední balík obsahuje třídy, které svojí funkcionalitou nepatří do předchozích popsaných balíků. Patří sem například třída `Settings` pro ukládání uživatelského nastavení do *SharedPreferences* nebo třída pro šifrování a hashování hesla. Dále zahrnuje také třídu `InputValidation`, která obsahuje metody pro validaci některých vstupů. Jeden z těchto vstupů je například IČO u obchodníka.

6.2 Server

Server navržený v předchozí kapitole byl naprogramován pomocí skriptovacího jazyka *PHP*. K vývoji serveru bylo využito vývojové prostředí *PhpStorm v2018.1.3* zaregistrované na studentský účet a pro návrh databázového modelu byl využit program *MySQL Workbench*. Jako serverová databáze byla zvolena databáze MariaDB.

6.2.1 Struktura projektu

Všechny soubory a složky serveru jsou umístěny ve složce `api`. Popis struktury projektu je zobrazen v následující tabulce 6.2.

Tabulka 6.2: Struktura projektu serveru

Složka	*Obsah*
`api`	Soubory starající se o jednotlivé funkce serveru,
`api/db_connection`	soubory pro definování databázového spojení,
`api/files`	složka pro soubory s informacemi,
`api/functions`	soubory s pomocnými funkcemi.

6.2.2 Popis obsahu složek

Jak již bylo zmíněno, struktura projektu je dělena do složek. Podrobný popis jednotlivých složek a obsažených souborů je popsán zde.

`api`
V této složce jsou dostupné skripty, které plní jednotlivé funkce serveru. Soubory, jež jsou zde obsaženy jsou: `login.php`, `register.php`, `push.php` a `pull.php`

`api/db_connection`
Složka obsahující dva skripty pro práci s databází. V prvním skriptu s názvem `config.inc.php` jsou uvedeny konstanty pro spojení s databází. Druhý soubor `DbConnection` se již stará o přímé spojení s databází.

`api/files`
Složka, která obsahuje jeden soubor s názvem `dates.txt`, ve kterém jsou uloženy informace o důležitých termínech, který využívá mobilní aplikace.

`api/functions`
Složka se třemi soubory, které obsahují pomocné funkce, jež jsou využity ostatními skripty. První soubor `basic_functions` zahrnuje základní funkce pro autentizaci uživatele, kontrolu existujícího uživatele apod. Další dva soubory s názvy `push_functions.php` a `pull_functions.php` jsou využity k synchronizaci dat se serverem.

7 Testování

V této kapitole je popsán postup testování aplikace. Nejprve jsou uvedeny zařízení, na kterých byla vytvořená aplikace testována, následně jsou popsány jednotlivé metody testování a jejich výsledek.

7.1 Testovací zařízení

Výsledná aplikace byla testována na fyzických i virtuálních zařízeních. Přehled jednotlivých fyzických zařízení a jejich parametrů je zobrazen v tabulce 7.1. Vývoj aplikace probíhal na zařízení *HTC Desire 610*.

Tabulka 7.1: Fyzická testovací zařízení

HTC Desire 610	
Operační systém:	Android v4.4.2, API 19
CPU:	Qualcomm Snapdragon 400 1,2 GHz
RAM:	1024 MB
Rozlišení obrazovky:	540 x 960 px (4,7")

Samsung GT-I8200N	
Operační systém:	Android v4.2.2, API 17
CPU:	ST-Ericsson U8420 1 GHz
RAM:	1024 MB
Rozlišení obrazovky:	480 x 800 px (4")

Xiaomi Pocophone F1	
Operační systém:	Android v9, API 27
CPU:	Octa-core Max 2,8GHz
RAM:	6144 MB
Rozlišení obrazovky:	2246 x 1080 px (6,18")

Samsung Galaxy J5	
Operační systém:	Android 6.0.1, API 23
CPU:	Exynos 7 Octa 1.6 GHz
RAM:	2048 MB
Rozlišení obrazovky:	1280 x 720 px (5,2")

7.2 Testovací scénáře

V této kapitole budou popsány jednotlivé scénáře, které byly navrhnuty pro otestování nejdůležitějších funkcí aplikace a následně budou uvedeny jejich výsledky. Pro testování navrhnutých scénářů byla využita zařízení uvedená v kapitole 7.1.

7.2.1 Registrace

Popis scénáře
Před spuštěním aplikace se připojíme k internetu. Spustíme aplikaci Živnostníček, která by se měla spustit v aktivitě pro přihlášení. Z této aktivity přejdeme do aktivity pro registraci stisknutím tlačítka „Registrace zde“. V této aktivitě by mělo dojít k zobrazení registračního formuláře. Nejprve vyplníme registrační formulář nevalidními daty (e-mailová adresa ve špatném formátu, příliš krátká hesla). Stiskneme tlačítko „Registrace“ a mělo by dojít k zobrazení hlášky o tom, že jsou vstupní hodnoty nesprávné.

Následně formulář vyplníme validními daty a stiskneme opět tlačítko „Registrace“. Tím by mělo dojít k zobrazení dialogového okna, které informuje uživatele o probíhající registraci. Po úspěšné registraci by se mělo přestat zobrazovat dialogové okno a mělo by dojít k informování uživatele o úspěšné registraci a přepnutí do přihlašovací aktivity.

Dále vyzkoušíme zaregistrovat uživatele se stejnou e-mailovou adresou jako má již registrovaný uživatel, což by nemělo být umožněno. Do registračního formuláře jsou vloženy stejné údaje jako v předchozí registraci a je stisknuto tlačítko „Registrace“. Následně by se měla zobrazit hláška, jež informuje uživatele o již existujícím uživateli.

Popis výsledku
Aplikace reagovala na předem popsaný scénář jak měla a registrace nového uživatele byla provedena. Při pokusu o registraci již vytvořeného uživatele aplikace informovala správně o tom, že tento uživatel již existuje.

7.2.2 Přihlášení

Popis scénáře
Po úspěšně vytvořené registraci uživatele z předchozího scénáře se nacházíme v aktivitě pro přihlášení. Mobilní zařízení musí být stále připojené k internetu. Vyplníme údaje, které jsme použili při registraci a stiskneme tlačítko „Přihlásit“ bez zaškrtnuté volby „Zůstat přihlášen“. Mělo by se zobrazit dialogové okno informující o probíhajícím přihlášení. Po zmizení dialogového

okna by mělo dojít k zobrazení domovské aktivity, ve které jsou dostupné všechny funkce aplikace.

Po restartu aplikace by měla být opět spuštěna aktivita pro přihlášení. Následně budeme stejný scénář opakovat se zaškrtnutou volbou „Zůstat přihlášen“. Tato volba by měla zařídit, že po restartu aplikace se uživatel nemusí znovu přihlašovat.

Popis výsledku
Aplikace reagovala na scénář přihlášení dle očekávání. V obou variantách, s volbou a bez volby „Zůstat přihlášen“, pracovala aplikace podle představ a byla spuštěna domovská aktivita. Dále byl přihlašovací formulář testován pro nesprávné varianty e-mailové adresy a hesla, jež nebyly akceptovány.

7.2.3 Práce s obchodníky a skladovými položkami

Popis scénáře
Tento scénář bude ověřovat, zda korektně funguje vytváření, editace a mazání jednotlivých záznamů obchodníků a skladových položek. Nejprve vytvoříme záznam obchodníka u kterého vyplníme všechny údaje a uložíme.

Přejdeme do aktivity „Obchodníci“, kde stiskneme ikonu v pravém dolním rohu. Po stisknutí by se měla spustit aktivita pro přidání nového obchodníka. Vyplníme všechny požadované údaje a stiskneme tlačítko „Přidat obchodníka“. Následně by mělo dojít k zobrazení zprávy o úspěšném přidání obchodníka, přepnutí do aktivity pro seznam se všemi obchodníky s jedním novým záznamem.

Poté klikneme na vytvořený záznam a měl by se zobrazit detail obchodníka. V pravém horním rohu stiskneme ikonu pro rozbalení menu a vybereme volbu „Editovat“. Po vybrání by se měla zobrazit aktivita pro editaci obchodníka. Změníme název obchodníka a záznam uložíme. Po uložení by se měla změna projevit v náhledu obchodníka.

Následně otevřeme opět menu v pravém horním rohu a vybereme volbu „Smazat“. Mělo by se zobrazit dialogové okno, které se táže, zda chceme opravdu záznam smazat. Stiskneme volbu „Ano“ po které by mělo dojít ke smazání tohoto obchodníka. Scénář budeme opakovat analogickým způsobem pro skladové položky.

Popis výsledku
Reakce aplikace na výše popsaný scénář je dle očekávání jak v případě obchodníků tak i skladových položek. Dochází k vytváření nových záznamů, jednotlivé záznamy lze editovat a mazat.

7.2.4 Práce s příjmy a výdaji

Popis scénáře

Scénář popisující přidávání, editaci a mazání příjmů a výdajů. Bude zde popsán scénář pro výdaj, který je vesměs shodný se scénářem příjmu. Před přidáním výdaje si vytvoříme druh faktury. Přejdeme pomocí bočního menu do aktivity „Výdaje“ a stiskneme tlačítko pro rozbalení menu v pravém horním rohu. Zvolíme volbu „Přidat druh faktury“, po které by se mělo zobrazit dialogové okno pro přidání druhu. Tento dialog vyplníme a uložíme.

Dále stiskneme tlačítko pro přidání výdaje v pravém dolním rohu. Tím by se měla spustit aktivita pro vytvoření výdaje. Vyplníme všechny potřebné údaje. Následně stiskneme volbu „Vyberte druh faktury“, kde byl měl být uveden námi vytvořený druh a vybereme jej. Dále stiskneme ikonu pro přidání fotografie a vybereme volbu použít z fotoaparátu. Tím by mělo dojít ke spuštění aktivity fotoaparátu. Pořídíme fotografii a potvrdíme ji. Následně by se měla zpět zobrazit aktivita pro přidání výdaje s pořízenou fotografií.

Na závěr přidáme dvě položky faktury. První, která již existuje ve skladu a druhou, která dosud ve skladu není. Následně uložíme záznam stisknutím tlačítka „Přidat výdaj“. Po uložení by se výdaj měl ukázat v seznamu výdajů a ve skladu přibýt vyplněné položky.

Dále se pokusíme o editaci výdaje. Stiskne výdaj a měla by se spustit aktivita s podrobným náhledem výdaje. V pravém horním rohu stiskneme volbu pro rozbalení menu a vybereme volbu „Editovat“. Tím dojde k zobrazení aktivity pro editaci záznamu. Změníme název, částku a datum a záznam uložíme stisknutím tlačítka „Uložit výdaj“. Po uložení by se měly editované informace změnit.

Na závěr provedeme smazání výdaje volbou „Smazat“ z rozbalovacího menu v pravém horním rohu. Po této volbě my mělo dojít k zobrazení dialogového okna, které se táže uživatele, zda chce záznam opravdu smazat. Stiskneme volbu „Ano“ a tím by mělo dojít ke smazání záznamu. Celý tento scénář budeme analogicky opakovat pro příjem.

Popis výsledku

Aplikace na popsaný scénář reagovala dle předpokladu a práce s příjmy a výdaji funguje korektně. Při výběru obchodníka se správně zobrazují již přidaní obchodníci. Položky, které byly přidány na fakturu se zobrazily ve skladu se správným množstvím. Přiřazení fotografie k faktuře funguje správně, jak z fotoaparátu, tak z galerie.

7.2.5 Zobrazení grafů

Popis scénáře
Pro testování správného zobrazování grafů vyplníme několik záznamů v záložkách příjmů a výdajů. Jednotlivé záznamy přiřadíme k různým druhům a také vložíme rozdílné termíny.

Přejdeme do domovské aktivity stisknutím volby „Přehled“ v bočním navigačním menu. Zde by se měl zobrazit graf se všemi příjmy a výdaji. V horní části vybereme rok a měsíc, pro který jsme data vkládali. Tím by se měla data v grafu aktualizovat. Podobným způsobem projdeme všechny grafy pomocí rozbalovacího menu v pravé horní části. Každý graf vizuálně zkontrolujeme a přepočteme, zda částky odpovídají.

Popis výsledku
Aplikace na výše popsaný scénář zareagovala dle očekávání. Ve všech grafech jsou korektně zobrazovány hodnoty podle vybraného data. Hodnoty v grafu pro zobrazení DPH jsou správně vypočteny.

7.2.6 Načítání kurzů

Popis scénáře
Scénář popisující ověření funkcionalit pro načítání aktuálních kurzů ze zdroje ČNB [13]. Nejprve zkusíme data aktualizovat bez připojení a až poté s připojením k internetu.

Z bočního menu vybereme volbu „Informace“. Po zobrazení karet s volbami vybereme volbu s názvem „Kurzovní lístek“. Při spuštění aktivity by mělo dojít o pokus k aktualizaci kurzů. Bez připojení k internetu by se měla zobrazit hláška „Připojte se k internetu“. Následně zapneme připojení k internetu a akci opakujeme. Po opětovném spuštění aktivity by mělo dojít k aktualizaci kurzů a zobrazení zprávy o úspěšné aktualizaci kurzů.

Popis výsledku
Aplikace na popsaný scénář reagovala dle předpokladu a došlo úspěšně k aktualizaci všech dostupných kurzů. Po opětovném zapnutí aktivity jsou již dostupné aktuální kurzy.

7.2.7 Záloha dat

Popis scénáře
Při tomto scénáři je zapotřebí připojení k internetu. Z předchozího testování

jsou v aplikaci data, která budou využita k otestování zálohování dat na server. Nejprve bude otestována ruční a poté automatická záloha dat.

Přepneme se pomocí bočního menu do aktivity *Synchronizace dat*. Přeškrtnutá ikona mráčku by měla signalizovat, že data nejsou zálohována. Stiskneme tlačítko „Zálohovat data nyní“, po stisku tlačítka by mělo dojít k zobrazení dialogového okna, které informuje uživatele o probíhajícím zálohování. Po zmizení dialogového okna je uživatel informován o výsledku zálohy. Následně zaškrtneme volbu „Zapnout automatickou zálohu dat“. Přepneme se do aktivity *Výdaje* a přidáme libovolný záznam. Po přepnutí zpět do aktivity *Synchronizace dat* by měla nepřeškrtnutá ikona mráčku signalizovat, že všechna data jsou zálohována.

Popis výsledku
Aplikace se chovala dle očekávání a výše popsaný scénář zálohy splnila bez problémů. Po nahlédnutí do serverové databáze se všechna zálohovaná data shodovala s daty na zařízení.

7.2.8 Obnova dat

Popis scénáře
Při tomto scénáři je opět zapotřebí připojení mobilního zařízení k internetu. K testování obnovy dat ze serveru budou využita data, která byla zálohována v předchozím scénáři.

Před obnovou dat ze serveru smažeme všechna data aplikace pomocí nastavení mobilního zařízení. Aplikaci vypneme a přepneme se do „Nastavení“, kde přejdeme do záložky „Aplikace“. Dále zvolíme „Živnostníček“ a stiskneme tlačítko „Vymazat data“ a potvrdíme dialogové okno.

Dále zapneme aplikaci Živnostníček a znovu se přihlásíme. Po úspěšném přihlášení by v aplikaci neměly být uloženy žádné záznamy. Přejdeme do záložky „Synchronizace dat“ a stiskneme tlačítko „Obnovit data nyní“. Po stisknutí tohoto tlačítka by se mělo zobrazit dialogové okno, které se táže uživatele, zda chce všechna data obnovit. Stiskneme tlačítko „Ano“ a vyčkáme na obnovení dat ze serveru. Po obnovení dat zkontrolujeme, jestli došlo k obnově všech záznamů.

Popis výsledku
Na výše popsaný scénář aplikace reagovala dle očekávání. Obnova dat ze serveru proběhla úspěšně a všechny původní záznamy byly obnoveny.

7.3 Jednotkové testy

Ke každé aktivitě, která je obsažena v aplikaci Živnostníček, byly provedeny jednotkové testy, jež jsou součástí projektu. Každý jednotkový test nese název aktivity doplněný slovem „Test“. Test pro přihlašovací aktivitu je například nazván `LoginActivityTest`.

Jednotkové testy ověřují korektní inicializaci jednotlivých grafických prvků aktivit, které obsahuje. V každém testu dojde k pokusu o inicializaci jednotlivých *view* pomocí metody `findViewById`. Následně je využito metody `assertNotNull`, která ověří, zda došlo ke správnému načtení tohoto *view*.

Všechny jednotkové testy byly spouštěny na zařízení *HTC Desire 610* a proběhly úspěšně.

8 Možná rozšíření aplikace

V této kapitole budou popsány náměty pro možná rozšíření systému, který byl v rámci této práce navrhnut a následně implementován.

8.1 Webová aplikace

Výhodou mobilní aplikace je bezesporu práce s daty z jakéhokoliv místa. Občas však může být pohodlnější pracovat s daty na počítači.

Prvním možným rozšířením je vytvořit webovou aplikaci, která by mohla pracovat s daty, jež jsou uložena v serverové databázi po synchronizaci. Webová aplikace by mohla obsahovat stejné funkce, které jsou implementovány v mobilní aplikaci pro vytváření, editaci a mazání jednotlivých záznamů. Dále by mohla obsahovat některé rozšiřující funkce, které nejsou vhodné pro mobilní zařízení, například vytváření a zobrazování složitějších statistik příjmů, výdajů a skladového hospodářství.

8.2 Generování výkazu

Dalším rozšířením, které je možné přidat do existující aplikace, je generování nejrůznějších výkazů. Tyto výkazy by mohly obsahovat statistiky vypočtené z dat uložených v aplikaci. Mohlo by se například jednat o výkaz obsahující statistiku o příjmech a výdajích za celé účetní období. Následně by také aplikace mohla generovat výkaz obsahující přehled pohybu skladových položek za určité časové období.

8.3 Zálohování fotografií

Na server jsou momentálně odesílány pouze adresy umístění fotografií na lokálním zařízení a nejsou odesílány samotné fotografie. Pokud dojde ke smazání fotografií na lokálním zařízení není je možno obnovit. Proto by bylo dalším možným rozšířením přidat do aktivity „Synchronizace dat“ možnost, kde by si mohl uživatel zvolit, zda chce zálohovat i fotografie přiřazené k jednotlivým příjmům a výdajům.

8.4 Funkce zakázek

Dalším vhodným rozšířením je přidat do aplikace funkci zakázek. V této funkci by bylo možné vytvořit zakázku, která bude v budoucnu realizována. Ke každé zakázce by bylo možno přiřadit obchodníka, potřebný materiál, poznámku, datum realizace a případně částku, jež by po realizaci byla inkasována. Následně by bylo možné zakázku převést mezi příjmy.

8.5 Převod aplikace na platformu iOS

Nezanedbatelnou část trhu tvoří mobilní zařízení s operačním systémem *iOS* od společnosti Apple, proto by bylo dalším vhodným rozšířením přepsat výslednou aplikaci pro platformu *iOS*. Tím by bylo umožněno využívat aplikaci ještě většímu počtu uživatelů.

9 Závěr

V této práci byla navržena a vytvořena mobilní aplikace pro podporu živnostníků na platformě Android, která by mohla být využita k základnímu přehledu o podnikatelské činnosti živnostníka nebo malého podnikatele. Součástí této aplikace je serverová část, která bude využita zejména pro synchronizaci dat uživatele.

Návrhu aplikace nejprve předcházel teoretický rozbor, který se zabýval analýzou existujících aplikací, analýzou potřebných knihoven a rozborem technologií pro platformu Android. S využitím poznatků z teoretického rozboru se praktická část věnovala návrhu celkového systému, který byl následně implementován. Na závěr byl implementovaný systém otestován a byla zmíněna možná rozšíření, která by mohla být tématem některé další práce.

Výsledná aplikace s názvem Živnostníček je zejména určena pro živnostníky a malé podnikatele, kteří by ji mohli využít k základnímu přehledu o příjmech, výdajích, skladových položkách a obchodnících. Ke každému příjmu nebo výdaji lze přiřadit obchodníka, jednotlivé položky faktury a nebo také fotografii účtenky. Dále jsou v aplikaci zobrazeny statistiky v přehledných grafech obsahující informace např. o ročním přehledu příjmů a výdajů, přehledu DPH a nebo hodnocení obchodníků. Tím by aplikace mohla ulehčit a zjednodušit některé administrativní úkony živnostníků. Není však určena pouze pro toto spektrum uživatelů, může být také využita ke správě rodinných a osobních financí.

Poznatky, které byly v této práci zmíněny, by také mohly být využity při vývoji některých dalších aplikací na platformě Android. Ať už se jedná o využití jednotlivých knihoven, komunikaci aplikace se serverem nebo o zabezpečení aplikace.

Seznam zkratek

- **AES** – Advanced Encryption Standard – je standardizovaný algoritmus používaný k šifrování dat v informatice
- **API** – Application Programming Interface – rozhraní pro programování aplikací
- **ART** – Android Runtime – nový virtuální stroj, který v systému Android vytváří běhové prostředí pro aplikace napsané v programovacím jazyce Java
- **CSV** – Comma-separated values – jednoduchý souborový formát určený pro výměnu tabulkových dat
- **HTTP(S)** – Hypertext Transfer Protocol (Secure) – protokol umožňující (zabezpečenou) komunikaci v počítačové síti
- **IDE** – Integrated Development Environment – vývojové prostředí
- **JSON** – JavaScript Object Notation – způsob zápisu dat nezávislých na počítačové platformě určený pro přenos dat
- **JVM** – Java Virtual Machine – modul virtuálního stroje ke spuštění dalších počítačových programů a skriptů vytvořených v jazyce Java
- **MIT** – Massachusetts Institute of Technology – Massachusettský technologický institut
- **NDK** – Native Development Kit – nástroj umožňující programovat v C / C++ pro zařízení Android
- **PDF** – Portable Document Format – univerzální souborový formát pro ukládání dokumentů
- **PHP** – Hypertext Preprocessor – skriptovací programovací jazyk, který je určený především pro programování dynamických internetových stránek a webových aplikací
- **REST** – Representational State Transfer – architektura rozhraní navržená pro distribuované prostředí
- **SQL** – Structured Query Language – strukturovaný dotazovací jazyk používaný pro práci s daty v relačních databázích

- **SSL** – Secure Sockets Layer – protokol poskytující zabezpečenou komunikaci šifrováním a autentizaci komunikujících stran
- **TLS** – Transport Layer Security – nástupce protokolu SSL
- **URL** – Uniform Resource Locator – řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací na Internetu
- **XML** – Extensible Markup Language – obecný značkovací jazyk, který se používá pro serializaci dat

Literatura

[1] *Android 6.0 APIs* [online]. Android developers, 2019. [cit. 2019/04/08]. Dostupné z: `https://developer.android.com/about/versions/marshmallow/android-6.0.html`.

[2] *Android Volley vs Retrofit* [online]. Technology Reaching Us In Time – Online, 2015. [cit. 2019/03/25]. Android Volley vs Retrofit | Better Approach? Dostupné z: `https://www.truiton.com/2015/04/android-volley-vs-retrofit-better-approach/`.

[3] *Apache license* [online]. The Apache Software Foundation, 2004. [cit. 2019/03/24]. Apache license distribution. Dostupné z: `http://www.apache.org/licenses/`.

[4] Bernard, B. *Úvod do architektury MVC* [online]. Zdroják, 2009. [cit. 2019/04/08]. Dostupné z: `https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/`.

[5] *Camera API* [online]. Android developers, 2019. [cit. 2019/04/08]. Dostupné z: `https://developer.android.com/guide/topics/media/camera`.

[6] *Distribution dashboard* [online]. Android developers, 2019. [cit. 2019/04/08]. Dostupné z: `https://developer.android.com/about/dashboards`.

[7] Herout, P. *Učebnice jazyka Java*. KOPP, 2008. ISBN 978-80-7232-355-5.

[8] Judd, S. *About Glide* [online]. GitHub, 2018. [cit. 2019/04/08]. Dostupné z: `https://bumptech.github.io/glide/`.

[9] Kauffman, L. *About Secure Password Hashing* [online]. Android developers, 2013. [cit. 2019/04/08]. Dostupné z: `https://security.blogoverflow.com/2013/09/about-secure-password-hashing/`.

[10] Khan, A. *What Is Odex And Deodex In Android [Complete Guide]* [online]. AddictiveTips, 2010. [cit. 2019/04/08]. Dostupné z: `https://www.addictivetips.com/mobile/what-is-odex-and-deodex-in-android-complete-guide/`.

[11] Kodytek, S. *Úvod do jazyka Kotlin, platformy a IntelliJ* [online]. ITnetwork.cz, 2018. [cit. 2018/03/18]. Úvod do jazyka Kotlin, platformy a IntelliJ. Dostupné z: `https://www.itnetwork.cz/kotlin/zaklady/uvod-do-jazyka-kotlin-platformy-a-intellij`.

[12] Kosik, S. *Synchronization algorithm for exchanging data in the "Client – Server" model via REST API* [online]. Havrl blogspot, 2013. [cit. 2019/04/08]. Dostupné z: http://havrl.blogspot.com/2013/08/synchronization-algorithm-for.html.

[13] *Kurzy devizového trhu na www stránkách ČNB* [online]. Česká národní banka, 2019. [cit. 2019/04/01]. Dostupné z: https://www.cnb.cz/cs/faq/kurzy_devizoveho_trhu.html.

[14] Maheshwari, M. *Top Programming Languages for Android App Development* [online]. DZone, 2018. [cit. 2018/03/18]. Android languages. Dostupné z: https://dzone.com/articles/most-used-programming-languages-for-android-app-de.

[15] *Minimum password length* [online]. Microsoft Corporation, 2017. [cit. 2019/04/08]. Dostupné z: https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/minimum-password-length.

[16] *MIT License* [online]. Technopedia, 2016. [cit. 2019/03/24]. MIT license. Dostupné z: https://www.techopedia.com/definition/3287/mit-license.

[17] *Permissions overview* [online]. Android developers, 2019. [cit. 2019/04/08]. Dostupné z: https://developer.android.com/guide/topics/permissions/overview.

[18] *Smartphone Market Share* [online]. IDC Corporate, 2018. [cit. 2018/03/18]. Smartphone Market Share. Dostupné z: https://www.idc.com/promo/smartphone-market-share.

[19] *TIOBE Index for March 2019* [online]. TIOBE, 2019. [cit. 2018/03/18]. TIOBE Index programming languages. Dostupné z: https://www.tiobe.com/tiobe-index/.

[20] *What is SSL, TLS and HTTPS?* [online]. Symantec Corporation, 2018. [cit. 2019/04/08]. Dostupné z: https://www.websecurity.symantec.com/security-topics/what-is-ssl-tls-https.

[21] Ľuboslav, L. *Vývoj aplikací pro Android*. Computer press, 2015. ISBN 978-80-251-4347-6.

Přílohy

A Instalační příručka

Tato příloha obsahuje návod k instalaci vytvořené mobilní aplikace a serveru.

A.1 Aplikace

Pro instalaci a spuštění aplikace bude zapotřebí mobilní zařízení s operačním systémem Android verze 4.1 (Jelly Bean) nebo vyšší.

Před samotnou instalací aplikace nejprve musíme v mobilním zařízení povolit instalaci z neznámých zdrojů. Pro toto povolení musíme otevřít nastavení telefonu a zvolíme záložku „Zabezpečení“. Poté zaškrtneme volbu „Neznámé zdroje“, která nám umožní instalaci aplikace Živnostníček viz obrázek A.1a na další straně.

Poté připojíme mobilní zařízení k počítači pomocí USB a zkopírujeme soubor `zivnostnicek.apk`, který je uložen ve složce `/instalace/aplikace` na přiloženém CD. V mobilním zařízení umístíme soubor do některé složky, do které máme přístup (například složka `Download`). Následně spustíme na mobilním zařízení Správce souborů a přejdeme do složky, ve které se nachází soubor `zivnostnicek.apk`. Klikneme na tento soubor a spustí se průvodce instalací aplikace, kde již dále postupujeme podle pokynů.

Po úspěšné instalaci se v nabídce objeví ikona aplikace Živnostníček viz obrázek A.1b na další straně.

A.2 Server

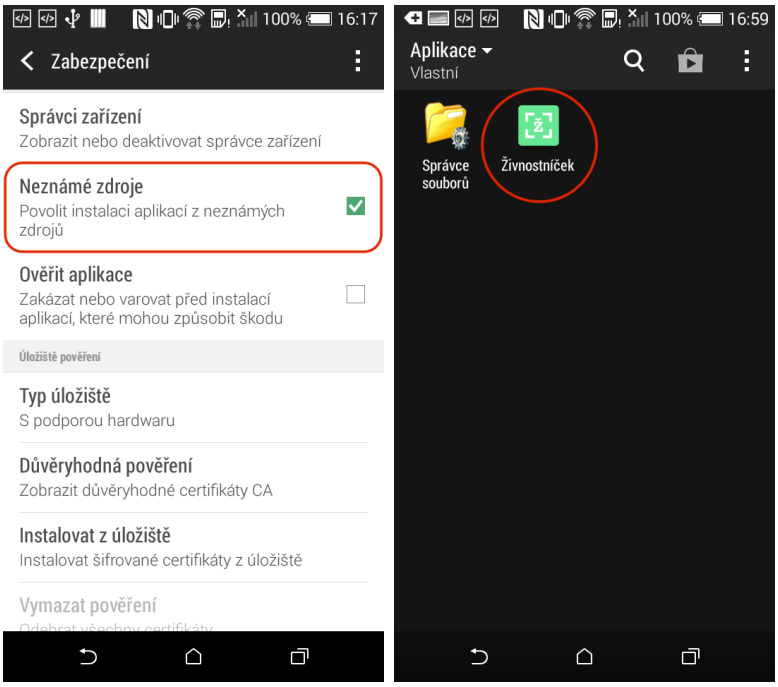
Pro instalaci serverové části bude zapotřebí webového serveru podporující služby HTTP, například veřejně dostupný *Apache*. Na serveru bude dále zapotřebí skriptovacího jazyka PHP s minimální verzí 7.3.2 a relační databáze MariaDB s minimální verzí 10.3.14.

Nejprve musíme zkopírovat adresář s názvem `api`, který je uložen ve složce `/instalace/server`, na přiloženém CD, do kořenové složky webového prohlížeče (pro Apache se například složka nazývá `htdocs`). Struktura složek a souborů v adresáři `api` musí být nutně zachována.

Poté je zapotřebí vytvořit databázi, se kterou bude server pracovat. Pro vytvoření databáze využijeme nástroje *phpMyAdmin* a soubor s názvem `create_db.sql`, který je uložen ve složce `/instalace/server/databaze` na

přiloženém CD. Spustíme nástroj *phpMyAdmin* a přihlásíme se pomocí uživatelského jména a hesla. Po úspěšném přihlášení stiskneme tlačítko „SQL“ v horním menu. Zobrazí se okno pro zapisování SQL příkazů. Do toho okna vložíme celý obsah souboru `create_db.sql` a stiskneme tlačítko „Proveď“. Tím by měla být databáze vytvořena.

Následně musíme nastavit parametry pro připojení k databázi. Tyto parametry lze nastavit v souboru `config.inc.php`, který je uložen ve složce `./api/db_connection`. Po nastavení těchto parametrů lze využít skript s názvem `test.php`, který otestuje spojení s databází. Otevřeme si webový prohlížeč a do okna pro URL adresu napíšeme `adresa_serveru/api/test.php`. Pokud je vše správně nastaveno, tento skript vypíše na obrazovku „success“. V opačném případě vypíše chybu, kvůli které se spojení s databází nezdařilo. Po úspěšném otestování spojení je možno tento soubor smazat.



(a) Povolení instalace aplikací z neznámých zdrojů

(b) Ikona pro spuštění aplikace Živnostníček

Obrázek A.1: Instalace aplikace Živnostníček

B Uživatelská příručka

V této příloze je uvedena uživatelská příručka pro obsluhu základních funkcí aplikace Živnostníček.

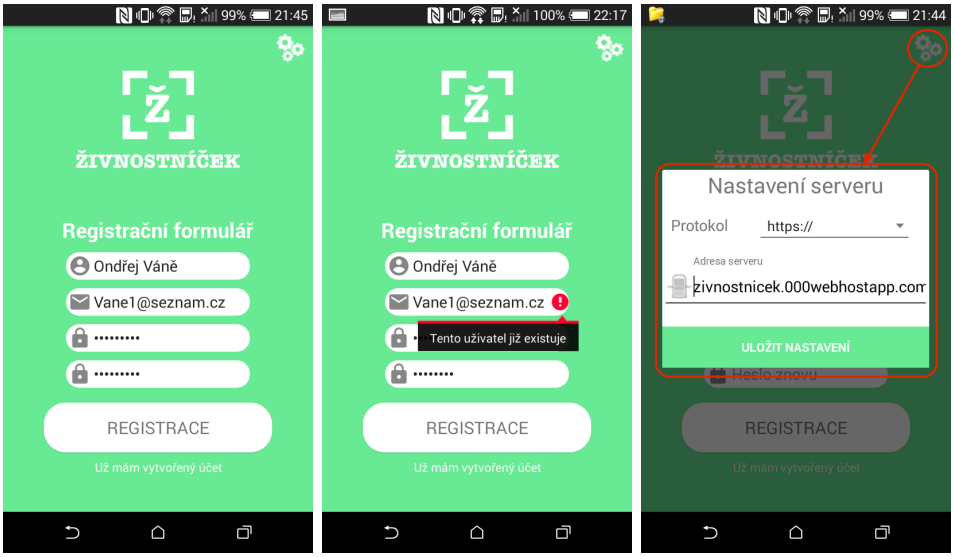
B.1 Registrace uživatele

Před využíváním funkcí samotné aplikace je nejprve nutná registrace nového uživatele. Do registračního formuláře se dostaneme stisknutím tlačítka „Registrace zde“ v přihlašovací aktivitě zobrazené na obrázku B.3a.

Registrační formulář obsahuje pole pro celé jméno, e-mailovou adresu a heslo uživatele viz obrázek B.2a. Po vyplnění všech těchto údajů stiskneme tlačítko „Registrace“. Pokud dojde k úspěšné registraci je aplikace přepnuta do aktivity pro přihlášení.

Jestliže uživatel se stejnou e-mailovou adresou již existuje, je vypsána chybová hláška, která o této skutečnosti informuje uživatele viz obrázek B.2b.

Další funkcí této aktivity je nastavení serveru, vůči kterému bude probíhat registrace, přihlášení a synchronizace dat. Toto nastavení je zobrazeno po stisknutí ikony v pravém horním rohu viz obrázek B.2c. Zde je možno nastavit protokol a URL adresu webového serveru, se kterým bude aplikace komunikovat.



(a) Registrační formulář pro nového uživatele (b) Chybová hláška při registraci uživatele (c) Dialogové okno pro nastavení serveru

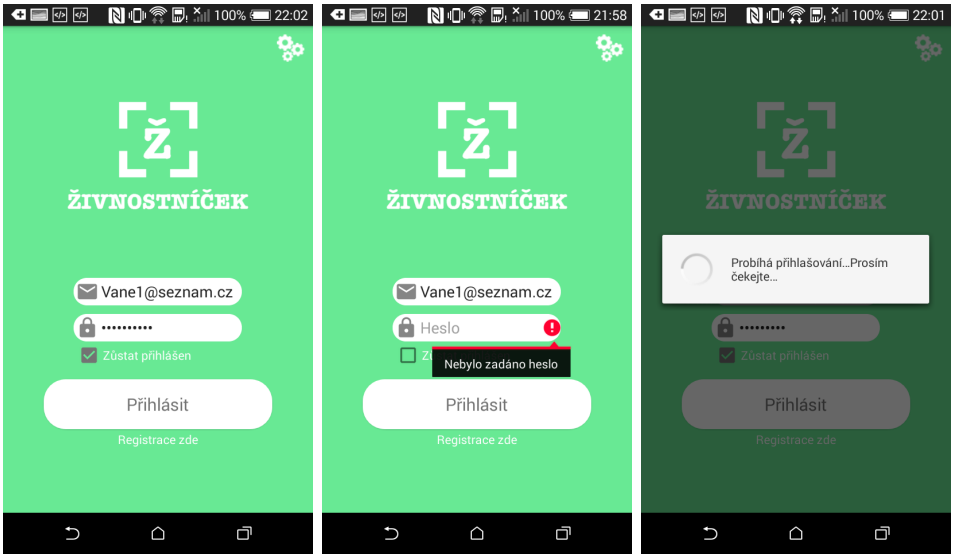
Obrázek B.2: Registrace nového uživatele

B.2 Přihlášení uživatele

Po úspěšné registraci je aplikace přepnuta do aktivity pro přihlášení, která obsahuje přihlašovací formulář. Tento formulář zahrnuje dvě vstupní pole. První je pole pro e-mailovou adresu zaregistrovaného uživatele a druhé je pole pro heslo uživatele viz obrázek B.3a.

Nejprve je nutno všechna pole správně vyplnit a poté stisknout tlačítko „Přihlásit“. Pokud nedojde k vyplnění některého z polí, je vypsána chybová hláška viz obrázek B.3b.

V průběhu procesu přihlašování je zobrazeno dialogové okno, které informuje uživatele o probíhajícím přihlašování viz obrázek B.3c.



(a) Přihlašovací formulář pro uživatele

(b) Chybová hláška při přihlášení uživatele

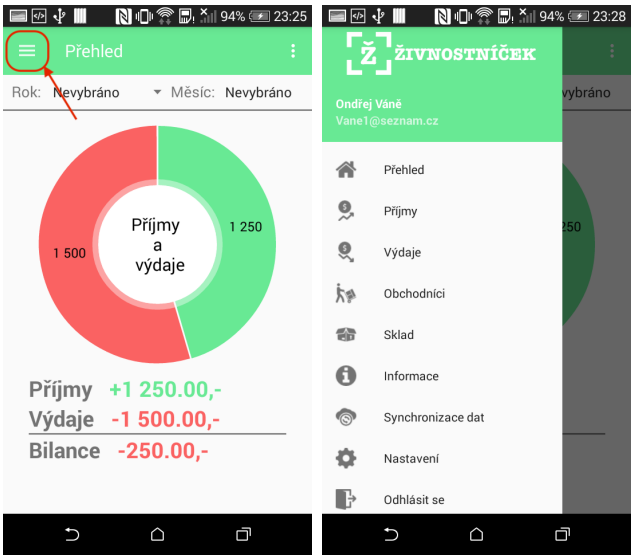
(c) Dialogové okno probíhajícího přihlašování

Obrázek B.3: Přihlášení uživatele

B.3 Navigační menu

Po úspěšném přihlášení uživatele je spuštěna domovská aktivita, ve které je zobrazen základní graf přehledu příjmů a výdajů viz obrázek B.4a. Pokud se jedná o první přihlášení, tak v aplikaci nejsou dostupná žádná data k zobrazení.

Po stisknutí ikony v levém horním rohu (viz obrázek B.4a) je zobrazen hlavní navigační panel celé aplikace, ze kterého jsou dostupné jednotlivé funkcionality aplikace viz obrázek B.4b. V tomto navigačním menu jsou také zobrazeny základní údaje o přihlášeném uživateli, jako je jméno, příjmení a e-mailová adresa.



(a) Ikona pro zobrazení navigačního menu

(b) Boční navigační panel aplikace

Obrázek B.4: Navigační menu aplikace

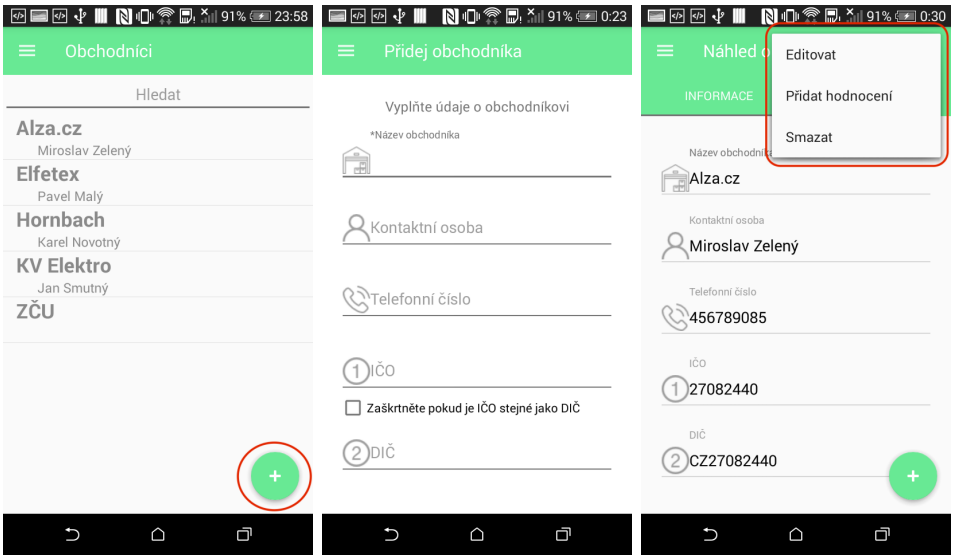
B.4 Správa obchodníků

Stisknutím záložky „Obchodníci“ v bočním navigačním menu se přepneme do aktivity, která se stará o vytváření, náhled, editaci, mazání a hodnocení jednotlivých obchodníků.

Po přepnutí do této aktivity je zobrazen seznam všech existujících obchodníků a po kliknutí na ikonu v pravém dolním rohu lze přidat nového viz obrázek B.5a. Po stisknutí tohoto tlačítka se spustí aktivita obsahující formulář pro vytvoření nového obchodníka (viz obrázek B.5b). Stisknutím tlačítka „Přidejte obchodníka“ se uloží nový záznam a je zobrazen v celkovém seznamu všech obchodníků.

Pro náhled obchodníka stačí kliknout na jeho záznam v seznamu, a tím se spustí aktivita pro náhled, v níž lze zobrazit adresu obchodníka na mapě stisknutím tlačítka „Zobrazit na mapě“. Pro editaci, mazaní nebo přidání nového hodnocení lze využít menu v pravé horní části viz obrázek B.5c.

Pro přidání hodnocení k vybranému obchodníkovi stačí stisknout tlačítko „Přidat hodnocení“. Aktivita pro přidání hodnocení je zobrazena na obrázku B.6a. Pro uložení tohoto hodnocení stiskneme tlačítko „Přidat hodnocení“. Pro zobrazení všech hodnocení s vypočteným průměrem stačí přejít do záložky „Hodnocení“ viz obrázek B.6b.

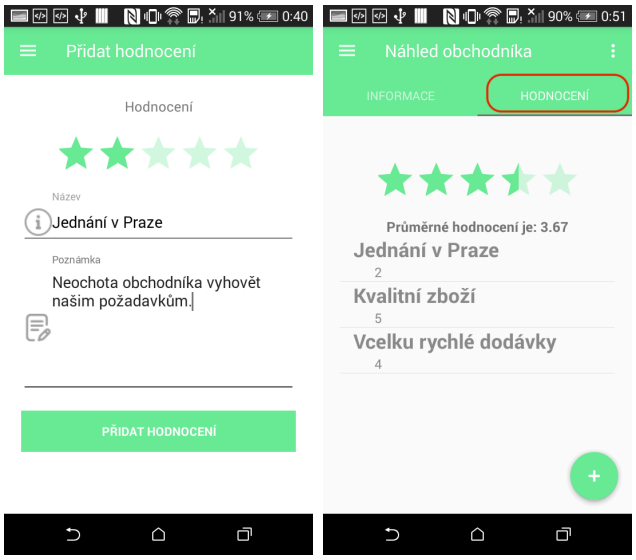


(a) Seznam všech obchodníků

(b) Formulář pro vyplnění nového obchodníka

(c) Menu pro práci se záznamem obchodníka

Obrázek B.5: Správa obchodníků



(a) Formulář pro vytvoření hodnocení

(b) Zobrazení přehledu hodnocení obchodníka

Obrázek B.6: Vytváření a náhled hodnocení

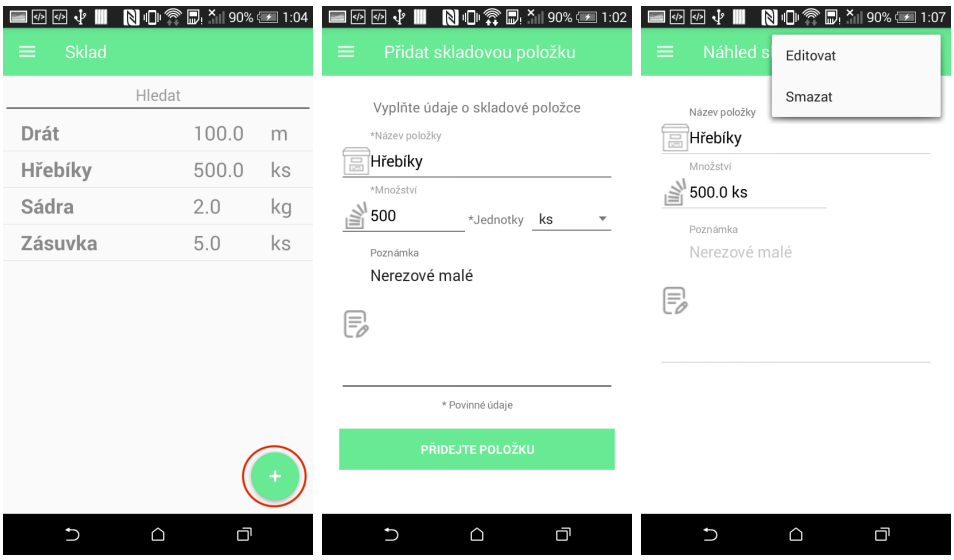
B.5 Správa skladových položek

Pro správu skladových položek přejdeme do záložky „Sklad“, kde jsou zobrazeny jednotlivé položky ve skladu a jejich množství viz obrázek B.7a.

Pro přidání nové skladové položky stiskneme tlačítko v pravém dolním rohu aktivity viz obrázek B.7a. Následně se zobrazí formulář pro přidání nové skladové položky. Po vyplnění formuláře stiskneme tlačítko „Přidejte položku“ viz obrázek B.7b a nová položka se zobrazí v seznamu.

Náhled skladové položky je dostupný po kliknutí na vybranou položku v seznamu. Po stisknutí vybrané skladové položky se zobrazí náhled viz obrázek B.7b, kde jsou zobrazeny informace o skladové položce.

Pro editaci nebo smazání záznamu lze využít pomocné menu, které se zobrazí po kliknutí na ikonu v pravém horním rohu viz obrázek B.7c.



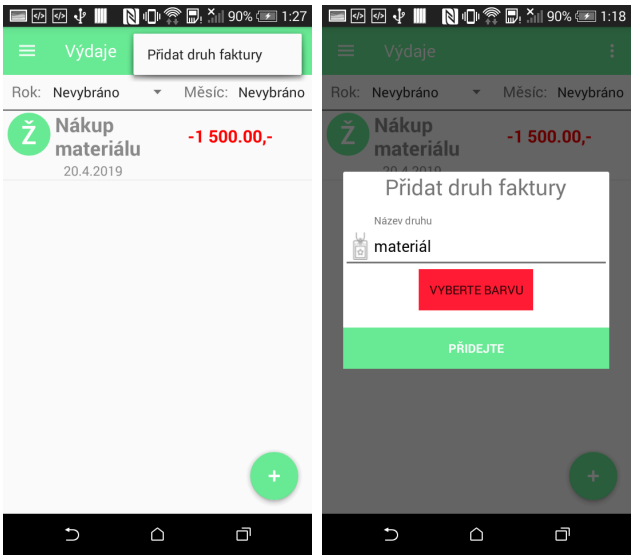
(a) Seznamu skladových položek (b) Formulář pro vytvoření skladové položky (c) Menu pro editaci skladové položky

Obrázek B.7: Správa skladových položek

B.6 Vytváření druhů faktur

Ke každé faktuře může být přiřazen druh, který lze vytvořit. Pro vytvoření nového druhu přejdeme do záložky „Příjmy“ nebo „Výdaje“ a stiskneme ikonu v pravém horním rohu aktivity viz obrázek B.8a.

Tím dojde k zobrazení dialogového okna pro přidání druhu, kde lze vyplnit název a přiřadit mu libovolnou barvu viz obrázek B.8b. Pro uložení tohoto druhu stiskneme tlačítko „Přidejte“. Poté již můžeme k příjmům a výdajům přiřazovat jednotlivé druhy.



(a) Formulář pro vytvoření hodnocení

(b) Zobrazení přehledu hodnocení obchodníka

Obrázek B.8: Vytváření druhu

B.7 Správa příjmů a výdajů

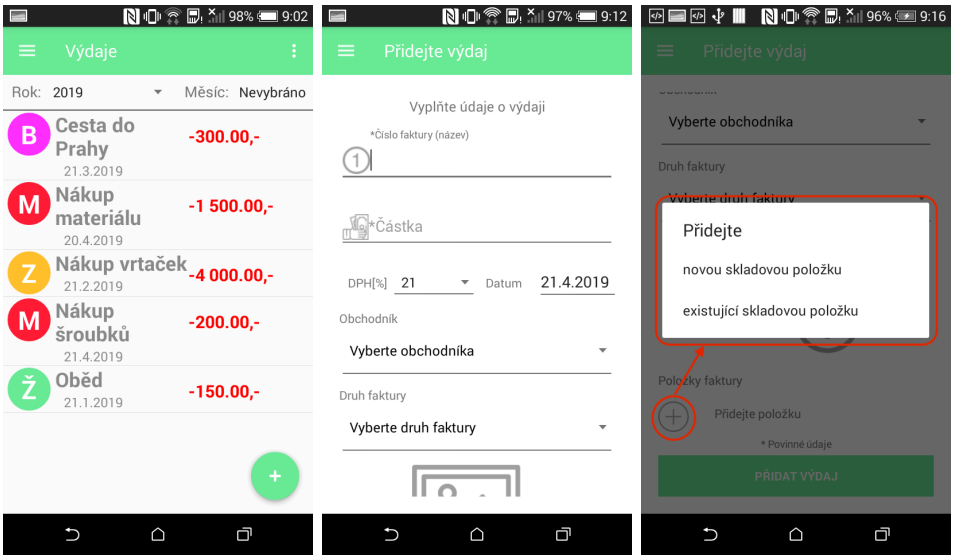
Do aktivity pro správu příjmů a výdajů lze přejít pomocí bočního menu stisknutím záložky „Příjmy“ nebo „Výdaje“. V této kapitole bude popsána pouze práce s výdaji, jelikož práce s příjmy funguje analogicky.

V horní části aktivity pro správu výdajů jsou dvě políčka, ve kterých lze zvolit rok a měsíc, pro které se mají výdaje zobrazovat. Dále je zde seznam výdajů s názvem, částkou a přiřazeným druhem. Počáteční písmeno a barva druhu je uvedena v kolečku u každého výdaje viz obrázek B.9a. Pokud není druh přiřazen je v kolečku písmeno „Ž“ jako žádný.

Pro přidání nového výdaje stiskneme tlačítko v pravém dolním rohu viz obrázek B.9a. Poté se spustí aktivita, ve které lze vyplnit údaje o výdaji viz obrázek B.9b. V této aktivitě lze vyplnit název, částku, datum, DPH, druh výdaje, obchodníka, fotografii nebo jednotlivé položky faktury viz obrázek B.9b.

Položku na fakturu lze přidat tlačítkem se znakem plus viz obrázek B.9c. Tím se spustí dialogové okno, které se táže uživatele, zda chce pouze přidat množství k existující položce nebo vytvořit novou skladovou položku.

Po vyplnění všech údajů, uložíme výdaj tlačítkem „Přidat výdaj“. Editace a mazání jednotlivých záznamů výdajů funguje analogicky, jako v případě skladových položek.



(a) Seznamu jednotlivých výdajů

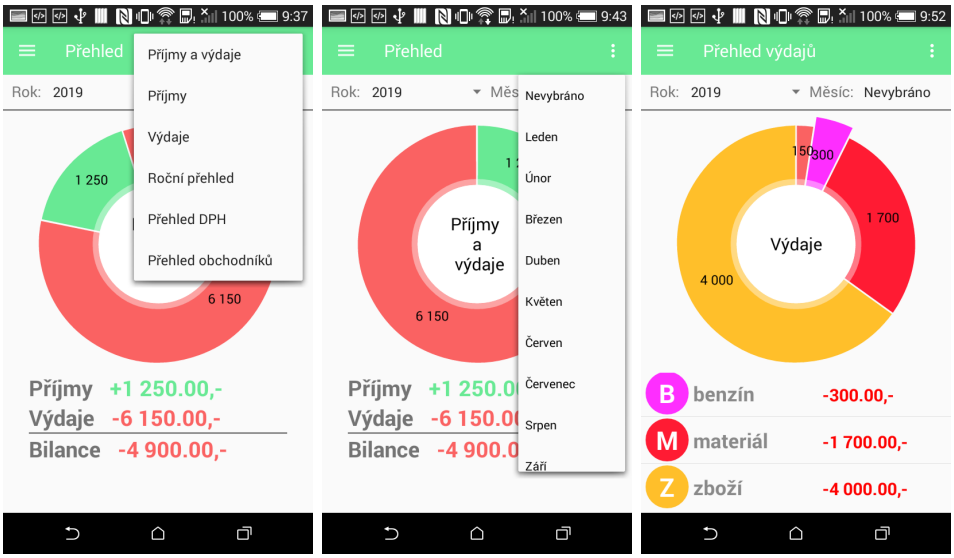
(b) Formulář pro vytvoření výdaje

(c) Dialogové okno pro přidání položky faktury

Obrázek B.9: Správa výdajů

B.8 Zobrazení jednotlivých grafů

Pro zobrazení grafů nejprve musíme přejít do domovské aktivity stisknutím záložky „Přehled“ v bočním menu.



(a) Menu pro přechod mezi grafy

(b) Volba roku a měsíce pro zobrazení dat

(c) Graf zobrazující výdaje za rok 2019

Obrázek B.10: Zobrazení grafů

Pro přechod mezi jednotlivými grafy je určeno menu, které se zobrazí po stisknutí tlačítka v pravém horním rohu viz obrázek B.10a. U většiny grafů lze nastavit rok a měsíc, pro který jsou data v grafu zobrazována viz obrázek B.10b.

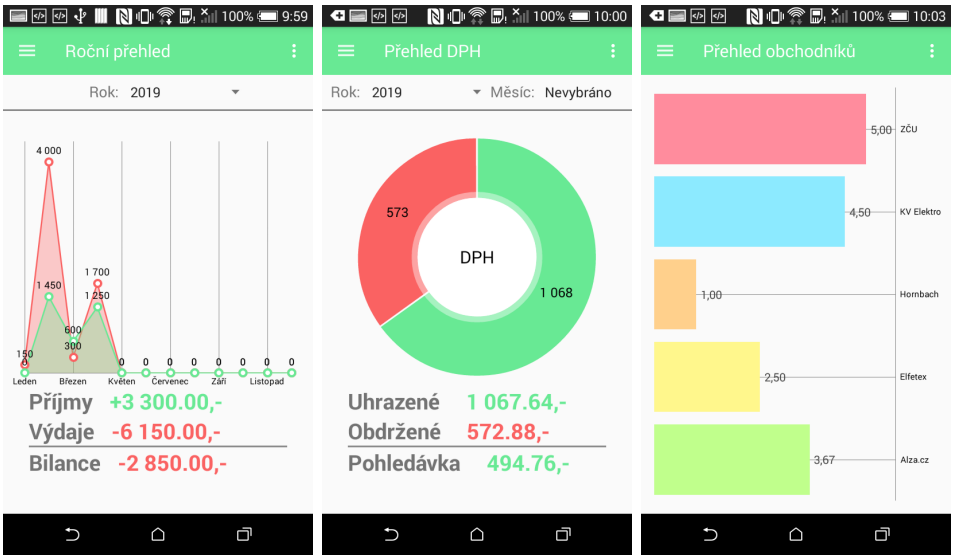
Graf s názvem „Příjmy a výdaje“ zobrazuje kumulativní součet všech příjmů a výdajů podle vybraného období. Pod grafem jsou zobrazeny vypočtené hodnoty a také bilance příjmů a výdajů viz obrázek B.10b.

Grafy s názvem „Příjmy“ a „Výdaje“ zobrazují podrobnější přehled kumulativních příjmů a výdajů členěných podle druhů. Na obrázku B.10c je zobrazen graf výdajů. V dolní části aktivity je uveden seznam s jednotlivými druhy a k nim odpovídající částka.

Další graf s názvem „Roční přehled“ zobrazuje bilanci příjmů a výdajů v jednotlivých měsících viz obrázek B.11a. V horní části lze vybrat rok, pro který jsou data zobrazována.

Graf s názvem „Přehled DPH“ zobrazuje kumulativní součet vypočteného DPH příjmů a výdajů viz obrázek B.11b. V horní části lze opět vybrat rok a měsíc, za který mají být data zobrazována. V dolní části obrazovky je vypočtená bilance DPH.

Posledním grafem v aplikaci je graf s názvem „Přehled obchodníků“, který zobrazuje průměrné hodnocení všech obchodníků viz obrázek B.11c.



(a) Graf zobrazující roční přehled (b) Graf zobrazující vypočtené DPH (c) Graf zobrazující hodnocení obchodníků

Obrázek B.11: Zobrazení grafů

B.9 Informace

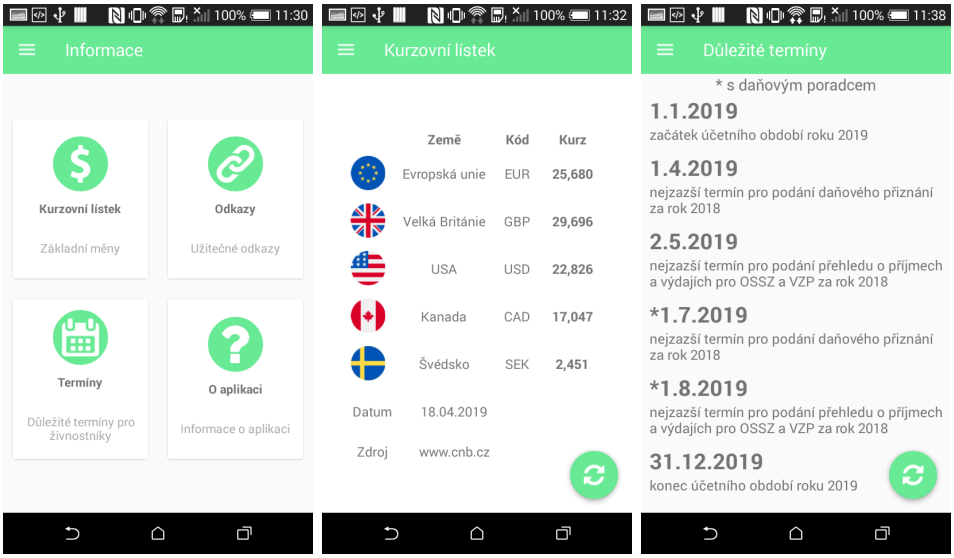
Pro zobrazení informací stiskneme záložku „Informace“ v hlavním navigačním menu. Tím se zobrazí menu karet, které je určené pro přechod mezi jednotlivými aktivitami viz obrázek B.12a.

První záložka je kurzovní lístek, jenž obsahuje aktuální kurzy získané z ČNB viz obrázek B.12b. Pokud je mobilní zařízení připojené k internetu, jsou kurzy aktualizovány při startu této aktivity. Dále je také možno využít tlačítka v pravém dolním rohu pro ruční aktualizaci kurzů.

Další záložkou jsou „Termíny“ obsahující informace o aktuálních termínech pro podnikatele viz obrázek B.12c. Tyto informace jsou staženy ze serveru opět při startu aktivity. Pro ruční aktualizaci termínů je určeno tlačítko v pravém dolním rohu. Při stisknutí termínu se zobrazí dialogové okno pro přidání události do kalendáře.

Dále je zde dostupná aktivita pro zobrazení některých vybraných odkazů, které mohou být užitečné pro podnikatele. Po stisknutí odkazu se spustí aplikace pro prohlížení webu a je zde zobrazena odpovídající stránka.

V poslední kartě s názvem „O aplikaci“ jsou uvedeny základní informace o aplikaci Živnostníček.



(a) Menu pro přechod mezi informacemi

(b) Přehled základních kurzů

(c) Aktivita zobrazující důležité termíny

Obrázek B.12: Zobrazení informací

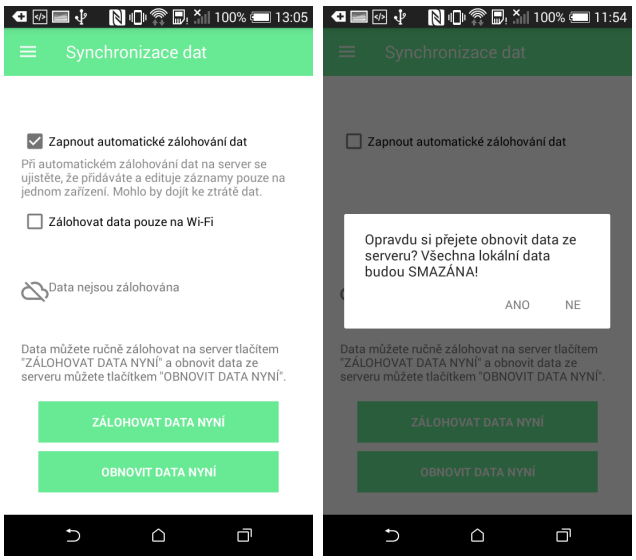
B.10 Synchronizace dat

V původním nastavení aplikace je automatické zálohování záznamů vypnuto. Pro provedení ruční zálohy nebo pro nastavení automatické zálohy přejdeme do aktivity „Synchronizace dat“ pomocí bočního menu.

Pro zapnutí automatické zálohy stačí zaškrtnout volbu „Zapnout automatickou zálohu dat“. Pokud chcete provádět zálohu pouze s připojením k Wi-Fi, můžete zaškrtnout volbu „Zálohovat data pouze na Wi-Fi“. Ikona mráčku signalizuje, zda jsou všechna data zálohována viz obrázek B.13a.

Ruční zálohu lze provést stisknutím tlačítka „Zálohovat data nyní“ viz obrázek B.13a.

Pro obnovu dat stiskneme tlačítko „Obnovit data nyní“ a následně se zobrazí dialogové okno, které se táže uživatele, zda chce opravdu data obnovit viz obrázek B.13b.



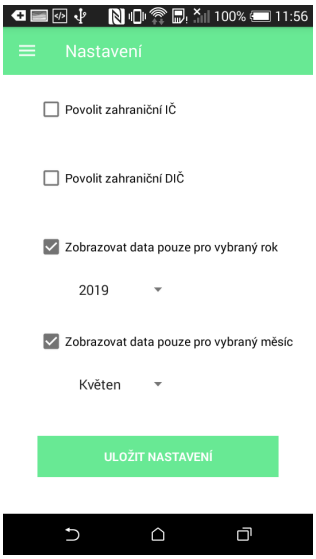
(a) Nastavení automatické zálohy dat

(b) Obnova dat ze serveru

Obrázek B.13: Nastavení synchronizace dat

B.11 Nastavení

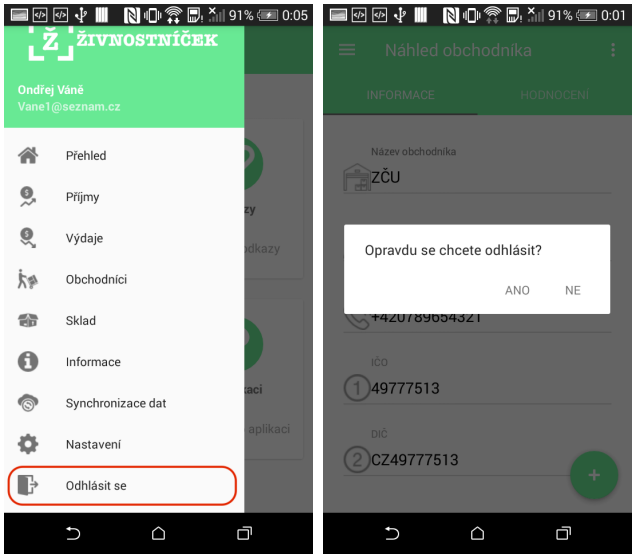
V záložce „Nastavení“ je dostupné základní nastavení aplikace. Je zde například povolení vkládání zahraničního IČO a DIČ (mohou mít odlišný formát). Dále je zde také možné nastavit rok a měsíc. Tyto vybrané hodnoty budou používány skrze celou aplikaci pro zobrazování grafů, příjmů a výdajů. Tím mohou být například oddělena jednotlivá účetní období. Aktivita nastavení je zobrazena na obrázku B.14.



Obrázek B.14: Nastavení aplikace

B.12 Odhlášení uživatele

Pro odhlášení uživatele stačí kliknout na záložku s názvem „Odhlásit se“ ve spodní části navigačního menu, viz obrázek B.15a. Následně je zobrazeno dialogové okno, které se táže uživatele, zda se chce opravdu odhlásit viz obrázek B.15a. Po stisknutí tlačítka „Ano“ dojde k odhlášení uživatele a je spuštěna aktivita pro přihlášení.



(a) Volba v menu pro odhlášení uživatele

(b) Dialogové okno potvrzující odhlášení

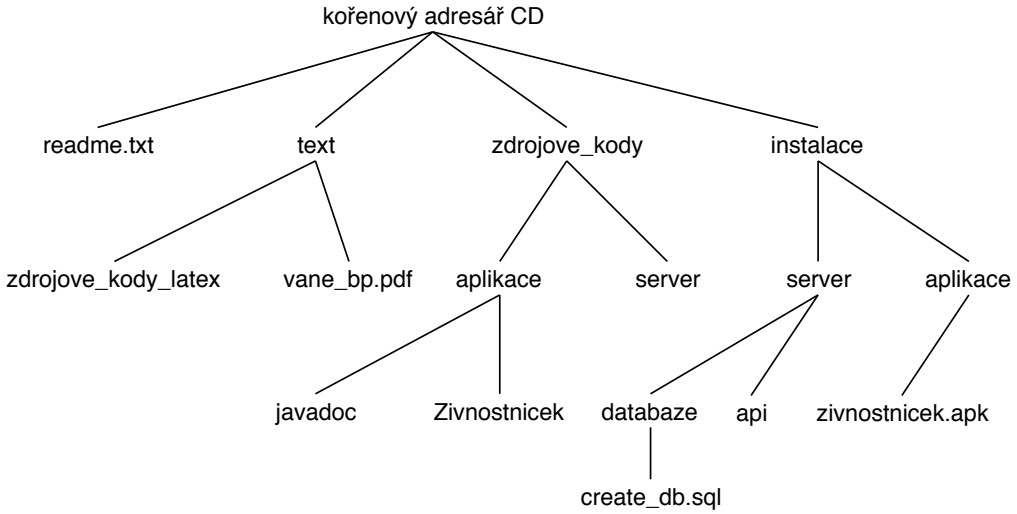
Obrázek B.15: Odhlášení uživatele z aplikace

C Obsah přiloženého CD

Popis adresářové struktury a obsahu přiloženého CD:

- `readme.txt` – Textový soubor popisující obsah cd a adresářovou strukturu.
- `text` – Složka obsahující text bakalářské práce a zdrojové kódy textu v LaTeXu.
- `zdrojove_kody` – Složka obsahující dva adresáře. První adresář `aplikace` zahrnuje zdrojové kódy mobilní aplikace a vygenerovaný Javadoc. Druhý adresář s názvem `server` obsahuje zdrojové kódy serveru.
- `instalace` – Složka obsahující opět dva adresáře. První s názvem `aplikace`, kde je uložen instalační soubor `zivnostnicek.apk`. Ve druhém adresáři `server` jsou obsaženy soubory nezbytné pro fungování serveru.

Základní adresářová struktura je zobrazena na obrázku C.16.



Obrázek C.16: Adresářová struktura přiloženého CD