FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Aplikace pro správu dlužných částek pro Android
Student:	Lenka Molinová
Vedoucí:	Ing. Marek Kodr
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem bakalářské práce je vytvořit mobilní aplikaci pro operační systém Android, která pomůže uživatelům mít pod konkrolovou finance ze společných akcí. Aplikace bude přehlědně zobrazovat dlužné částky mezi subjekty a uchovávat historii transakcí.

Proveďte následující kroky:
1) Analyzujte konkurenci na trhu a potřeby uživatelů.
2) Na základě výsledků proveďte analýzu a definujte požadavky na aplikaci.
3) Implementujte aplikaci pro operační systém Android.
4) Otestujte aplikaci a implementujte analytics.
5) Zpřístupněte aplikaci uživatelům.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 19. listopadu 2021 v Praze.

Bakalářská práce

APLIKACE PRO SPRÁVU DLUŽNÝCH ČÁSTEK PRO ANDROID

Lenka Molinová

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Marek Kodr
10. května 2023

Odkaz na tuto práci: Molinová Lenka. *Aplikace pro správu dlužných částek pro Android.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chci poděkovat vedoucímu práce Ing. Marku Kodrovi za odborné vedení práce a cenné rady. Své rodině za mnoho trpělivosti a neutuchající podporu nejen při studiu. Přátelům za oporu a cenné lekce. Danny za morální oporu a mnoho rad do života.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 10. května 2023

Abstrakt

Tato bakalářská práce se zabývá analýzou, návrhem a implementací mobilní Android aplikace pro správu skupinových financí. Aplikace uživatelům umožnuje evidovat transakce ve skupinách, a na jejich základě zobrazuje dopočet dluhů a pohledávek v rámci skupiny. V rámci práce je provedena analýza potenciálních uživatelů a konkurence. Jsou definovány konkrétní uživatelské požadavky. Je vypracován technický návrh, na jehož základě je implementována aplikace využívající služeb Firebase. Aplikace je otestována. Je diskutována budoucnost aplikace.

Klíčová slova mobilní aplikace, Android, Kotlin, finanční aplikace, aplikace pro skupinové finance

Abstract

This bachelor's thesis deals with the analysis, design and implementation of an Android mobile application for shared finances management. The application lets users record transactions for groups, and displays the computed state of claims and debts within the group. An analysis of potential users and competition is conducted. Specific user requirements are determined. Technical design is developed, based on which the application using Firebase services is implemented. Testing of the application is conducted. The application's future is discussed.

Keywords mobile application, Android, Kotlin, financial application, application for shared finances

Úvod

Jednou z oblastí života, jejíž zvládání mohou usnadnit mobilní zařízení, jsou osobní finance. Například je možné, místo dohledávání ztrácejících se papírků s poznámkami o dlužných částkách, ukládat takovéto informace online a přes mobilní telefon k nim přistupovat téměř odkudkoli. Zajímavým úkolem je potom vedení takovéto evidence zpřístupněné více lidem, přehledně a s automatickými dopočty.

Tato práce má za cíl vytvořit mobilní aplikaci pro systém Android (nejvíce používaný mobilní operační systém v ČR [1] i ve světě [2]), která umožní uživatelům napříč demografickými skupinami si snadno držet přehled v dlužných částkách. Forma mobilní aplikace splňuje přenositelnost, zároveň umožní (na rozdíl od webové stránky) zachovat alespoň část funkcionalit (nahlížení na již uložená data) i bez přístupu k internetu.

Nepořádek v osobních financích mívá nepříjemné důsledky, a to nejen finanční: může i vážně poškodit mezilidské vztahy. Užitek z práce mohou tedy mít všichni uživatelé mobilních zařízení s operačním systémem Android, kteří doposud nenašli vyhovující způsob udržování pořádku ve skupinových financích.

Motivací výběru tématu je právě poskytnutí možného řešení tohoto problému pro běžné uživatele. Ač jsou na trhu jistá řešení dostupná, stále není saturovaný.

Cíl práce

Hlavním cílem práce je vytvořit mobilní aplikaci pro systém Android, která umožní uživatelům mít pod kontrolou dluhy vzniklé z vícečetných společných nákupů.

Nejdříve bude uskutečněna analýza (1) aktuálně dostupných řešení na trhu a potřeb uživatelů. Budou identifikovány silné i slabé stránky konkurenčních řešení. Od potenciálních uživatelů bude zjištěno, co používají k propočtu dluhů, a co jim na těchto postupech vyhovuje a nevyhovuje.

Na základě této analýzy budou identifikovány konkrétní uživatelské požadavky (1.3) na aplikaci. Ta bude navržena (2) a implementována (3) tak, aby je splňovala a byla do budoucna rozšiřitelná o další užitečné funkcionality.

Aplikace bude umět pracovat s transakcemi mezi kombinacemi uživatelů a ne-uživatelů. Umožní transakce snadno zadávat, bude přehledně zobrazovat historii transakcí i výsledný stav dluhů.

Po implementaci bude aplikace zpřístupněna a její uživatelská přívětivost bude otestována (4.2).

Kapitola 1
Analýza

Správně provedená analýza dodá celému projektu ukotvení a vytvoří pevný podklad pro návrh.

Je potřeba zjistit, kdo jsou potenciální uživatelé vznikající aplikace, jaké by měli na aplikaci nároky a zda mají nějaké atypické potřeby. Také je na místě zjistit stav konkurenčních řešení na trhu, inspirovat se jejich silnými stránkami a zjistit jejich nedostatky.

Následovat bude analýza požadavků, ve které budou konkrétně vydefinovány klíčové vlastnosti a funkcionality vznikající aplikace. První verze aplikace patrně nebude mít všechny myslitelné funkcionality – cílem není kvantita, ale kvalitní a uživatelsky přívětivá implementace podstatných funkcionalit.

1.1 Potenciální uživatelé

Skupinové finance (bereme-li dva lidi jako skupinu) řeší v nějaké míře téměř každý člověk, potřeba sdílení či rozpočítávání nákladů se objevuje při: soužití v domácnosti; objednávání jídla či pití pro více lidí; zajišťování ubytování a jízdenek pro společné cestování; nákupu lístků na sousedící sedadla v kině; skládání dvou objednávek na e-shopu do jedné za účelem ušetření na dopravě; atp. Demograficky jsou tedy potenciální uživatelé aplikace pestrou skupinou, od celkové populace je patrně odlišuje jen absence malých dětí, lidí, kterým se z osobních důvodů příčí idea udržování přehledu o financích, a ne-uživatelů mobilních telefonů s Android OS.

Analýza požadavků potenciálních uživatelů začala jako rozhovory s pár z nich[1].

Dva zástupci uživatelů obdobných mobilních aplikací (1.2.3 Settle Up a 1.2.1 Splitwise) odpověděli, že jejich aplikace je „lepší než nic“, a často je, díky možnosti pohodlně zadávat výdaje z mobilního telefonu, pohodlnější/praktičtější než alternativní způsoby sledování a výpočtu vzájemných dluhů (např. sdílená tabulka s výpočetními formulemi, do které by členové skupiny přepisovali výdaje – není pohodlné pracovat s většími tabulkami na mobilním telefonu, pro tabulkově nezdatné členy skupiny to typicky není pohodlné řešení ani při dostupnosti osobního počítače). Ovšem oba měli výtky ke své aplikaci: u obou poukazovali na možná zlepšení celkového pohodlí užívání, u Splitwise navíc bral uživatel jako velmi velkou nevýhodu komplikovanost přidávání členů do skupiny.

Tři ne-uživatelé obdobných aplikací se nezávisle na sobě shodli, že by takové aplikaci neměli problém dát šanci, kdyby její rozhraní bylo pohodlné, snadno srozumitelné a ovladatelné. Neměli bohužel ani mnoho konkrétních návrhů či požadavků.

[1] Šlo o pět známých autorky, kteří prošli dvěma výběrovými kritérii: užívají chytrý mobilní telefon a na dotaz, zda by k řešení doménové problematiky byly potenciálně ochotní používat mobilní aplikaci, odpověděli „ano“.

1.1.1 Dotazník

Pro získání objektivnějších podkladů byl vytvořen anonymní dotazník.

Respondentů bylo 22. Převážně se označili jako věková kategorie 18–26 let (82 %). Jako svou aktuální primární metodu udržování přehledu o skupinových financích označilo mobilní aplikace 6 (27 %) respondentů. Zajímavostí může být, že stejný počet označil metodu „Pamatuji si to“. Zbytek označil jiné způsoby. Ač tedy odpovídala převážně generace velmi zvyklá užívání mobilních telefonů, 63 % patrně neobjevilo dostatečně dobrou mobilní aplikaci pro správu skupinových financí.

Záměrem dotazníku bylo zjistit, které funkcionality by potenciální uživatelé považovali za podstatné. Tato informace měla přispět k identifikaci klíčových funkcionalit, které by bylo vhodné implementovat již v první fázi aplikace. Za tímto účelem měli uživatelé ohodnotit jednotlivé funkcionality na této škále:

1. Vůbec bych nevyužil(a).
2. Využil(a) bych málokdy, tato funkcionalita pro mě není podstatná.
3. Využil(a) bych častěji (ale nejde o *nezbytně nutnou* funkcionalitu).
4. Naprosto nezbytné.

Ohodnocení každé z funkcionalit na této škále bylo povinné pro odevzdání dotazníku.

Odpovědi jsou zachyceny v tabulce 1.1. Tučně modře jsou mezi čísly zvýrazněny mediány pro danou funkcionalitu (kde medián vyšel mezi hodnoty škály, jsou zvýrazněné obě sousedící hodnoty).

Jak si čtenář může snadno všimnout, z 13 prezentovaných funkcionalit spadá medián do možnosti „Využil(a) bych častěji (ale nejde o *nezbytně nutnou* funkcionalitu).“ u 8 funkcionalit a u dalších 2 vychází mezi tuto a sousedící možnost. Toto je tedy první neočekávaná překážka, přes kterou se tato práce bude muset dostat: dotazník sice nastiňuje, na kterých funkcionalitách by mohlo uživatelům záležet, ovšem ne všechny (relativně) populární funkcionality bude možné v rámci tohoto projektu realizovat z časových důvodů.

Fakt, že nejčastější medián hodnocení byl „nejde o nezbytně nutnou funkcionalitu“ a ne „naprosto nezbytné“, však lze interpretovat i lehce optimisticky: nemálo potenciálních uživatelů by mohlo být ochotno aplikaci používat, i když ihned nedodá všechny tyto funkcionality.

Respondenti se relativně shodli na dvou nezbytných funkcionalitách: možnost specifikování podílu na výdaji konkrétními částkami a možnost vedení evidence i pro ne-uživatele aplikace. Tyto by nepochybně měly být zahrnuty mezi požadavky na vznikající aplikaci.

Tabulka 1.1 Očekávaná míra využití potenciálních funkcionalit respondenty. Čísla v buňce ukazují, kolik respondentů vybralo danou možnost, zvýrazněn je medián.

Funkcionalita	Vůbec nevyužije	Využije málokdy	Využije častěji	Naprosto nezbytné
Možnost dělení částky výdaje procenty	2	7	**8**	5
Možnost dělení částky výdaje podílem (např. „3:2“)	4	**7**	**6**	5
Možnost určit konkrétní částky, kterými mají účastníci podíl na výdaji	3	0	2	**17**
Možnost kombinovat dělení konkrétní částkou a procenty/podílem	4	5	**9**	4
Měnové konverze	5	**8**	1	8
Možnost přiřazovat výdajům kategorie (např. jídlo, sport, ...)	6	4	**11**	1
Možnost uložení fotek účtenek k nákupu	3	7	**9**	3
Generování QR kódu pro platbu k vyrovnání dluhů	4	5	**7**	6
Možnost evidovat transakce i pro lidi, kteří aplikaci nepoužívají	3	1	5	**13**
Možnosti sdílet aktuální stav účetnictví s ostatními	3	3	**8**	8
Možnost nechat ostatní přidávat záznamy o transakcích	4	4	**5**	9
Možnost mít různá práva pro různé členy skupiny (např. jen administrátoři mohou mazat transakce)	2	4	**5**	**11**
Možnost sporovat transakce, které se mě týkají	3	6	**7**	6

1.2 Konkurence v Google Play

Existuje mnoho softwarových řešení pro zaznamenávání dlužných částek. Analyzovány budou jen aplikace splňující tato kritéria:

- Jsou dostupné pro Android, konkrétně přes Google Play.
- Jsou orientovány na management spíše drobnějších dlužných částek mezi menšími skupinami; modelovou situací může být např. 8 kolegů dělících se o objednávky pracovních obědů, nikoli skupina 100 lidí shánějící pokročilejší účetní software.
- Z aplikací splňujících předchozí 2 body budou vybráni populární aktivní zástupci. Po revizi situace byly parametry pro poslední bod nastaveny takto:
  - alespoň 1 milion stažení,
  - hodnocení alespoň 4 hvězdy z 5, a
  - aktualizace aplikace v prvním pololetí roku 2022.

Aplikace zde budou seřazeny podle počtu stažení, od nejvyššího. Hodnoceny budou pouze možnosti jejich využití pro správu dlužných částek; jiné funkcionality nebudou brány v potaz.

1.2.1 Splitwise

Nejpoužívanější konkurent má 10 milionů stažení [3]. Jde o freemium[2] aplikaci. Služba je přístupná i přes web.

Na snímcích obrazovky 1.1 vidíme náhled skupiny a užitečnou možnost nerovného rozdělení výdaje pro situaci, kdy jeden či více účastníků mají zaplatit specifické částky navíc, a zbytek výdaje má být rozdělen rovným dílem. Nejen tento způsob dělení je zároveň uživateli vysvětlen i s ilustrací. V nastavení skupiny lze zvolit, zda se má minimalizovat počet převodů pro vypočítávané vyrovnání dluhů.

Rozhraní je většinou celkem přehledné (ve světlém i tmavém režimu) a snadno použitelné. Výjimkou je lišta s tlačítky pro akce ve skupině. Na testovacím zařízení se lišta nevejde ani na dvojnásobek šířky obrazovky 1.2. Vývojový tým by mohl, chce-li zachovat obdobný design, lištu alespoň udělat dvouřádkovou, aby se uživatelé mohli ke zpřístupňovaným funkcionalitám dostat rychleji a pohodlněji. Další problém je u výběru způsobů nerovného dělení – chtěnou možnost je třeba vybrat ze záložek nahoře na obrazovce buďto kliknutím (což je na dnešních velkých telefonech nepohodlné) nebo se gestem vodorovného posunutí postupně přesouvat mezi záložkami (což je při pěti záložkách zdlouhavé).

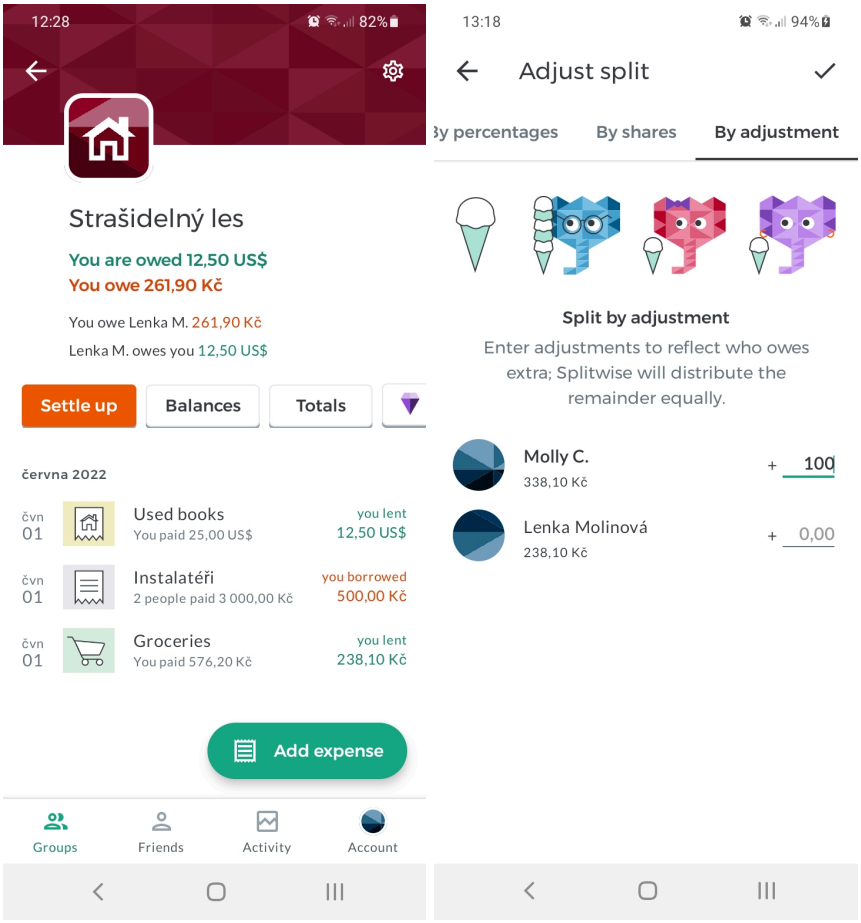
Výdaje skupiny lze exportovat do CSV.

Zajímavostí u Splitwise může být, že skupiny nemají nic jako „defaultní měnu“. Když uživatel přidává výdaj, je předvybrána jím naposledy použitá měna, kterou může samozřejmě změnit. Případně si uživatel může definovat vlastní defaultní měnu.

1.2.1.1 Výhody

- Snadné použití. Lze se přihlásit přes Google účet nebo kombinací e-mailu a hesla. Rozhraní je přehledné, uživatel snadno najde základní operace. Přidat kontakty lze vyplněním jejich e-mailu, telefonního čísla, či naskenováním QR kódu z jedné aplikace do druhé.
- Přepočítává dluhy se stejným člověkem napříč skupinami.
- Podporuje u nákupů více plátců.

[2] Část funkcionalit aplikace je zdarma, část vyžaduje zakoupení pokročilejší licence.



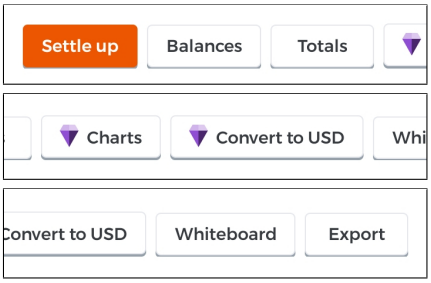
Obrázek 1.1 Splitwise: náhled skupiny a dělení nákupu.

- Mnoho způsobů definice podílů účastníků na transakci: rovný podíl, *podíl ze zbytku*[3], procenta, konkrétní částky a varianta *s korekcemi*[4].
- K výdajům lze přidávat komentáře.

1.2.1.2 Nevýhody

- Do skupiny nelze přidat účastníka bez kontaktu (telefonní číslo či e-mail)[5]. K tomuto kontaktu se jim následně vytváří Splitwise účet, pokud ho již nemají. Splitwise alespoň umožňuje účty sloučit [4], pokud má vlastník přístup k oběma účtům i e-mailům. Přesto ale z potřeby zadání kontaktu k účastníkovi plyne několik problémů:
  - Mnoho lidí nemá jen jeden e-mail. Tedy i v příznivé situaci, kdy uživatel zakládá skupinu na Splitwise a zná e-maily všech členů, má možnost buďto od všech členů zjistit[6], jestli již nepoužívají pro Splitwise jiný e-mail, nebo očekávat, že pro několik stávajících uživatelů vygeneruje nový účet, který si oni následně budou muset slučovat se stávajícím.

[3] Rozdílná *váha* podílu účastníků, např. „1+n“, kde „n“ je počet dalších lidí, za které daný účastník hradí.
[4] K jednotlivým účastníkům je možné napsat částku, kterou z výdaje platí jen oni. Zbytek výdaje je mezi účastníky rozdělen rovným dílem, a dříve zmíněným jsou přičteny právě specifikované částky.
[5] Je sice možné sdílet odkaz pro připojení do skupiny, účastník by si však musel následně zakládat účet.
[6] Což, pokud nejsou daní lidé v dané chvíli na stejném místě, znamená typicky čekání na odpovědi.



Obrázek 1.2 Splitwise: příliš dlouhá lišta s akcemi ve skupině (snímky procházení).

- Uživatel zakládající skupinu v aplikaci ale nemusí na všechny členy mít e-mail/telefon. V takovém případě pravděpodobně bude chtít tyto údaje od daných členů získat. Dokud je nezíská, nemá možnost do aplikace korektně zadat jakékoli transakce, na nichž měli tito členové podíl – Splitwise ho tedy nutí evidovat do té doby transakce i jinde, než ve Splitwise.
- Pokud skupina lidí záměrně nechce, aby někteří členové neměli ke skupině přístup a přesto jim byly evidovány transakce, mají jedinou možnost: zakládat nové e-maily. To může být (nepříliš pohodlnou) cestou pro skupiny s pár nekooperativními členy. Ale pro situace, kdy by větší skupinu mělo spravovat jen několik málo členů[7], je stav velmi nešťastný.

- Převod měn je součástí pouze placené verze (ač jde o výpočet, který by mohl být proveden přímo na telefonu uživatele, kde by nezatěžoval servery provozovatele).
- Části uživatelského rozhraní by mohly být snadněji ovladatelné.
- Nemá českou lokalizaci.

1.2.2 Tricount

Jedná se o freemium aplikaci dostupnou zde [5]. Služba je přístupná i přes web.

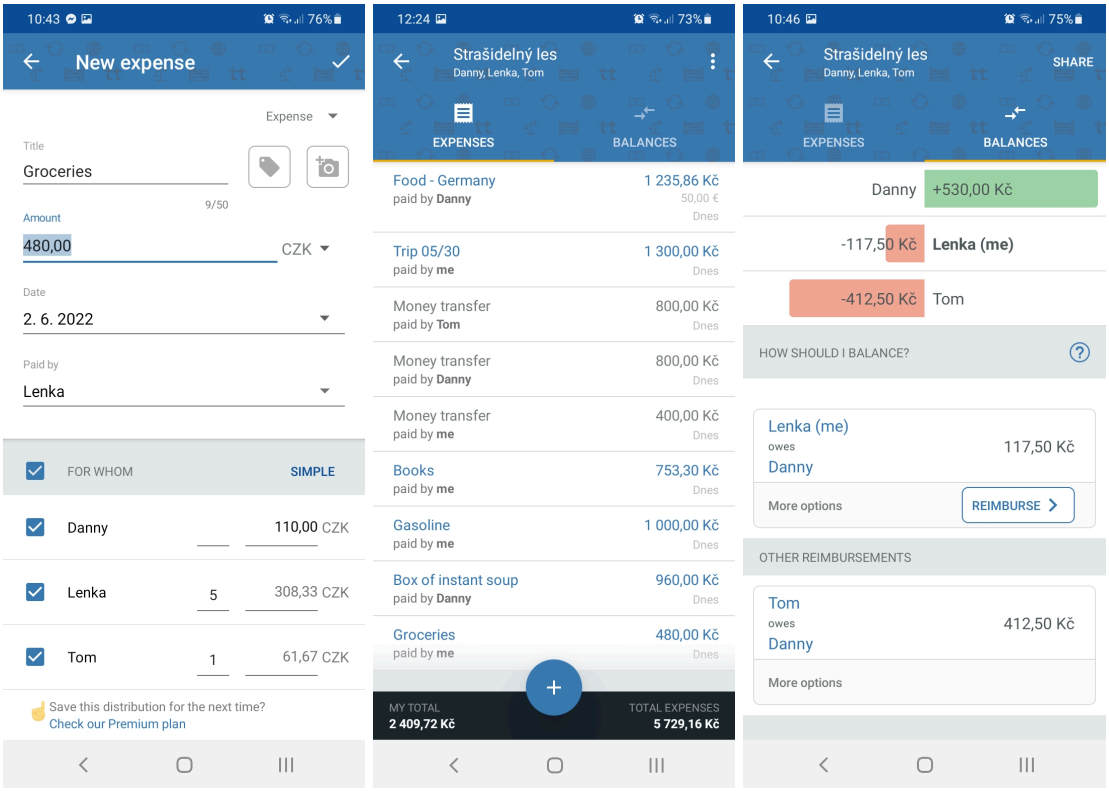
Tato aplikace pojímá problematiku značně odlišně. Lze ji používati i bez účtu, uživatel si vytvoří skupinky, ve kterých vytvoří účastníky – aniž by měl možnost je nějak napojit např. na jejich existující Tricount účet. Uživatel vybere, za kterého z účastníků vystupuje (lze měnit), a zadává výdaje a převody peněz mezi účastníky.

Více plátců u výdaje není podporováno. Dělit výdaje je možné rovnoměrně, poměrově, konkrétními částkami, nebo kombinací konkrétních částek pro část uživatelů a poměrů pro jiné (uživatelé s částkami mají podíly právě danými částkami, zbytek celkové ceny je rozdělen dle poměrů mezi ostatní). Je podporováno více měn: skupina má základní měnu, při přidání výdaje v jiné je automaticky přidán aktuální kurz pro přepočet, který lze upravit.

Uživatelské rozhraní je minimalističtější (obrázek 1.3), ve světlém i tmavém režimu přehledné. S pohodlností ovládání je to horší – opět zde máme mnoho interaktivních prvků nahoře na obrazovce. Taktéž není k dispozici překlad do češtiny.

Samotný uživatelský účet (pokud si ho uživatel vytvoří) je standardně zabezpečen přístupem přes účet Google nebo kombinací e-mailu a hesla. U této služby je ale problematický způsob sdílení skupin, resp. absence jakéhokoli zabezpečení sdílené skupiny. Skupinu lze sdílet (jen a pouze) pomocí vygenerované URL, kterou lze otevřít jak v aplikaci, tak na webu. Kdokoli takto URL navštíví (aniž by se musel přihlašovat do Tricount účtu) si následně vybere, za kterého z účastníků

[7] Tato práce sice řeší jen nekomerční používaní takovýchto aplikací, ale i tam taková situace občas nastává. Skupina lidí na akci/výletě/dovolené může určit jen velmi malou pod-skupinu členů jako „účetní“, např. výměnou za nižší účast na jiných pracích. Daní členové mohou být určeni na základě důvěryhodnosti, skupina nemusí chtít, aby všichni členové měli možnost editovat účetnictví. Daní členové by měli mít možnost využít pro usnadnění svých povinností aplikaci.



Obrázek 1.3 Tricount: vytváření transakce s různými podíly, náhledy výdajů a bilancí skupiny.

vystupuje, a může následně projít veškeré transakce ve skupině a jakkoli je upravovat: měnit existující, mazat, přidávat nové... má plný přístup. Při testování nebyla nalezena možnost toto chování jakkoli upravit, ani např. pro případ zjištěného úniku možnost změny URL či zablokování editace ne-vlastníkem.

Aplikace pro skupinovou správu dluhů vždy vyžadují jistou míru důvěry v uživatele s přístupem ke skupině. Ale měla by být umožněna kontrola toho, kdo smí do skupinových výdajů nahlížet, a hlavně kdo je smí editovat. Toto Tricount nesplňuje.

1.2.2.1 Shrnutí

Tricount lze doporučit v situacích, kdy uživateli nebudou překážet jeho nevýhody:

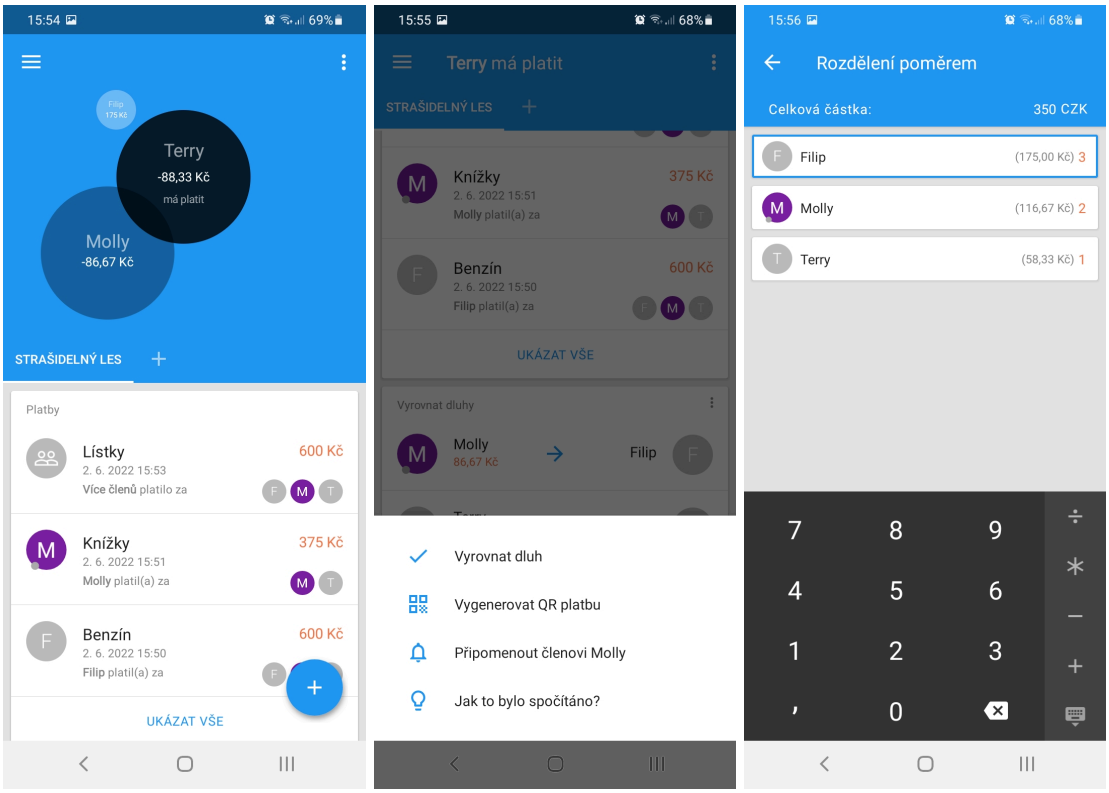
- Nemožnost zákazu úprav přes URL skupiny. To by nemusel být problém v některých situacích:
  - Když má vedení skupinové evidence na starosti jediná dedikovaná osoba, která nemá a nebude evidenci s nikým napřímo sdílet.
  - Když si účastníci plně důvěřují, jak ve svých úmyslech, tak ve svých schopnostech omylem nesdílet URL na špatných místech.
  - Když jde o krátkodobou skupinovou akci, s menším počtem účastníků a v součtu drobnými částkami, kde nikoho netrápí možnost úniku či zničení evidence.
- Chybějící podpora vícero plátců u transakce. To by nemusel být problém, pokud skupina nebude:

- Realizovat mnoho transakcí v hotovosti. Zvláště při cestování po zemi s jinou měnou, než je domácí měna členů, pokud skupina plánuje často platit hotově, hrozí, že se z důvodu nedostatku hotovosti ve správné měně budou muset skládat.
- Realizovat transakce tak velké, že hrozí, že se budou muset skládat kvůli aktuálním zůstatkům na účtu. Zde by však alespoň nemusel být takový problém zaevidovat přesuny jako oddělené transakce.

1.2.3 Settle Up

Opět freemium aplikace (dostupná z [6]) s možností připojit se z webu. Z předchozích jsou její funkcionality podobnější Splitwise.

Pro přístup je nutný účet či přihlášení přes Google, Apple či Facebook účet.



Obrázek 1.4 Settle Up: náhled skupiny, možné akce u dluhů, dělení poměrem.

Také se soustředí na skupiny. K vazbě mezi účastníkem a uživatelem je zde zatím nejpružnější přístup: v transakcích vystupují účastníci skupiny, a uživatelé si u libovolného počtu z nich mohou zaškrtnout, že chtějí vystupovat „jako tento účastník“, což jim upraví barevné zvýrazňování „vlastních“ transakcí, ale pro návrhy vyrovnávání dluhů se pořád počítají jako oddělení účastníci.

Další uživatele je možné pozvat mj. přes odkaz. Možnost připojit se přes daný odkaz je sice defaultně aktivní, ale uživatel při generování odkazu rovnou vidí přepínač pro pozastavení této možnosti. Přes tento odkaz je zároveň možné se podívat na stav transakcí a dluhů ve skupině (včetně záznamů, kdo co kdy přidal). Jsou zde ovšem 2 podstatná omezení. Zaprvé bez přihlášení nelze zobrazit detaily výdajů, a tudíž nelze zobrazit poměry dělení daných výdajů. Kdyby chtěl účastník bez účtu ověřit, zda nebyly podíly upraveny v jeho neprospěch, místo pouhé kontroly podílů u výdajů by musel provést celkový přepočet. Zadruhé, odkaz pro zobrazení bez účtu je

shodný s odkazem pro připojení, a když je možnost připojení deaktivována, nefunguje odkaz ani pro zobrazování.

Je podporováno mnoho měn a převody mezi nimi.

Je podporováno více plátců.

Dělení mezi účastníky je možné rovnoměrně, podíly nebo konkrétními částkami. U podílů i částek je možné při úpravě využít funkce kalkulačky, ale použitý výraz je po interakci s jakýmkoli jiným prvkem vyhodnocen a není možné ho znovu zobrazit, uživatel kontrolu správnosti zadaného výrazu tedy nemůže provést zpětně.

Stejně jako ve Splitwise je možné zvolit, zda bude počet převodů pro vyrovnání dluhů minimalizován.

K účastníkovi lze připojit číslo účtu pro snadnější vyrovnávání dluhů.

Jako jediná ze srovnávaných aplikací má český překlad.

Seznam transakcí s detaily je možné exportovat ve formátech CSV a TXT.

Settle Up má propracované rozhraní (obrázek 1.4). Je přehledné ve světlé i tmavé verzi. Zároveň jsou všechny často využívané interaktivní prvky pěkně na dosah. V přehledu skupiny plní barevné „bubliny“ obdobnou funkci, jako u Tricount zeleno-červené grafy. Bublinky jsou hezčí na pohled, ovšem za cenu snížené čitelnosti, neboť mohou být i velmi malé. Přehlednosti pomáhá i přidání avatarů členů s podílem na výdaji do pravého dolního rohu karty výdaje. Je použita buďto iniciála (v tom případě jsou barevně zvýrazněny výskyty účastníka propojeného s aktuálním uživatelem; bohužel jde jen o jedno písmeno, snadno tedy nastane kolize) nebo fotka daného účastníka, byla-li přidána (zde naopak přicházíme o barevné zvýraznění uživatele).

1.2.3.1 Shrnutí

Jedná se o aplikaci s propracovaným rozhraním. Má oproti Splitwise i Tricount menší možnosti specifikování podílů účastníků na výdajích. I na jiných místech občas působí, že estetická stránka dostala přednost před praktičností: velkou část detailu skupiny zabere směs těžko čitelných bublin, při nahrání profilových obrázků ztrácí uživatel vizuální zvýraznění svých transakcí. I přes to je aplikace stále relativně pohodlná na používání.

Má zatím nejvíce flexibilní přístup k vazbě uživatel–účastník.

Jako jediná z analyzované trojice nabízí českou lokalizaci.

1.2.4 Závěr

U konkurence se vyskytuje mnoho propracovaných i nepohodlných schémat.

Nejširší výběr dělení podílu na výdajích má Splitwise. Nejvíce flexibilní přístup k vazbě mezi uživatelem aplikace a účastníkem výdajové skupiny má Settle Up.

Vytvářená aplikace má šanci konkurovat ostatním těmito způsoby:

- Splitwise: Velkou výhodou by byla možnost snadného vedení evidence i pro ne-uživatele aplikace. Také možnost snadno napojit nového uživatele aplikace na již existujícího účastníka skupiny by byla značnou výhodou. Uživatelé by také mohli ocenit možnost bezplatných převodů měn.
- Tricount: Možnost vytvářet transakce s více plátci. Také možnost zabránění úpravám dat skupiny zlomyslnými uživateli – i kdyby v začátku jen pomocí kombinace vyžadování uživatelského účtu pro editaci a možnosti nahlásit zlomyslné uživatele na podporu pro zablokování jejich účtu – by byla výhodou.
- Settle Up: Přehlednější, čistější a pohodlnější grafické rozhraní, ve kterém uživatel nebude přehlcen a bude se snadněji a rychleji intuitivně orientovat.[8]

[8] Tuto potenciální výhodu bude patrně nejtěžší realizovat, a v rámci této práce se to nemusí plně povést. Avšak s uživatelským testováním na konci práce, zapracováním jeho výsledků a následném opakování obdobných cyklů, by mělo být možné tohoto cíle dosáhnout.

Ve všech třech aplikacích také chybí možnost sdílení dat skupiny a dlužných stavů v rámci ní jen pro čtení. Dodání této funkcionality patrně nebude z časově-kapacitních důvodů součástí první verze aplikace, která vznikne v rámci této práce. Do budoucna by ale byla podstatnou konkurenční výhodou, neboť ostatní aplikace takovou funkcionalitu nenabízejí, a přitom v reálném světě uživatelé budou narážet na situace, kdy sice nechtějí někomu povolit úpravy dat o transakcích, ale chtějí mu dát možnost do nich nahlédnout. Aplikace bude navržena tak, aby bylo možné tuto funkcionalitu v budoucnu dodat.

1.3 Analýza požadavků

Dotazník vytvořený pro analýzu potenciálních uživatelů (1.1.1) byl záměrně stručný, aby bylo docíleno co největšího počtu vyplnění. Pro analýzu požadavků by, ani při jiných výsledcích, sám o sobě zdaleka nemohl poskytnout dostatek informací. Z dotazníku však alespoň máme informaci o dvou požadavcích, které respondenti označili za nezbytné, a tudíž by tyto požadavky měly být zanalyzovány a zahrnuty do seznamu: možnost specifikování podílu na výdaji konkrétními částkami a možnost vedení evidence i pro ne-uživatele aplikace.

Většina podkladů pochází z rozhovorů s několika potenciálními uživateli. Mezi těmito byli pravidelní uživatelé konkurenční aplikace (konkrétně 1.2.3 Settle Up), osoby nevyužívající žádnou obdobnou aplikaci, i osoby využívající obdobnou aplikaci příležitostně.

Vývoj plné verze takovéto aplikace může být velmi časově náročný. Velmi dlouhý vývoj bez průběžného uživatelského testování (se zapracováním poznatků do zadání pro vývoj) by také mohl vést k aplikaci, která není uživatelsky přívětivá. Cílem této práce je vytvořit funkční prototyp, který bude pokrývat klíčové funkcionality a který bude možné uživatelsky otestovat. Následující seznamy (hlavně funkční požadavky) tedy nezahrnují funkcionality, které by pro ostrý běh aplikace sice byly potřebné, ale nebývají využívány příliš často, nejsou specifické pro tento typ aplikace, a při uživatelském testování by pravděpodobně nepřinesly informace o uživatelské použitelnosti navržené aplikace. Proto v následujícím seznamu chybí např. možnosti změny a resetu hesla, ověření a změny e-mailu, smazání účtu.

1.3.1 Funkční požadavky

Aplikace bude umožňovat zaznamenávat společné nákupy účastníků a platby mezi účastníky. Data bude primárně ukládat na úložišti dostupném přes internet.

1.3.1.1 Registrace uživatele [FP1]

Uživatel si bude moci vytvořit účet vázaný ke svému e-mailu. Účet zabezpečí heslem.

1.3.1.2 Přihlášení uživatele [FP2]

Uživatel se pro používání aplikace bude muset přihlásit k účtu. Bez přihlášení nebude mít přístup k údajům v aplikaci.

1.3.1.3 Skupiny [FP3]

Pro udržení přehlednosti evidence transakcí budou uživatelé moci vytvářet skupiny a k existujícím skupinám umožňovat připojení dalším uživatelům. Jeden uživatel může být v libovolně mnoho skupinách.

Uživatel v aplikaci uvidí seznam skupin, k nimž je připojen, a bude se moci podívat na jejich detail.

Skupiny budou mít název a měnu. Připojení uživatelé budou moci obojí upravovat.

Uživatel může skupinu, ke které je připojen, opustit (i pokud skupinu sám vytvořil).

1.3.1.4 Lokální členové skupin [FP4]

Aby uživatelé mohli snadno zaznamenávat transakce i pro skupiny, jejichž všichni členové si nezařídili účet v aplikaci a přístup ke skupině, budou se transakce vázat ne přímo na uživatelský účet, ale na *lokální členy* skupiny. Ve skupině je tedy seznam zástupných entit – *lokálních členů* – mezi které uživatelé transakce rozdělují. Vztah reálný uživatel–lokální člen může být 1:1, ale i 0:1; nic nebrání vytvoření zástupného *lokálního člena* pro ne-uživatele nebo pro nekooperativního uživatele, kterého zbytek nechce do skupiny přizvat. Může být i N:1 – mohou vzniknout i skupiny, kde lokální členové nezastupují individuální osoby, ale třeba celé rodiny, domácnosti, páry či jiné podskupiny, které se např. rozhodly za výdaje z konkrétní akce, pro kterou skupina vznikla, platit dohromady. Varianta 1:N v rámci jedné skupiny nebude podporována (uživatel tedy ve skupině nemůže vystupovat jako více *lokálních členů* zároveň).

Uživatelé budou moci do skupiny přidávat *lokální členy*.

Uživatelé připojení ke skupině si budou moci z těchto *lokálních členů* jednoho vybrat, aby měli zvýrazněné výskyty daného člena v transakcích.

Od tohoto bodu v textu budou zástupné entity členů ve skupině označovány právě jako *lokální členové* dané skupiny. Uživatelé aplikace s přístupem ke skupině budou označování jako *připojení uživatelé* (*uživatelé připojení* k dané skupině). Tam, kde budou použita samotná spojení „člen/účastník skupiny“, buďto dvojznačnost nebude na škodu, nebo bude[9] z kontextu jasné, zda jde o *lokální členy* či *připojení uživatele*.

1.3.1.5 Přidávání transakcí [FP5]

Člen skupiny může do skupiny přidat nový záznam transakce mezi libovolnými členy skupiny. Položky, které u transakce vyplní, jsou:

- Cena: celková číselná částka s měnou.
- Platící účastníci: *lokální členové*; s částkami, které zaplatili; musí být alespoň jeden.
- Účastníci, za které (či kterým) bylo placeno, spolu s jejich podíly. Tito účastníci mohou být i mezi platícími. Musí být vyplněn alespoň jeden. Podporované dělení bude konkrétními částkami a podílovými koeficienty[10].
- Datum transakce (bude předvyplněno na aktuální datum).
- Volitelně název (prostý text).
- Volitelně poznámka (prostý text).

Podporované měny budou: česká koruna, euro a americký dolar.[11]

1.3.1.6 Zobrazení transakcí [FP6]

Uživatel bude mít možnost prohlédnout si seznam již zaznamenaných transakcí v rámci jedné skupiny, řazený od transakcí s nejnovějším datem uskutečnění po nejstarší. Seznam může obsahovat jen stručnější informace o transakcích. Plné informace si uživatel bude moci zobrazit proklikem na detail vybrané transakce, kde uvidí všechny vyplněné položky (viz výše 1.3.1.5 FP5).

[9] Doufejme!

[10] Číselné určení poměrů mezi podíly, „vah“. Tedy chceme-li částku v hypotetické měně 100 M rozdělit mezi A a B 2:3, vyplníme koeficienty „2“ a „3“, a ve výsledku bude mít A podíl 40 M a B podíl 60 M.

[11] V dotazníku uživatelé vyjádřili, že možnost konverze měn pro ně není příliš podstatná. Zavedení více měn do evidence a výpočtu od počátku implementace by však mělo do budoucna ušetřit mnoho náročných úprav kódu a nekompatibilních úprav databázového modelu, a to za cenu jen malého zvýšení pracnosti v implementační fázi této práce. Podpora více světových měn navíc umožní zpřístupnit aplikaci i uživatelům mimo Českou republiku.

1.3.1.7 Upravování a mazání transakcí [FP7]

U veškerých transakcí, které si uživatel může zobrazit, může i provést libovolné úpravy (smí tedy měnit všechny informace zadávané při vytváření, viz 1.3.1.5 FP5).

Všechny transakce, které si uživatel může zobrazit, může i mazat (bude vyžadováno potvrzení akce, např. odkliknutím v dialogovém okně).

1.3.1.8 Zobrazení aktuálních dlužných stavů ve skupině [FP8]

Ve skupině si uživatel bude moci zobrazit dlužné stavy členů vůči skupině jakožto celku.[12]

Stavy budou zobrazeny v měně skupiny. Bude-li pro výpočet potřeba převodu měn, postačí prozatím použití pevných konstant (podle dotazníku není převod měn pro potenciální uživatele příliš podstatný; do budoucna však má architektura aplikace podporovat přechod na dynamičtější zdroj převodových konstant).

1.3.1.9 Zobrazení celkového vlastního dlužného stavu [FP9]

Uživatel si bude moci zobrazit součty dlužných stavů všech svých *lokálních členů* ve všech svých skupinách:

- součet dluhů pro všechny své členy, kteří jsou „v mínusu“,
- součet pohledávek pro všechny své členy, kteří jsou „v plusu“,
- a rozdíl těchto dvou čísel.

Tyto součty se zobrazí v uživatelově preferované měně, kterou může uživatel upravit. Pokud uživatel svou preferovanou měnu nezadal, nemusí se součty zobrazit. Bude-li pro výpočet potřeba převodu měn, opět postačí použití konstant.

1.3.2 Nefunkční požadavky

1.3.2.1 Zařízení a orientace

Aplikace bude určená pro zařízení s operačním systémem Android verze alespoň 5.0. Bude optimalizovaná pro vertikální zobrazení na mobilních telefonech s úhlopříčkou displeje alespoň 5 palců. Pro vertikálně orientovaná zařízení s úhlopříčkou přesahující 7 palců nebude optimalizovaná, ale bude fungovat. Nemusí být podporována ani zařízení s úhlopříčkou menší než 5 palců, ani horizontální režim.

1.3.2.2 Možnost čtení offline

Pro operace zahrnující úpravu dat či přihlašování k účtu bude internetové připojení samozřejmě vyžadováno. Pouhé čtení dat již přihlášeného uživatele by ale středně dlouhou (do 6 hodin) nedostupností internetového připojení nemělo být narušeno, pokud uživatel s aplikací před ztrátou připojení (méně než tři hodiny před ní) pracoval.

1.3.2.3 Stálost přihlášení

Uživatel se při znovu-otevření aplikace (po až jednotkách dní jejího nepoužívání) nebude muset znovu přihlašovat, pokud se předtím neodhlásil. Pokud se odhlásil, bude se samozřejmě pro používání aplikace muset nejdříve znovu přihlásit.

[12] Příklad: máme skupinu se členy A, B a C, evidujícími transakce v měně M. B si půjčí od A 10 M, B zaplatí za C 3 M, a obě transakce jsou zaevidovány. Konečný stav „vůči skupině“ bude, že B dluží 7 M, C dluží 3 M a A je má pohledávku 10 M.

1.3.2.4 Lokalizace uživatelského rozhraní

Textace prototypu aplikace bude v anglickém jazyce. Zůstanou tak otevřené dveře možnému dřívějšímu zpřístupnění následujících verzí mnohem více uživatelům po světě, což by značně zvedlo množství potenciálních uživatelských hodnocení, a tím i možnost uživatelské výtky zapracovávat.

Dodání překladu v budoucnu by však nemělo být příliš pracné, uživateli prezentované texty (mimo chybových hlášek) musí standardně být odděleny od zdrojového kódu.

1.3.2.5 Očekávaná velikost skupin

Očekávané využití aplikace je nekomerční, pro skupiny o řádově 2 až 15 účastnících. Se zvyšováním počtu účastníků přes onu horní hranici musí aplikace stále fungovat, ale uživatelské rozhraní se může stávat úměrně méně pohodlné na užívání (např. dlouhé procházení seznamem účastníků při vytváření/úpravě transakce).

1.3.2.6 Přehledné uživatelské rozhraní, přístupnost

V designu grafického rozhraní bude kladen důraz na přehlednost. Uživatel by se v grafickém rozhraní měl být schopen snadno orientovat. Měl by být schopen aplikaci používat intuitivně.

Uživatelské rozhraní musí respektovat volbu světlého či tmavého režimu nastavenou na zařízení v době používání aplikace. V obou barevných režimech musí být kontrast vizuálních prvků dostatečný pro pohodlné čtení textu a ovládání aplikace za světelných podmínek běžných pro volbu daného režimu (tedy za běžného až jasného světla pro světlý režim, za šera až temna pro tmavý režim).

Aplikace však nebude optimalizována pro funkce přístupnosti: v rámci bakalářské práce bohužel není možné stihnout důkladně vyladit a otestovat uživatelskou přívětivost aplikace při používání systémově zvětšeného textu, čteček displeje či ovládání hlasem.

1.3.2.7 Škálovatelnost backendového řešení

Backendové řešení musí být škálovatelné beze změny kódu či nutnosti vydávání nové verze aplikace. Rozšíření kapacit backendu by mělo být co nejméně technicky a časově náročné, mimo očekávané nutnosti navýšení plateb za služby by měl být pro rozšíření maximálně jeden pracovní den běžného pracovníka v oboru informačních technologií.

Kapitola 2
Návrh

2.1 Výběr technologií

2.1.1 Nativní nebo hybridní aplikace

Aplikace pro Android mohou být nativní nebo hybridní. Hybridní vývoj umožňuje aplikaci s jedním zdrojovým kódem spouštět na vícero operačních systémech[1]. Ač pro investory či společnosti s větším týmem developerů může být hybridní vývoj přitažlivý – je-li záměrem vyvíjet aplikaci se stejnými funkcionalitami pro vícero operačních systémů – má hybridní vývoj i několik nevýhod. Například bývají hybridní aplikace lehce pomalejší oproti nativním, neboť k potřebným nativním funkcionalitám musí přistupovat přes pluginy. Pokud je cílem mít jednu uživatelsky přívětivou aplikaci pro různé operační systémy, musí se buďto udělat kompromisy v designu (jelikož uživatelé různých systémů bývají zvyklí na jiné komponenty, schémata navigace apod.) nebo pracně udržovat vícero verzí aplikace; pokud podpora vícero mobilních operačních systémů potřeba není (což je případ vznikající aplikace), ztrácí hybridní vývoj svou hlavní výhodu.

Podstatnou nevýhodou vývoje hybridních aplikací je velmi velká pracnost jejich ladění. Pokoušet se o to v „týmu“ skládajícím se z jediného člověka by bylo časově riskantní a benefity hybridního vývoje tento risk nekompenzují. Navíc je při nativním vývoji možné snadno používat standardní Android komponenty, na které jsou uživatelé Androidu zvyklí. Stylovat komponenty v hybridním kódu tak, aby vypadaly podobně, by bylo značně časově náročné.

Byl tedy zvolen nativní vývoj.

2.1.2 Implementační jazyk

Nativní aplikace pro Android lze psát v jazycích Java a Kotlin. Oba využívají stejné JVM[2] a jsou značně kompatibilní - v Kotlinu lze využívat existující knihovny i jiný kód v jazyce Java.

Kotlin má oproti Javě několik výhod, pokud jde o „pohodlí“ a rychlost psaní kódu a hlavně pro následné udržování kódu. Hlavní výhodou je explicitní rozlišování *nullable*[3] typů. Za zmínku také stojí třídy `data class` – používají se pro uchovávání strukturalizovaných dat, standardně mají immutable atributy. Na základě hodnot těchto atributů jsou automaticky přetíženy metody `equals` a `hashCode`, což velmi usnadňuje nejen porovnávání individuálních objektů, ale i práci se

[1] Pro mobilní vývoj typicky Android a iOS. Hybridní vývoj typicky využívá směs webových komponent pro zobrazování a pluginů pro přístup k nativním funkcionalitám jako jsou např. kamery.

[2] *Java Virtual Machine*

[3] Typy, které smí mít hodnotu `null`.

`Sety` (množinami) daných objektů. Zároveň je na základě signatury konstruktoru vygenerovaná velmi nápomocná `copy` metoda.

2.1.3 Grafický framework

Pro vykreslování UI pro Android aplikace v Kotlinu se typicky využívá jedna ze dvou možností: starší Android View System a novější Jetpack Compose. Android View System je sice mocný nástroj, ale vyžaduje definici vzhledu (elementů, jejich vizuálních vlastností i jejich rozložení) obrazovek v XML souborech a definici jejich chování v korespondujícím souboru s kódem. Oproti tomu Jetpack Compose umožňuje definovat vzhled i chování UI přímo Kotlin kódem, tudíž je možné pro UI mít *single source of truth*[4]. Toto velmi usnadňuje vytváření i udržování aplikace, tudíž byl zvolen Compose.

2.1.4 Backend řešení

Firebase [7] je služba od společnosti Google, která zpřístupňuje backend služby pro vývoj mobilních a webových aplikací. Nabízí mimo jiné hosting, autentizaci uživatelů, NoSQL databázi, úložiště, distribuci aplikace, monitoring a infrastrukturu pro automatické testy. V omezené míře zpřístupňuje tyto služby i zdarma. Samozřejmě není nutné Firebase využívat jako jedinou backendovou službu, kombinování nic nebrání.

Níže jsou popsány vlastnosti jednotlivých Firebase služeb, které budou využity. Cenové podmínky využití těchto služeb pro produkční provoz jsou výhodné, a zvýšení objemu obsluhovaných uživatelů je pouze otázkou úměrného navýšení plateb [8, 9, 10]. Jednotlivé služby (např. databázi a autentizaci) není třeba mezi sebou pracně integrovat. Pro potřeby testovacího provozu pro vývoj aplikace v rámci této práce navíc budou plně postačovat bezplatné verze služeb.

2.1.4.1 Analytics

Firebase Crashlytics je velmi snadné zaintegrovat do projektu: stačí přidat do aplikace závislost na knihovně a v nastavení Firebase projektu Crashlytics zapnout [11].

Webové rozhraní, ve kterém lze následně sledovat informace o pádu aplikace, je přehledné a poskytuje mnoho informací pro usnadnění opravy chyb způsobujících pády. Crashlytics sdružují pády do skupin podle podobnosti ve výpisech zásobníku a chybových hlášek ve výjimkách [11]. Skupiny se řadí podle počtu pádů. Ve skupině si lze procházet jednotlivé výskyty chyb a zobrazit si detaily o chybě i o zařízení, na kterém nastala: je k dispozici výpis zásobníku, typ chyby, značka zařízení, verze OS, orientace zařízení v době pádu, atd.

Pro Firebase jsou také bezplatně dostupné Google Analytics [12]. Webové rozhraní nástroje přehledně zobrazuje, jak se počty uživatelů a množství jejich aktivity vyvíjí v čase, z jakých jsou zemí, jakou verzi aplikace používají. Pro prototyp aplikace tyto informace podstatné nebudou, ale po zpřístupnění aplikace veřejnosti budou jistě užitečné.

2.1.4.2 NoSQL databáze

Firebase Firestore [8] umožňuje i pro velké objemy dat vytvářet rychlé dotazy.

Na Firestore se lze v Androidu napojit pomocí knihovny přímo od Firestore. Ta umožňuje jak jednorázové načtení aktuální verze hledaných dat (jako je běžné v RESTu), tak i průběžnou aktualizaci dat. Zároveň podporuje ukládání offline kopie dat na zařízení pro čtení bez přístupu k internetu, nebude tedy potřeba toto řešit jiným způsobem. [8]

Firestore se velmi snadno škáluje, rozšíření kapacity je (i pro velmi velké objemy dat) pouze finanční otázkou. Při překročení limitu objemu uložených dat či překročení denních limitů na objem provozu, počty čtení a úprav, je potřeba pouze platit odpovídající dražší vyúčtování. [8]

[4] „Jediný zdroj pravdy.“ Tedy takovou strukturu, ve které je každá věc definovaná na právě jednom místě.

2.1.4.3 Autentizace uživatelů

Firebase Authentication umožňuje nejen registraci a přihlašování přes kombinaci e-mailu a hesla (a související ověřování e-mailů a resetování hesel), ale i přihlašování přes běžně používané externí služby (Google, Facebook, ...) a ověřování přihlášení druhým faktorem [13]. Aplikaci tak v budoucnu nebude problém o tyto funkcionality rozšířit.

Navíc je toto komplexní řešení již ve výchozím stavu integrováno na Firestore, takže lze Firebase autentizaci ověřovat v pravidlech pro přístup k datům uložených ve Firestore [14].

2.1.5 Distribuce

Pro distribuci bude použita distribuční služba Firebase [15], která se sice nehodí pro distribuci běžným uživatelům, ale je dostatečně uživatelsky pohodlná, aby s ní neměli problém aplikaci instalovat testeři.

Jak bylo zmíněno při definici požadavků (1.3), výstupní stav aplikace bude prototyp určený pro uživatelské testování. Zpřístupnění aplikace širší veřejnosti je plánováno až po vyhodnocení výsledků daného testování, zapracování problémů při něm nalezených, a dodání funkcionalit potřebných pro dlouhodobý běh aplikace (změna a reset hesla, ověření e-mailu). Prozatím tedy Firebase distribuce plně postačí.

2.2 Případy užití

Funkční požadavky (1.3.1) byly převedeny na případy užití – diagram[5] 2.1. V případech užití figurují dva „typy“ uživatelů: nepřihlášený uživatel (který může a nemusí mít účet) a přihlášený uživatel.

Nepřihlášený uživatel si bude moci vytvořit účet, a k účtu se bude moci přihlásit. Jiné akce v aplikaci nebude moci provádět.

Přihlášený uživatel si bude moci upravit svou preferovanou měnu a bude si moci zobrazit:

- Seznam skupin, k nimž je připojený. (Připojený je jak ke skupinám, k nimž se připojil, tak ke skupinám, které sám vytvořil.)
- Detail libovolné z připojených skupiny, včetně jejího identifikátoru (např. pro sdílení pro pozvání dalších uživatelů), seznamu všech transakcí a seznamu *lokálních členů* skupiny s jejich aktuálními dlužnými stavy.
- Detail libovolné transakce z libovolné z připojených skupin.
- Svůj dlužný stav skrz připojené skupiny (má-li vyplněnou preferovanou měnu).

Ve skupinách, ke kterým je připojený, může vytvářet nové transakce. Libovolnou transakci, kterou si může zobrazit, může také upravit či smazat.

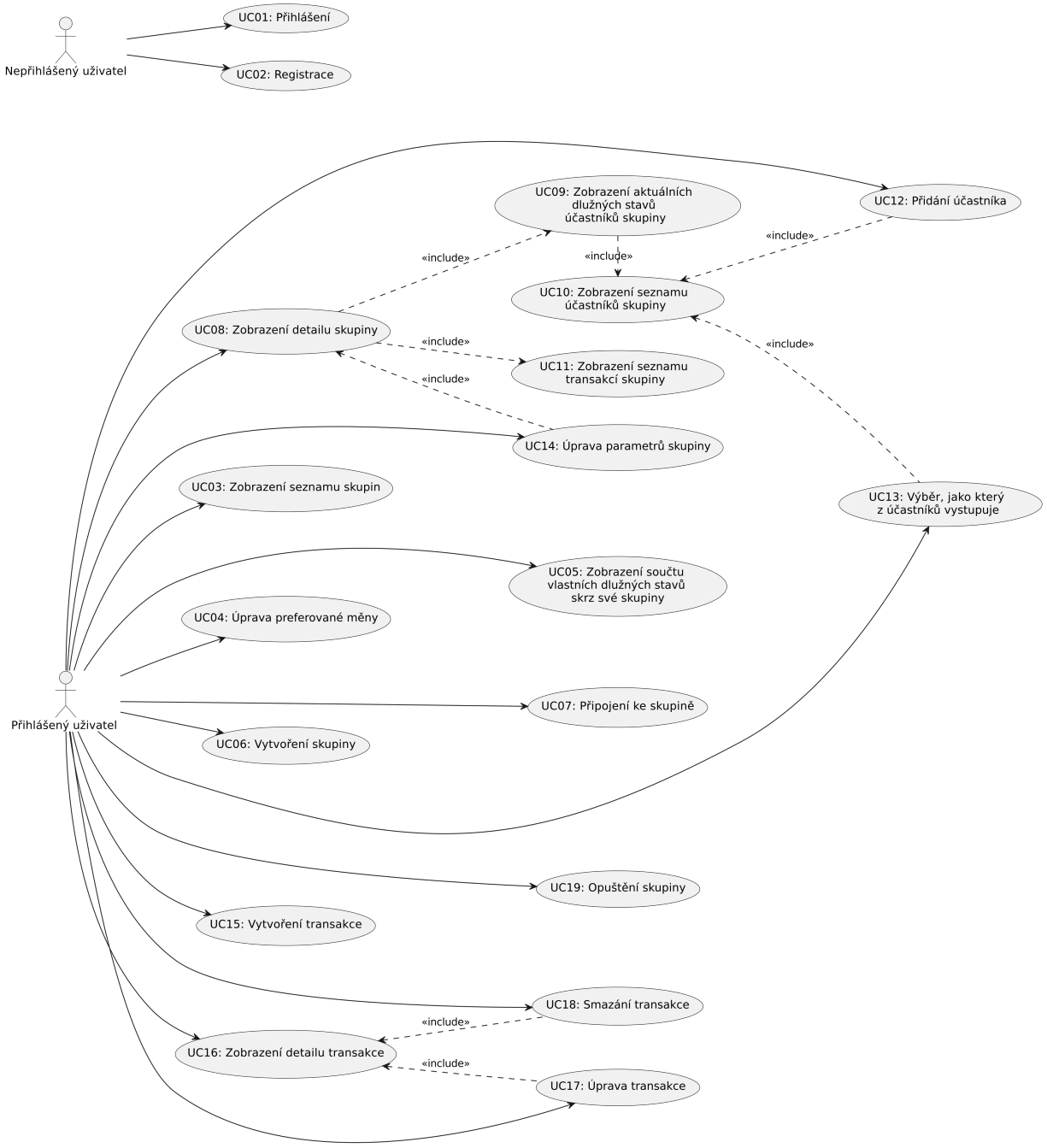
Může vytvářet nové skupiny. Může se připojovat ke skupinám, jejichž identifikátor má k dispozici.

Veškeré skupiny, ke kterým je připojený, může upravovat (změnit jim název, skupinovou měnu, přidávat členy). Může upravit, jako který z účastníků skupiny v dané skupině vystupuje (včetně možnosti nevystupovat jako žádný z nich). Libovolnou skupinu, ke které je připojený, může opustit.

Pokrytí funkčních požadavků případy užití je v tabulce 2.1. Byly pokryty všechny funkční požadavky.

Pro přehlednost jsou očíslované případy užití (bodovitě, bez detailů) vypsány i níže.

[5] Pro definici diagramů pro tuto práci byl použit jazyk PlantUML [16], pro konverzi v obrázky stránka [17].



Obrázek 2.1 Diagram případů užití.

Tabulka 2.1 Pokrytí funkčních požadavků případy užití.

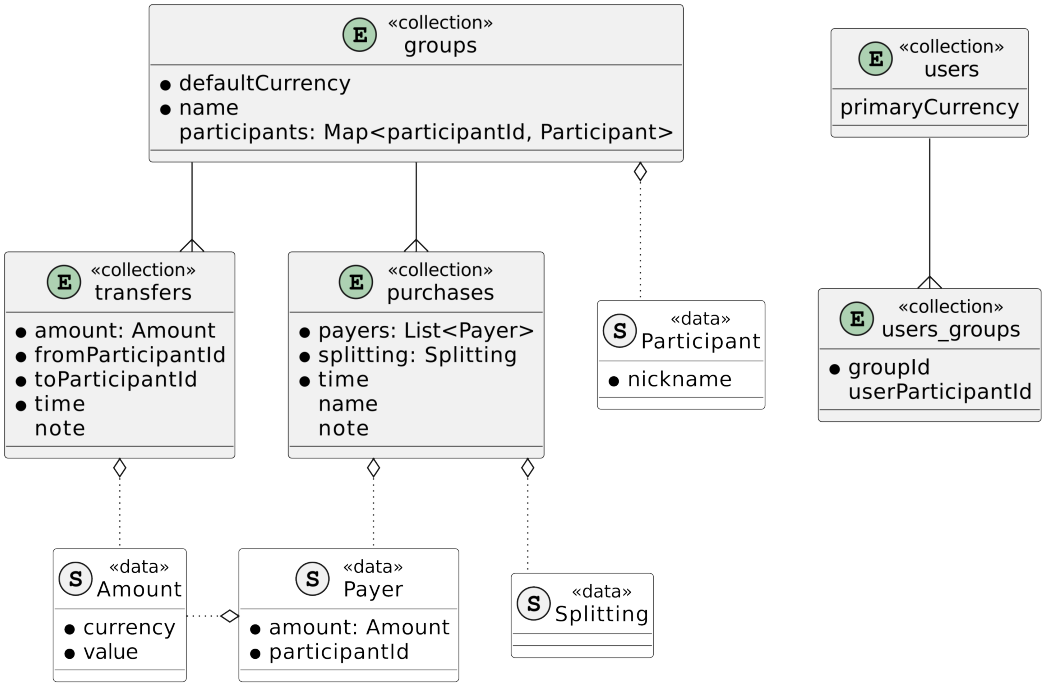
Případ užití	FN1	FN2	FN3	FN4	FN5	FN6	FN7	FN8	FN9
UC01	✓								
UC02		✓							
UC03			✓						
UC04									✓
UC05									✓
UC06			✓						
UC07			✓						
UC08			✓	✓		✓		✓	
UC09				✓				✓	
UC10				✓					
UC11						✓			
UC12				✓					
UC13				✓					
UC14			✓						
UC15					✓				
UC16						✓			
UC17							✓		
UC18							✓		
UC19			✓						

- UC01: Přihlášení.
- UC02: Registrace.
- UC03: Zobrazení seznamu připojených skupin.
- UC04: Úprava preferované měny.
- UC05: Zobrazení součtu vlastních dlužných stavů skrz své skupiny.
- UC06: Vytvoření skupiny.
- UC07: Připojení ke skupině.
- UC08: Zobrazení detailu skupiny.
- UC09: Zobrazení aktuálních dlužných stavů účastníků skupiny.
- UC10: Zobrazení seznamu účastníků skupiny.
- UC11: Zobrazení seznamu transakcí skupiny.
- UC12: Přidání účastníka.
- UC13: Výběr, jako který z účastníků vystupuje.
- UC14: Úprava parametrů skupiny (názvu, měny).
- UC15: Vytvoření transakce.
- UC16: Zobrazení detailu transakce.
- UC17: Úprava transakce.
- UC18: Smazání transakce.
- UC19: Opuštění skupiny.

2.3 Databázové schéma

Firestore ukládá data do kolekcí dokumentů. Dokumenty mají identifikátor, který je v rámci dané kolekce unikátní. V dokumentech jsou data ukládána do polí sestávajících z názvu a hodnoty[6]. Pod dokumenty lze zakládat sub-kolekce, ve kterých mohou být další dokumenty. [18]

V NoSQL databázích obecně (ani ve Firestore konkrétně [18]) není nutné mít zadefinované pevné schéma [19]. Dokumenty ve stejné kolekci mohou mít naprosto odlišnou strukturu. Pro udržitelnost do budoucna však bývá vhodné navrhnout pro jednotlivé kolekce konvence a omezení. Navíc je ve Firestore nutné rozhodnout, které skupiny dat budou v jednom dokumentu a které budou rozdělené do vícero dokumentů (a v jaké kolekci budou které dokumenty).



Obrázek 2.2 Orientační databázové schéma. Plné čáry značí sub-kolekce, tečkované zanořené objekty.

Navržené schema je na obrázku 2.2. Pro NoSQL databáze zatím není zaveden standard diagramů, zde jsou tedy kolekce dokumentů nadepsány `collection`, schémata vnořených objektů `data`. Atributy „na kolekcích“ budou v jejich dokumentech, atributy s tečkou jsou pro tyto dokumenty povinné. Sub-kolekce jsou značeny plnou čárou, zanořené objekty jsou s dokumenty spojeny tečkovanou agregační vazbou. Pro přehlednost u polí, jejichž hodnotou bude „jednoduchý“ nezanořený typ (textový řetězec, číslo, timestamp), typ vynechán, poznačeny byly jen zanořené typy. Typ `Splitting`, který musí umožňovat ukládat různé druhy dělení transakce mezi účastníky, byl ponechán pro zjednodušení prázdný: budou do něj ukládány různé struktury dat pro různé typy dělení.

Hodnoty mohou být nejen základních typů, jsou i dvě možnosti, jak zanořit (potenciálně) více objektů [20]:

- *Array*: řazený seznam objektů.
- *Map*: mapa/slovník. Bude použita jak pro párování hodnot k arbitrárním identifikátorům (na obrázku 2.2 hodnota pole `participants` v dokumentech kolekce `groups`), tak pro vytvá-

[6] Struktura připomínající JSON objekty.

ření zanořených objektů. Objekty označené `<<data>>` na obrázku 2.2 budou realizovány jako mapa, byly znázorněny takto jen pro přehlednost.

Při návrhu bylo nutné provést rozhodnutí, která data „k dokumentu“ budou atributem dokumentu, a která budou v nějaké jeho sub-kolekci. Pro nezanořené hodnoty, či hodnoty s velmi málo zanořeními a malou velikostí, bývá typicky nejlepší umístění přímo do dokumentu: není pak třeba provádět při čtení dokumentu a těchto hodnot vícero volání. Zajímavější je tato otázka pro situaci, kdy je potřeba k dokumentu uložit (potenciálně nemalé) seskupení elementů. Pak se zvažuje:

- Možná celková velikost elementů: Firestore dokument může mít maximálně 1 MB [18].
- Rychlost načítání: sub-kolekce musí být načteny zvlášť (ale lze načíst několik dokumentů ze sub-kolekce najednou), Firestore neumožňuje načtení dokumentu a jeho sub-kolekce „zabalit“ do jednoho volání.
- Možnosti formulace požadovaných omezení práv čtení: nejmenší granularita pro povolení či odmítnutí práva čtení je dokument, nelze jednomu uživateli povolit číst „jen některé položky dokumentu.“ (U pravidel pro zápis je možné řešit práva na úpravu jednotlivých polí dokumentu.) [14]

Zde jsou odůvodnění zvolených variant pro jednotlivé položky, pro které byla možnost oddělení do sub-kolekce zvažována:

- Seznam uživatelových skupin `users_groups`: byla zvolena sub-kolekce. Každý uživatel bude načítat jen svůj vlastní seznam, což nezpůsobí výrazný výkonostní problém. Ač v rámci této práce patrně nebudou benefity sub-kolekce využity, do budoucna bude velmi výhodná: v dokumentu v kolekci `users` je prostor na uživatelská data, ke kterým nemá mít přístup žádný jiný uživatel (ani pro čtení), zatímco z `users_groups` mohou v budoucnu mít např. administrátoři skupin možnost odstraňovat záznam o „své“ skupině (pro vyloučení záškodnického člena).
- Seznam *lokálních členů* skupiny `participants` pro dokumenty kolekce `groups`: byla zvolena mapa přímo v dokumentu. Pro rozumné nekomerční užití patrně nehrozí, aby s plánovaným schématem dokumenty `groups` překročily povolený velikostní limit, byla tedy zvolena rychlejší varianta. Ač je zatím v plánu pro členy evidovat jen přezdívku, je tato zabalená v zanořeném objektu pro snadné přidání případných dalších (velikostně nenáročných) atributů. Kdyby se do budoucna ke členům začaly ukládat značně větší datové objekty, mohou tyto být umístěny do nové sub-kolekce `groups` a párovány v aplikaci před identifikátor člena.
- Seznamy nákupů `purchases` a převodů `transfers`: byly zvoleny sub-kolekce kvůli možné celkové velikosti těchto množin. Zároveň bylo na základě rozhovorů s potenciálními uživateli rozhodnuto, že objekty pro nákupy a převody peněz budou mít jinou strukturu – nákupy musí mít strukturu bohatší, aby umožňovaly zachytit komplexnější situace, převod peněz však typicky proběhne od jediného účastníka k jedinému jinému účastníkovi, uživatelé by tedy mohli pro převody peněz ocenit jednodušší rozhraní. Byly tedy navrženy oddělené objekty, které budou umístěny do dvou různých kolekcí – pro implementaci tak bude minimalizováno množství nutné manuální deserializace[7] a problémů s ní spojených.

2.4 Návrh obrazovek

Obrazovky by pro uživatele být co nejpřehlednější: pokud uživatel otevře obrazovky s úmyslem zjistit nějakou informaci, měl by ji být schopen rychle a intuitivně dohledat; pokud otevře obrazovku se záměrem provést nějakou akci, měl by ji být schopen snadno a intuitivně provést.

[7] Pro skok vpřed v čase: 3.4.2 Firestore entity

Obrazovky budou realizovány v minimalistickém stylu.

V návrhu obrazovek je podstatná ergonomie, avšak i když je záměr řešit v rámci této práce jen ergonomii pro uživatele používající mobilní telefon pomocí typické zdravé pravé ruky, rok 2022[8] této snaze nepřeje: podstatná část vlastníků mobilních telefonů již používá telefon tak velký, že již mají problém pohodlně dosáhnout na vršek displeje palcem ruky, ve které telefon drží. Majitelé přežívajících menších telefonů typicky mohou pohodlně interagovat s (alespoň) celou spodní polovinou displeje a, drží-li zařízení v pravé ruce, alespoň s pravým horním rohem displeje. Majitelé větších telefonů mohou pohodlně utilizovat jen spodní (zhruba) polovinu displeje. Svět designu pomalu na trend odpovídá, více ovládacích prvků se přesouvá ke spodnímu okraji displeje[9]. Zůstává jeden drobný zádrhel: uživatelé jsou stále zvyklí vidět „nadpis“ obrazovky nahoře[10], mít tento element vespod obrazovky by působilo podivně. A když už je místo na vršku obrazovky vyčleněno pro tento text... co tam přidat nějaký ovládací prvek?

Jednou variantou by bylo mít pro různé velikosti displeje různé rozložení prvků. Což by mohlo být velmi moudré rozhodnutí ve větším týmu, ale pro jednoho člověka by to příliš ztížilo následné vývojářské testování během implementace. Vynaložit čas na odladění více variant rozložení prvků, a následně varianty testovat po každé větší úpravě, není v rámci této práce reálné.

Byly provedeny rozhovory s několika uživateli větších telefonů. Všichni se shodli, že jsou zvyklí pro méně často používané prvky sahat druhou rukou, než ve které zařízení drží. Také uvedli, že pro navigaci na předchozí obrazovku využívají raději systémové tlačítko „zpět“ či gesta, ne tlačítka „zpět“ v aplikaci. Tato práce se tedy – pro prvky, u kterých není očekávána příliš častá uživatelská interakce, včetně zpětné navigace – nebude bránit umísťování některých ovládacích prvků na horní část displeje, aby na přežívajících menších telefonech byl ušetřen prostor. Zmigrovat aplikaci v budoucnu na jiné schéma rozložení ovládacích prvků by nemělo být náročné.

2.4.1 Rozvržení obrazovek

Nejprve bylo potřeba udělat rozhodnutí, jaké obrazovky mají vzniknout: některé případy užití mohou sdílet jednu obrazovku, jiné mohou samy vyžadovat i několik dedikovaných obrazovek[11]. Při rozhodování byly brány v potaz nejen aktuálně plánované případy užití, ale i potenciální rozšiřování funkcionalit do budoucna: pro připojení ke skupině bude nejspíše vždy potřeba jen vložit identifikátor skupiny to jediného textového pole, ale editace parametrů účtu může v budoucnu zahrnovat nemálo polí (ač nyní je počítáno s jediným: uživatelova volba preferované měny).

Na základě vzniklého seznamu obrazovek s jejich funkcionalitami bylo potřeba navrhnout, jak bude uživatel moci mezi obrazovkami procházet. Nejvíce používané funkcionality by měly být „u sebe“, zároveň by měly možnosti přechodů být na místech, kde by je uživatel intuitivně hledal. Navrhnutá navigace je načrtnuta[12] na obrázku 2.3. Černé šipky ukazují, kam se „uživatel může vydat“, modré šipky jsou „přesměrování po té, co uživatel na zdrojové obrazovce úspěšně dokončí akci“ (nejde o pevnou klasifikaci), zelená část diagramu nastiňuje výběr, která obrazovka bude ukázána při spuštění aplikace. Pro zachování přehlednosti není znázorněno, které přechody bude uživatel moci realizovat i „v protisměru“ volbou „zpět“, která bude samozřejmě podporována na místech, kde není specifický důvod ji vyloučit (např. změna stavu přihlášení, znovu-otevření obrazovky pro úpravu dat, kde hrozí zmatení, zda byly/budou změny uloženy apod.). V diagramu jsou zároveň barevně zvýrazněny obrazovky, u kterých je očekáváno, že na nich uživatel stráví

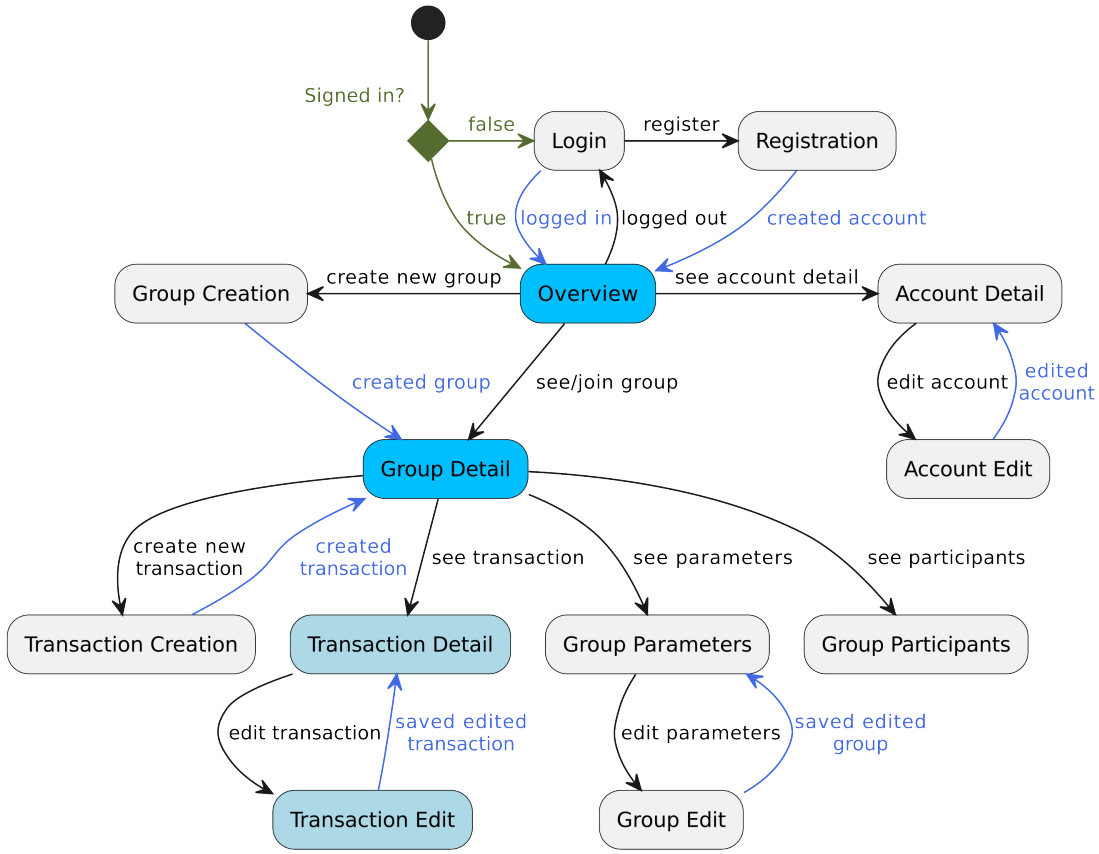
[8] ani 2023

[9] Např. v doporučeních od Google lze srovnat starší verzi koncentrující primární uživatelské akce do horního panelu [21], zatímco novější verze již doporučují mít např. navigační prvky ve spodním panelu [22].

[10] Je poměrně ilustrativní, že autorku při psaní nenapadá, jak lépe označit uživateli zobrazený „název“ obrazovky, než krásné české slovo NADpis.

[11] Například aktuální dlužné stavy účastníků skupiny lze zobrazit nad seznamem transakcí dané skupiny. Na druhou stranu údajů o skupině je mnoho – mít všechny zobrazené na jedné obrazovce (potenciálně včetně elementů pro navigaci k možnosti úpravy oněch údajů,) by snadno vytvořilo velmi nepřehlednou změť.

[12] Pro definici byl použit jazyk PlantUML[16], pro konverzi v obrázek stránka [17]. Ctěný čtenář snad promine netaktní zneužití elementů Activity diagramu pro tento účel.



Obrázek 2.3 Navigace mezi obrazovkami.

nejvíce času: uživatelská přívětivost těchto obrazovek bude obzvlášť důležitá.

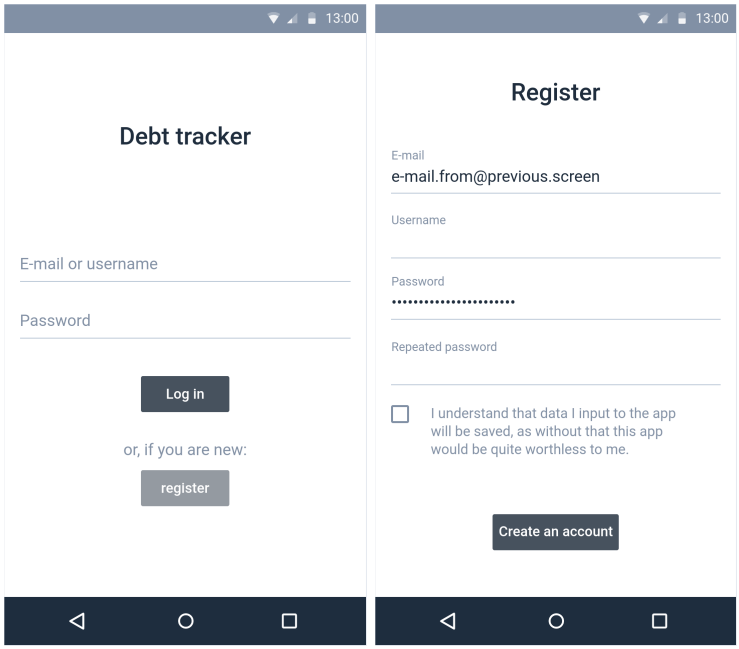
Vzniklý diagram navigace spolu se slovním popisem očekávaného obsahu obrazovek byl validován se dvěma potenciálními uživateli. Oba považovali rozložení za intuitivní a snadno pochopitelné. Oba byli následně vyřazeni ze seznamu potenciálních testerů.[13]

Pro následný návrh obrazovek bylo použito Lo-fi[14] prototypování: postup, při kterém se vytváří orientační náčrtek grafického rozhraní (ať už na papír, tabuli, či v grafickém nástroji). Důraz je kladen na rychlost výroby a následné úpravy tohoto prototypu: cílem je si na prototypu „vyzkoušet“ různé možnosti pozicování elementů. Takto je možné v únosném čase projít vícero možností rozložení elementů a vybrat to uživatelsky nejpřívětivější.

Pro několik obrazovek byl vytvořen prototyp v nástroji Marvel App [23] – bezplatná verze webové aplikace umožňuje rychle vytvořit návrh s pomocí směsi předdefinovaných komplexnějších vizuálních elementů, ikon, jednoduchých tvarů a textu. Konkrétně pro dvojici jednodušších obrazovek (obrázky 2.4), kde autorce pro prvotní pozicování prvků nestačil papír a tužka. Byl následně použit i pro některé komplexnější obrazovky (obrázky 2.5), kde byly mimo pozicování i naznačeny další detaily (použité barvy a styly textu jsou však stále velmi orientační, jde tedy o návrhy nespadající plně mezi Lo-fi ani Hi-fi prototypy).

[13] Konzultující by si mohli pamatovat rozhovor o plánovaném obsahu obrazovek, nemohli by tedy otestovat, zda vzniklá aplikace připadá intuitivní i uživateli, který neprošel podobným rozhovorem. Nutnost tohoto vyloučení byla také důvodem, proč byly konzultující jen dva: autorka si chce ponechat dostatek potenciálních testerů na finální testování aplikace.

[14] *low fidelity*, „málo přesné“



Obrázek 2.4 Prototypy jednodušších obrazovek: přihlášení a registrace.

2.4.2 Material Design

Společnost Google vydává Material Design [24] – doporučení pro návrh grafického rozhraní aplikací. Dostupné jsou nejen obsáhlé texty s doporučeními a doprovodnými ilustracemi, ale i knihovny s komponentami pro Jetpack Compose, které se těchto doporučení drží. Tato práce využije Material Design 2[15] – nemálo Compose komponent a část doporučení. Práce se však doporučení nebude plně držet, dovolí si i odchylky k jiným minimalistickým schématům. Zda tyto odchylky uživatelům vyhovují či překáží (a zda by tedy bylo vhodné je ponechat, upravit či odstranit), bude zjištěno v rámci uživatelského testování (4.2).

Čtenář si může všimnout, že prezentované návrhy obrazovek ani nepoužívají Compose komponenty z Material Design 2 (např. mnoho komponent má ostré rohy, ne zaoblené). Jak bylo zmíněno v 2.4.1, jedná se o hrubé prototypy, v implementaci je v plánu se více přiblížit vizuálním standardům Material Design 2.[16]

2.4.3 Obrazovky

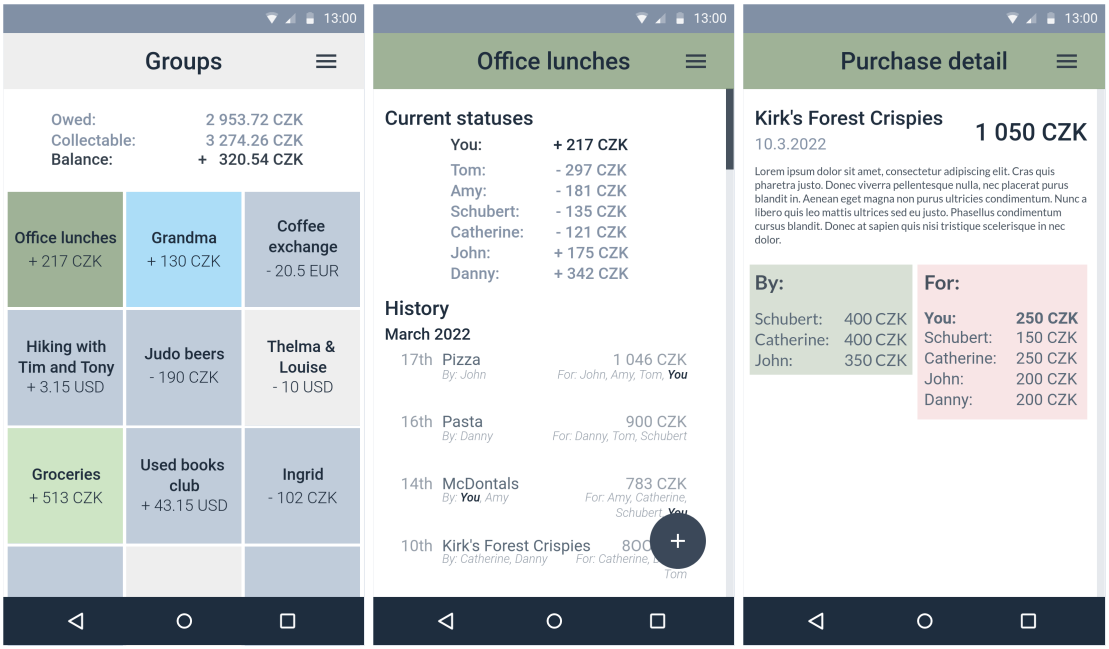
Níže jsou popsány návrhy dvou z komplexnějších obrazovek. Jelikož nebylo nutné se v práci od původních návrhů příliš odchýlit, je jednotlivým obrazovkám věnováno více textu v následující kapitole o implementaci (3.6), kde čtenář již místo načrtnutých prototypů uvidí snímky vzniklé aplikace.

2.4.3.1 Přehled a seznam skupin

„Domovská obrazovka“ aplikace, uživatel se na ní dostane z obrazovek pro přihlášení, registraci, kliknutím na ikonku domečku tam, kde bude ikona zobrazena i pouhým spuštěním aplikace (je-li již přihlášen). Prototyp viz 2.5.

[15] V době, kdy začala vznikat kostra implementace plánované aplikace, byl Material Design 3 dostupný jen v potenciálně nestabilní alpha verzi. Nyní již je dostupný ve stabilních verzích.

[16] Čtenář může poskočit dopředu v čase a podívat se např. na screeshoty výsledné aplikace v příloze A.



Obrázek 2.5 Prototypy komplexnějších obrazovek. Zleva: celkový přehled, detail skupiny,detail výdaje.

Zobrazuje:

- Seznam skupin – v náhledu je vidět jméno skupiny a aktuální dlužný stav uživatele v dané skupině.
- Sumy dlužných částek – zobrazuje součty aktuálních negativních i pozitivních stavů napříč skupinami a z nich vypočítanou celkovou bilanci. Částky jsou uváděny v preferované měně uživatele (pokud měnu ještě nezadal, zobrazí se místo bilancí jen informace, že si má měnu zvolit).

V menu má uživatel možnost:

- provést odhlášení,
- přejít na detail svého účtu,
- připojit se k existující skupině,
- začít vytvářet skupinu novou.

2.4.3.2 Detail skupiny

Prototyp viz 2.5. Na obrazovce uživatel uvidí:

- Transakce:
  - seřazeny podle data uskutečnění,
  - s datem uskutečnění,
  - s částkou (v měně, ve které byla zaznamenána)
  - s názvem, byl-li vyplněn,
  - s indikací, zda jde o převod peněz mezi účastníky či společný výdaj,

  - se seznamy účastníků, jichž že týká (platící a „dlužící“),
  - se zvýrazněnými výskyty uživatele mezi účastníky transakce (tučné písmo v seznamu z předešlého bodu)
- Aktuální dlužné stavy účastníků:
  - ve skupinové měně,
  - seřazené podle dlužného stavu,
  - se zvýrazněným výskytem uživatele (může být vyjmut z řazení a umístěn na začátek seznamu)

  Z obrazovky lze přejít:
- zpět na Přehled 2.4.3.1 (potenciálně navigační ikonou vlevo nahoře, minimálně však bude upravená vestavěná Android navigace „zpět“ tlačítkem/gestem, aby se při volbě „zpět“ uživatel vždy dostal na Přehled, bez ohledu na to, odkud na Detail skupiny přešel),
- na detail transakce (klepnutím na transakci),
- na vytváření nové transakce (přes *Floating Action Button*[17]),
- na seznam účastníků skupiny (přes menu),
- na obrazovku s parametry skupiny (včetně jejího ID; přes menu).

[17] Tlačítko „vznášející se“ nad částí obsahu, typicky umisťované do pravého spodního rohu displeje.

Kapitola 3
Implementace

Zdrojový kód byl od počátku implementace verzován pomocí systému Git a školního GitLab serveru. Pro vývoj bylo využito Android Studio [25], pomocí kterého byl i vygenerován základ projektu.

3.1 Architektura aplikace

Aplikace využívá návrhového vzoru *Model–View–ViewModel* (MVVM). To, co je z pohledu MVVM považováno za *Model*, je dále rozděleno na doménovou vrstvu a persistentní vrstvu (podobně jako dvě nižší vrstvy třívrstvých serverových aplikací). Důvodem je oddělení logiky ukládání dat ve Firestore od hlavní logiky aplikace. Kód je díky oddělení vrstev do budoucna jednodušší udržovat, upravovat, i případně rozdělit do modulů pro přepoužití nižších vrstev v jiném programu.

V datové/perzistentní vrstvě se využívají 3.4.2 Firestore entity. Jejich čtení a zápis na úrovni jednotlivých kolekcí či konkrétních dokumentů zpřístupňují implementace rozhraní `Db`. Tato volání skládají[1] *Repository*, které zároveň tvoří hranici mezi datovou a doménovou vrstvou: přijímá a vrací doménové objekty, interně pracují s Firestore entitami a provádí obousměrné mapování.

V doménové/aplikačně-logické vrstvě se používají objekty 3.4.1 doménového modelu. Operace, které budou požadovat *ViewModely*, jsou zaobaleny jako celek do *UseCase*[2] tříd. Když bude v budoucnu potřeba nějakou operaci upravit – například dodat transformaci či dekoraci dat někam, kde dříve nebyla – bude stačit upravit danou *UseCase* třídu, ve *ViewModelech* bude maximálně nutné vyřešit případnou změnu typů posílaných objektů. Kód je tak do budoucna snadněji udržovatelný, protože při změně operace nehrozí opomenutí původní logiky v některém z *ViewModelů* využívajících danou operaci.

ViewModely stojí mezi doménovou vrstvou a obrazovkami. Při implementaci je využito Android `ViewModel` třídy, které řeší korektní „přežívání“ instance *ViewModelu* v průběhu životního cyklu Android komponent [27].

ViewModel je zodpovědný za načítání dat pro obrazovku, a za udržování jak těchto načtených dat, tak případných komplikovanějších pracovních dat, které potřebuje obrazovka mít k dispozici. Za tímto účelem u sebe od inicializace drží aktuální hodnotu *ViewState* v `MutableStateFlow`. Pokud dané *ViewState* při inicializaci nemůže dostat k dispozici potřebná data pro všechny své

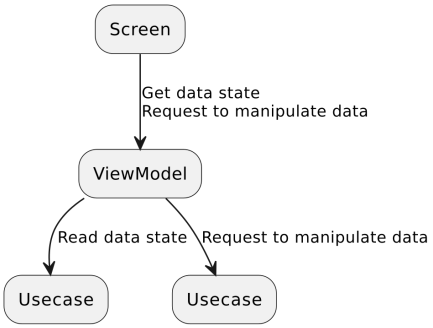
[1] Volání je ve schématu potřeba skládat relativně často, např. pro zobrazení detailu nákupu musí být data z dokumentu samotného nákupu odekorována o informace o přezdívky účastníků skupiny z dokumentu samotné skupiny.

[2] V této práci slovo *UseCase* nepoužívám jako anglické synonymum pro *případ užití*, ale jako konvenční jméno pro třídy zpřístupňující operace *ViewModelům*. Nejde o jediné možné pojmenování této skupiny tříd, ale je to běžná konvence (viz např. [26]), která je dodržena i v kódu vzniklé aplikace.

atributy (např. kvůli nutnosti jejich získání z Firestore), jsou dané atributy do jejich získání nastaveny na `null` a do jejich vyplnění je ve *ViewState* nastaven atribut `loading` typu bool na hodnotu `true` (pravda). *ViewModel* obrazovce ihned po inicializaci zpřístupňuje *ViewState* jako `StateFlow` – *immutable* variantu vnitřního atributu.

Obrazovky (*View*, prezentační vrstva) jsou realizovány jako Compose funkce. Navigace mezi nimi je řešena pomocí standardních Compose navigačních komponent [28], které umožňují bloky kódu (provolání funkce obrazovky, a kód umožňující následnou navigaci na další obrazovky) registrovat pod textovou cestou potenciálně zahrnující jednoduché proměnné jako např. identifikátory či textové řetězce. Pomocí této textové cesty je následně možné do daného bloku navigovat z jiného. (Schéma je podobné internetovým URL adresám.)

Obrazovky dostávají identifikátory objektů, se kterými mají pracovat (např. obrazovka pro zobrazení detailu nákupu dostává ID skupiny, pod kterou nákup spadá, a ID daného nákupu). Zavolaná obrazovka si získá svůj korespondující *ViewModel*, který případně na základě identifikátorů nechá na pozadí získat z doménové vrstvy data. Ilustraci vzájemného volání lze nahlédnout na obrázku 3.1.



Obrázek 3.1 Ilustrace volání mezi obrazovkou (*Screen*), *ViewModelem* a (zde dvěma) *UseCase*.

3.1.1 Flow

Skrz celou aplikaci se pro hodnoty, které se mohou měnit v čase, používá `Flow` [29]. Jedná se o asynchronní sekvenční stream, který po aktivaci postupně vrací data specifikovaného typu. Stream může být předán volajícímu „rovnou“, bez ohledu dobu trvání operací nutných k poslání dat streamem, a je pak na volajícím, zda a kdy nad streamem zavolá `collect`, čímž jej aktivuje a po obdržení hodnoty provede kód v `collect` bloku. `Flow` je také použito pro obalení volání Firestore databáze, která výsledky operací zpřístupňuje jen pro registrované *Listenery*[3] – implementace rozhraní `Db` registrují *Listenery* a ve `Flow` vrací jimi získaná data. Zbytek kódu aplikace tak nemusí řešit např. registraci *Listenerů* pro možné úspěšné i neúspěšné průběhy dané Firestore operace, jen konzumuje daný `Flow`.

Pro udržování hodnoty *ViewState* ve *ViewModelech* je použit již zmíněný `MutableStateFlow` [30] mající *thread-safe*[4] extension metodu `update` [31], která při pokusu o souběžnou úpravu hodnoty případně provolá lambdu ze svého argumentu opakovaně, dojde-li k souběžné změně hodnoty. Obrazovka pak nad předanou *immutable* kopií `StateFlow` volá Compose extension metodu `collectAsState` [32] umožňující rekompozici[5] na základě každé nové hodnoty, která ze streamu přijde.

[3] Místo např. funkce pro získání aktuálního stavu dokumentu (či streamu jeho stavů), umožňuje knihovna „zaregistrovat“ k danému dokumentu blok kódu, který spustí při každé změně jeho stavu.

[4] Bezpečná pro použití v asynchronním programu.

[5] Znovu-vykreslení těch částí grafického rozhraní, které byly ovlivněny změnou hodnoty.

3.1.2 Zpracovávání chyb

V programování jsou tři běžné způsoby předávání chyby z funkcí: vyhození výjimky, předání v návratové hodnotě či nastavení nějakého indikátoru (ať již úprava hodnoty předávané funkci v parametru referencí, či v případě assemblerových procedur nastavením bitu ve specifickém registru). Poslední varianta nebyla pro tuto práci zvažována.

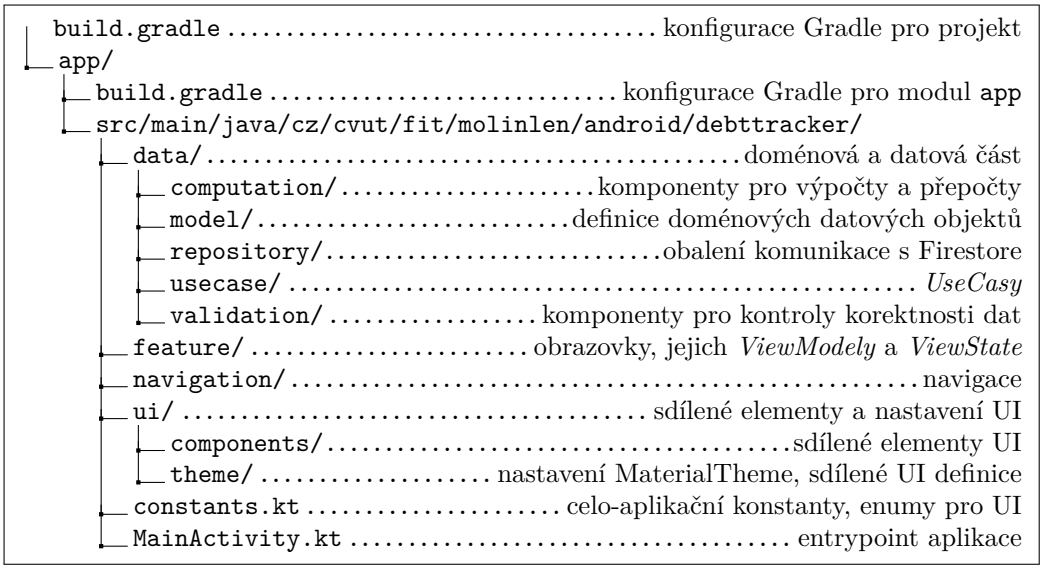
Vyhazování výjimek je velmi běžné: v mnoha jazycích včetně Kotlinu není nutné výjimky deklarovat v signatuře funkcí, pokud je do kódu přidán nový bod, kde může operace selhat, lze jen přidat nové vyhození výjimky, není třeba upravovat signatury. Co je na výjimkách méně intuitivní je řešení jejich konverze do indikací chyby vhodných pro prezentaci konzumentovi (ať jím je uživatel aplikace či program přistupující k vystavenému REST API). Díky tomu, že výjimky nejsou v signaturách funkcí povinné, není nikdy jasné, jaká výjimka může z kterého volání přijít. Má-li program více vrstev, které navíc používají různé knihovny (které typicky mívají vlastní množiny výjimek, které mohou vyhazovat), pak se toto typicky řeší globálními funkcemi pro odchyt výjimek na/před prezentační vrstvou, přebalováním výjimek na vrstvách, vytváření struktur dědičnosti v používaných výjimkách, či libovolnou kombinací. Řešení prezentovatelnosti chyb zkrátka bývá při použití výjimek kostrbaté.

Varianta s vracením chyby v návratové hodnotě vrací možné typy chyb do signatury funkcí a nutí programátory tyto možné chyby řešit, čímž se velmi snižuje šance, že aplikací proletí neodchycená výjimka – jen je nutné volání knihovních funkcí či jiných operací, které výjimky vyhazují, systematicky obalovat. Cenou je nutnost rozumného návrhu rozhraní – typ umožňující vrátit potenciální chybu je třeba přidávat do signatur kdekoli, kde do budoucna hrozí vracení chyby, v rozsáhlejších aplikacích bývá jinak zdlouhavé upravovat signatury a dodávat zpracovávání chyb zpětně. Realizovat vracení chyb v návratové hodnotě pomocí vlastních definic návratových objektů by mohlo být krkolomné, ale pro některé jazyky již existují knihovny, které toto umožňují dělat snadno, prakticky a čitelně. Pro tuto práci bude využita knihovna Arrow Core [33], dostupná pod licencí Apache 2.0, konkrétně její objekt `Either` [34] a k němu vázané pomocné funkce.

`Either` je generický typ o dvou generických parametrech: „levého“ typu chyby a „pravého“[6] typu nechybové hodnoty. Instance typu `Either<L,R>` v sobě bude nést buďto chybovou hodnotu typu `L`, nebo nechybovou hodnotu typu `R`. Metody `fold`, `map`, `flatMap`, `mapLeft` umožňují instanci `Either`u libovolně transformovat (podle toho, zda v daném místě v kódu chceme transformovat chybovou hodnotu, úspěšnou hodnotu, či obě) a umožňují tak psát kód pracující s instancemi `Either`u přehledně. Pro volání knihovních funkcí, které vyhazují výjimky, se hodí `Either.catch`, který ze zadaného bloku kódu vrátí instanci s levým typem `Throwable`[7] a pravým typem návratové hodnoty bloku. Za zmínku stojí ještě kombinace `either.eager` s blokem, ve kterém se vyskytuje `bind` – hodí se, když je třeba nad vícero hodnotami (i např. prvky kolekce) zavolat operaci vracející `Either`, a při prvním případném neúspěchu zpracovávání přerušit a vrátit danou chybu (na rozdíl od vyhazování výjimek však lze `bind` volat právě jen ve specializovaném bloku, nehrozí tedy opomenutí odchycení). [34]

[6] `Either.Left` - chyba, `Either.Right` - úspěch.

[7] Ve světě *Java Virtual Machine* výjimkový typ – vše, co lze „házet“ a „chytat“.



Obrázek 3.2 Struktura projektu.

3.2 Struktura projektu

Struktura hlavních složek se zdrojovým kódem je vidět v rozpisu 3.2.

Veškeré načítání, předpříprava a úpravy dat jsou ve složce *data*.

Veškerá logika pro potřebné výpočty částek se nachází ve složce *computation* – snižuje se tak šance, že stejné výpočty se budou provádět v kódu různě, komponenty ve složce jsou pokryty jednotkovými testy. Pokud budou v budoucnu využity nástroje pro automatickou kontrolu pokrytí kódu testy, mohou pro tuto složku být nastavena přísnější pravidla.

Veškeré volání Firestore, skládání těchto volání a transformace mezi persistentními a doménovými objekty je agregován ve složce *repository*. V podsložkách jsou zde entity, rozhraní `Db` a jeho implementace, transformační funkce i samotná *Repository*.

Znovu-použitelné komponenty pro validaci vstupů uživatele – pro jednotlivé hodnoty či komplexnější sady hodnot – jsou ve složce *validation*.

Metody, které jsou reálně volané z *ViewModelů*, jsou ve složce *usecase* – pro snadnější procházení souborů a s ohledem na malou velikost projektu jsou zde nestandardně rozhraní ve stejném souboru jako jejich implementace.

Obrazovka se svým *ViewModelem* a *ViewState* je s těmito objekty vždy v podsložce *feature*, pojmenované podle primárního využití dané obrazovky. Pokud obrazovky mají sdílet některé grafické prvky jen mezi sebou (např. obrazovky pro úpravu převodů a výdajů), mohou být složky obrazovek zanořeny do společné složky s oněmi sdílenými prvky. Grafické prvky obrazovky jsou v souboru obrazovky (prozatím postačilo, časem je možné případně rozdělit do více souborů), *ViewModel* má jeden soubor a v dalším je *ViewState*. V souboru *ViewState* mohou být případně zadefinované funkce pro transformaci datových objektů, u kterých se neočekává možnost použití i jinde v aplikaci.

Ve složce *components* jsou definované obecné grafické komponenty pro použití na libovolné obrazovce.

V *theme* jsou nastaveny hodnoty pro aplikační instanci *MaterialTheme*. Také je zde definováno několik málo vlastních hodnot používaných skrz aplikaci, které by bylo na místě zrevidovat při úpravách hodnot pro *MaterialTheme* (jde o pár barev pro zvýrazňování a definice tvarů komponent).

3.3 Integrace na Firebase

Byly využity přímo knihovny od Firebase pro Android aplikace v Kotlinu, které obstarávají volání Firebase API.

Pro získání aktuálního ID uživatele je Firebase knihovna volána napřímo z *ViewModelů*. Přihlášení, odhlášení a registrace jsou již obaleny do *UseCase*.

U databáze Firestore bylo zavedeno větší oddělení: rozhraní i implementace *UseCase* využívá doménových objektů. *UseCase* volají *Repository*, které mají také na rozhraní doménové objekty, až jejich implementace je transformují na 3.4.2 Firestore entity. *Repository* ve svých metodách volají různé metody databázového adaptéru, který zpřístupňuje jednotlivé datové operace s entitami, skládají jejich výsledky dohromady a transformují je na doménové objekty.

3.3.1 Konfigurace Firebase

Ve Firebase Authentication bylo nutné zapnout možnost přihlašování kombinací e-mailu a hesla.

Pro Firebase Firestore byla upravena bezpečnostní pravidla pro přístup ke kolekcím tak, aby mohli přistupovat jen přihlášení uživatelé. (Budoucí verze aplikace, které budou řešit práva uživatelů ve skupinách, vykazování ze skupin či jiné funkcionality by toto samozřejmě měly upravit podrobněji.)

Knihovna pro práci s Firestore ve výchozím nastavení realizuje ukládání lokální kopie dat pro možnost čtení bez přístupu k internetu, nebylo tedy potřeba její nastavení měnit.

Firebase Crashlytics jsou používány s výchozími nastaveními, postačilo tedy přidat korespondující knihovnu do závislostí aplikace.

3.4 Datový model

Doménová a prezentační vrstva mají pochopitelně velmi rozdílné požadavky na rozhraní tříd nesoucích informace o stavu dat, než má vrstva datová.

Datová vrstva bude číst data přímo z Firestore (kde budou uložena podle 2.3 Databázové schéma), a bude řídit manipulaci s těmito daty. Uložená data mají být minimalistická, pro snadné úpravy (a šetření místa v databázi) bez redundancí.

Naopak doménové objekty mají být bohatší, pohodlné pro práci v doménové a prezentační vrstvě.

Například u nákupu: datová vrstva bude jistě referencovat zúčastněné členy skupiny jen podle jejich identifikátorů. Ale pro zobrazení detailu nákupu (i malé karty nákupu v seznamu) uživateli bude jistě nejpohodlnější, když *UseCase ViewModelu* rovnou vrátí třídu, ve které jsou u členů účastnících se nákupu již vyplněny jejich přezdívky.

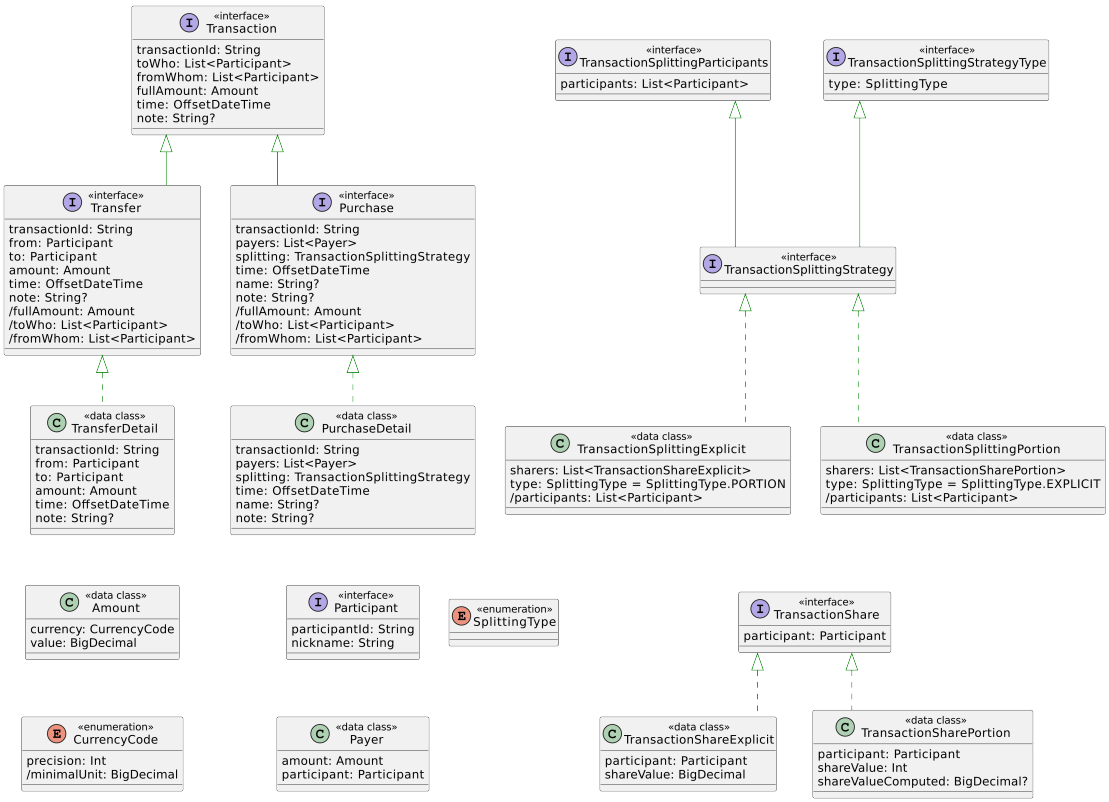
Byl tedy vytvořen oddělený doménový a persistentní model.

3.4.1 Doménový model

Diagram 3.3 ukazuje nejpoužívanější doménové třídy týkající se transakcí, diagram 3.4 nejpoužívanější doménové třídy týkající se skupin.[8]

V doménovém modelu byl kladen důraz na použití rozhraní v místech, kde je či do budoucna může být požadavek na vícero implementací. Část rozhraní byla také označena jako `sealed`: takovéto rozhraní může být implementováno pouze v rámci balíčku, ve kterém je definováno. Díky tomu jsou všechny jeho implementace známé již při kompilaci. Například pak není v Kotlinu třeba při použití `when` s argumentem definovat i případ, kdy přijde neznámý podtyp [36]. Když by časem

[8] Základ PlantUML [16] kódu použitého pro definování těchto diagramů byl vygenerován pomocí IntelliJ IDEA [35]. Vygenerovaný kód byl upraven.

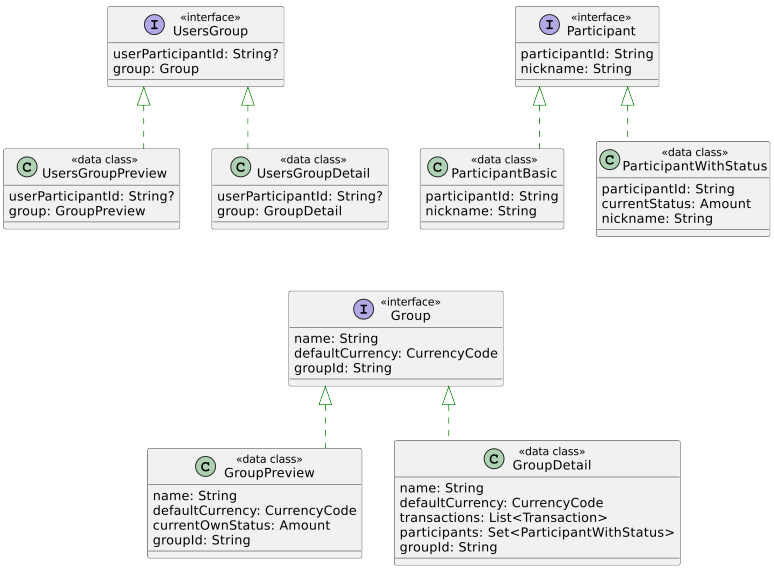


Obrázek 3.3 Diagram (části) tříd doménového modelu týkajících se transakcí.

někdo přidal nový typ, všechny argumentové `when` nad daným typem budou způsobovat chybu kompilace, a přidávající programátor tak bude nucen ošetřit veškeré chování závislé na daném typu. Tento stav je pro budoucnost aplikace bezpečnější, neboť alternativou je umožnění sestavení (a distribuce) verze aplikace, která nově přidaný podtyp podporuje jen v některých místech, zatímco v jiných s ním nefunguje korektně. Např. rozhraní `Transaction` je `sealed`, takže v kódu aplikace lze bezpečně předpokládat, že hodnota tohoto typu bude `Transfer` nebo `Purchase`.

Zajímavou otázkou bylo řešení reprezentace částek a měn. Pro čísla byl využit typ `BigDecimal`, který umožňuje provádět operace v desítkové soustavě a tím se vyhnout problémům s desetinnými čísly, které nelze reprezentovat binárně. Pro měny byla zvolena cesta vlastního `enum`u `CurrencyCode`. Bylo potřeba vyřešit i definici povoleného počtu desetinných míst, aby aplikace např. nenechala uživatele vkládat transakce s částkami ve zlomcích haléřů, či jim nezobrazovala, že mají dluh půl haléře. Množství povolených desetinných míst se liší mezi měnami[9], pro budoucí možné rozšíření seznamu měn byla tedy pro `CurrencyCode` přidána hodnota `precision`, která uvádí právě počet desetinných míst. Tato hodnota se využívá pro nastavení `scale` u `BigDecimal` např. při vytváření objektu pro částku `Amount` přes funkci `createAmount`, či při převodech mezi měnami.

[9] Ač jsou dvě desetinná místa velmi běžná, tak např. tuniský dinár počítá se třemi. [37]



Obrázek 3.4 Diagram (části) tříd doménového modelu týkajících se skupin.

3.4.2 Firestore entity

Entity ukládané do Firestore jsou realizací objektů navržených v 2.3 Databázové schéma.

Firestore knihovna umožňuje serializovat a deserializovat objekty posílané při komunikaci s Firestore serverem přímo do/z instancí `data class`. Této možnosti bylo využito, nejen kvůli ušetření času, ale i kvůli budoucí udržitelnosti kódu. Aby bylo možné knihovní (de)serializace využít, musely entitní `data class` splňovat několik podmínek a omezení (uvedena vždy podmínka/omezení následované postupem pro jejich splnění):

- Třída musí mít defaultní konstruktor.
- ▷ Splněno nastavením defaultních hodnot pro parametry konstruktoru, typicky `null`. Mnoho těchto parametrů v doménovém modelu hodnoty `null` (či jiné „prázdné“ hodnoty) nabývat nesmí. Toto bylo vyřešeno ve funkcích mapujících z entit do doménových objektů: funkce vracejí chybovou hodnotu, pokud narazí na prázdný povinný atribut.
- Třída musí mít *mutable*[10] atributy `var`, které musí být pojmenovány stejně jako korespondující pole v dokumentu.
- ▷ Jelikož knihovna není schopná atributy naplnit přes konstruktor, vyžaduje, aby byly mutable. Byly tak tedy definovány. Také byly atributy entitních tříd vyjmuty[11] z obfuskace kódu, která pro *release* variantu sestavení standardně nahrazuje jména tříd i atributů za kratší (úspornější) náhodné názvy.
- Chybějící podpora pro ukládání určitých typů.
- ▷ Problematické typy byly vyměněny za varianty, které knihovna zvládá (de)serializovat. Konkrétně šlo o `BigDecimal`[12] nahrazený za `String`; a `OffsetDateTime` nahrazený za `Timestamp`

[10] „Měnitelné, editovatelné“. Pro atributy `data class` nestandardní, typicky se používají *immutable* (neměnitelné) atributy `val`.

[11] Konkrétně bylo potřeba v souboru `proguard-rules.pro` jde o nastavení `keepclassmembers class` pro balíček obsahující entity. Pro info o použitém obfuskačním nástroji viz [38].

[12] Typ používaný v Javě a Kotlinu pro přesnou matematiku v desítkové soustavě, např. pro finanční částky. Konverze z i do textového řetězce je bezproblémová.

Výpis kódu 3.1 Definice entity PurchaseEntityReadData.

```
data class PurchaseEntityReadData(
    var name: String? = null,
    var payers: List<PayerEntity> = emptyList(),
    var splitting: TransactionSplittingStrategyReadEntity? = null,
    var time: Timestamp? = null,
    var note: String? = null,
) {
    data class TransactionSplittingStrategyReadEntity(
        var type: SplittingType? = null,
        var data: Map<String, Any> = emptyMap(),
    )
}
```

přímo z Firebase knihovny (jde o preferovaný způsob ukládání časových údajů ve Firestore [39]). O konverze se opět starají mapující funkce.

- Chybějící podpora pro deserializaci abstraktních typů.

▷ Pro (de)serializaci nákupů a převodů toto řeší jejich umístění do oddělených kolekcí. Aplikace ale musí být schopna ukládat různé typy dělení nákupů. Toto bylo vyřešeno vytvořením `PurchaseEntityReadData` (kód 3.1) (používanou pro čtení a deserializaci), která na rozdíl od `PurchaseEntityData` (používanou pro vytváření a serializaci) nese dělení transakce jako mapu s klíči typu `String` a hodnotami typu `Any`. Na takovouto mapu nemá knihovna problém převést libovolný objekt z databáze. V `PurchaseEntityData` se spolu s daty pro dělení ukládá i enum `SplittingType`, který se v `PurchaseEntityReadData` načte spolu s mapou dat, a podle hodnoty enumu se následně může mapovací funkce rozhodnout, kterou deserializační funkci[13] volat.

Firestore entity se od doménových objektů liší ještě v několika detailech:

- Reference na jiné objekty ve vztazích M:N (např. vztah účastník–transakce) není realizován zanořením, ale uložením identifikátoru na jedné či obou stranách (pro účastníkovu transakci je v transakci uloženo ID účastníka).

- Firestore knihovna neumožňuje referovat v datovém objektu k jeho ID (při serializaci ani deserializaci), funkce přijímají i vracejí ID a data zvlášť. Taktéž neumožňuje v jednom volání vytáhnout entitu (dokument) zároveň se subkolekcemi daného dokumentu. Obojí může být při práci s entitami v kódu nepříjemné. Proto byly vytvořeny obalující objekty, ve kterých je ID entity, její data, a případně i kolekce dat ze subkolekcí dokumentu entity. V doménových objektech jsou tyto tři kategorie dat na stejné úrovni, takže mapující funkce vytvářející doménové objekty přijímají tyto obalující objekty, ne pouhá data entit.

3.5 Výpočet dlužných částek

Výpočty dlužných stavů jsou v aktuální verzi aplikace realizovány pomocí jednoduchých algoritmů.

[13] Tyto deserializační funkce již nejsou knihovní, ale psané autorkou. Vzhledem ke kompaktnosti dat řešících podíly na nákupech není problém tyto malé funkce udržovat. Veškeré atributy mimo podílů v tomto řešení stále deserializuje knihovní funkce.

3.5.1 Výpočet bilancí v rámci skupiny

Pro možnost zobrazení stavů členů skupiny je proveden jeden průchod historií transakcí. Pro akumulaci slouží mapa, kde je klíčem identifikátor uživatele a hodnotou jeho (zatím nedopočítaný) stav. Před průchodem jsou pro členy nastaveny stavy na nulu. Při průchodem transakcemi se k uživatelovu stavu přičítají částky, které zaplatil, a odečítají se částky, které byly zaplaceny jemu / za něj, převedené na skupinovou měnu.

U výdajů dělených podílovými koeficienty je pro rozumnou přesnost přepočtu na částku vždy provedeno nejdříve násobení podílovým koeficientem a až následně dělení součtem všech podílových koeficientů. Pro drobné usnadnění práce s tímto mechanismem na více místech v kódu má rozpočet částky pomocí podílů na starosti samostatná komponenta, která vrací mapu – klíčem je podílový koeficient, hodnotou jeho korespondující částka.

Prozatím probíhají výpočty v přesnosti dané měny (tzn. pro českou korunu je nejmenší rozlišovaná jednotka jeden halíř).

Do budoucna by bylo možné výpočty zpřesnit: počítat s vyšší přesností a zaokrouhlovat až na samém konci výpočtu, nebo akumulovat pro členy stavy ve všech měnách použitých v transakcích skupiny a provádět měnovou konverzi obdobně v závěru výpočtu. Pro zjištění, zda by některý (či oba) z kroků byl na místě, by bylo možné aplikaci uvolnit reálným uživatelům, provést testovací výpočty nad vzorkem reálně vzniklých skupiny a zjistit odchylky jejich výsledků od aktuálního algoritmu.

Pokud by aplikace v budoucnu měla zobrazovat nejkratší možný seznam plateb k vyrovnání/minimalizaci dluhů ve skupině, bylo by k jeho určení možné využít výstupu tohoto výpočtu. Pokud by však pro seznam plateb nebyla jediným kritériem jeho délka, ale mělo být bráno v potaz, který konkrétní účastník platil za kterého jiného, byl by již nutný jiný, komplexnější přepočet z dat transakcí.

3.5.2 Výpočet uživatelovy bilance

Pro zobrazení karet skupin s uživatelovým stavem v daných skupinách a pro dopočet sum těchto stavů není zapotřebí provádět pro skupiny výpočet pro všechny jejich členy: místo toho je pro skupinu dopočítán jen stav uživatelova účastníka. (Nemá-li uživatel v dané skupině účastníka, je pro ni rovnou vrácena nula.)

Má-li ve skupině účastníka, je pro danou skupinu proveden algoritmus podobný předchozímu, ovšem akumuluje se pouze stav daného účastníka. Tento algoritmus lze případně v budoucnu vylepšit, obdobně jako předchozí. Výsledek je zobrazen na kartě skupiny.

Výpočet součtů dluhů a pohledávek účastníka se následně výsledky z jednotlivých skupin, převedené do účastníkovy měny, podle svého znaménka nasčítají do daných kategorií. Dopočet jejich rozdílu je již triviální.

3.6 Obrazovky

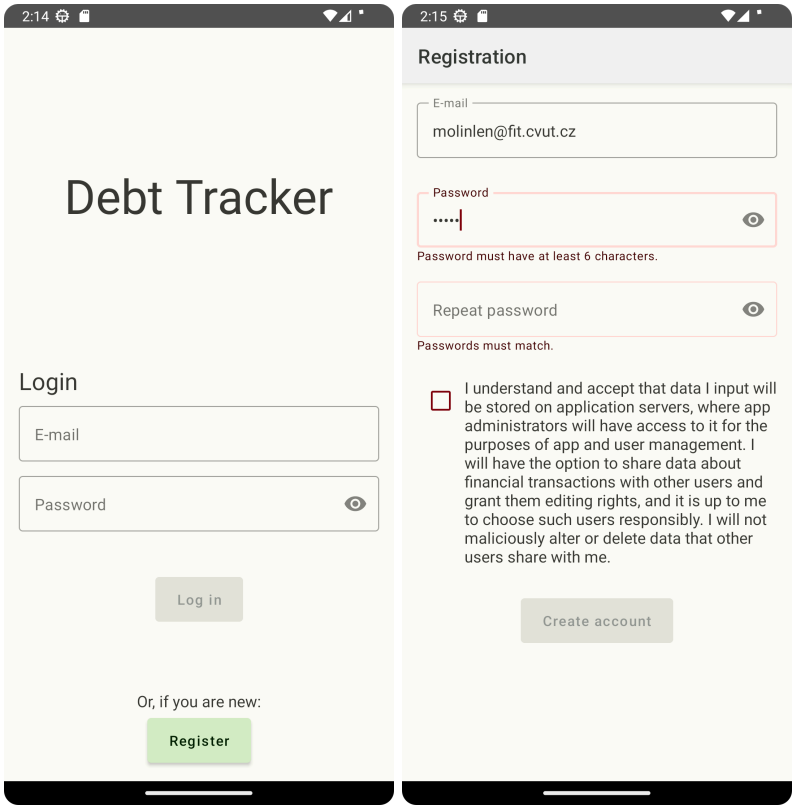
Dále jsou popsány klíčovější obrazovky pro nastínění, jak vypadají detaily implementace.

Obrázky v této kapitole jsou jen k popisovaným částem rozhraní. Více snímků obrazovek aplikace lze najít v příloze A.

3.6.1 Přihlášení a registrace

Obrazovky jsou na obrázku 3.5.

Není-li uživatel ještě přihlášen, zobrazí se mu při spuštění aplikace přihlašovací obrazovka s možností přejít k registraci.



Obrázek 3.5 Přihlašovací a registrační obrazovka.

Na přihlašovací obrazovce může uživatel do textových polí vyplnit e-mail a heslo. Obsah pole s heslem je ve výchozím stavu nahrazován korespondujícím počtem teček, a uživatel má možnost si obsah případně zobrazit pomocí ikonky oka.

U obou polí je přetíženo chování tlačítka „enter“ na klávesnici: v poli e-mailu přesměrovává na níže umístěné pole hesla, v poli hesla spouští přihlášení (u moderních klávesnic se i místo enteru zobrazí tlačítko „další“ resp. „hotovo“). Nechce-li uživatel pro přihlášení využít klávesnice, může ji schovat (systémovým tlačítkem „zpět“ či klepnutím na obrazovku mimo textová pole) a využít tlačítka „přihlásit“ (aktivní pouze, jsou-li vyplněna obě pole).

V průběhu procesu přihlášení je obrazovka zašedlá, zobrazuje načítací kolečko a dočasně nepovoluje uživatelské interakce.

Selže-li přihlášení, zobrazí se pod přihlašovacím tlačítkem červená karta s chybovou hláškou.

Níže je tlačítko „registrovat“, přesměrovávající na registrační obrazovku.

Vyplnil-li uživatel před zmáčknutím „registrovat“ nějaké údaje na přihlašovací obrazovce, bude případný obsah pole s e-mailem předvyplněn do registračního pole pro e-mail, a případný obsah pole s heslem bude předvyplněn do prvního registračního pole pro heslo. Tento detail byl zakomponován, jelikož autorka měla v běžném životě možnost pozorovat mnoho uživatelů pokoušejících se o registraci do různých typů (mobilních i webových) aplikací vyplněním informací do přihlašovací části místo registrační: záměrem je udělat důsledky případné obdobné chyby pro uživatele co nejméně nepohodlné.

Dále je na registrační obrazovce pole pro zopakování hesla, zaškrtávací políčko u provizorního textu pro souhlas s uložením uživatelem zadaných údajů na serveru, a tlačítko pro vytvoření účtu. Položky nevyplněné korektně jsou zvýrazněny červeně, u textových polí i s popiskem chyby, a do korektního vyplnění je tlačítko pro založení účtu neaktivní.

3.6.2 Přehled

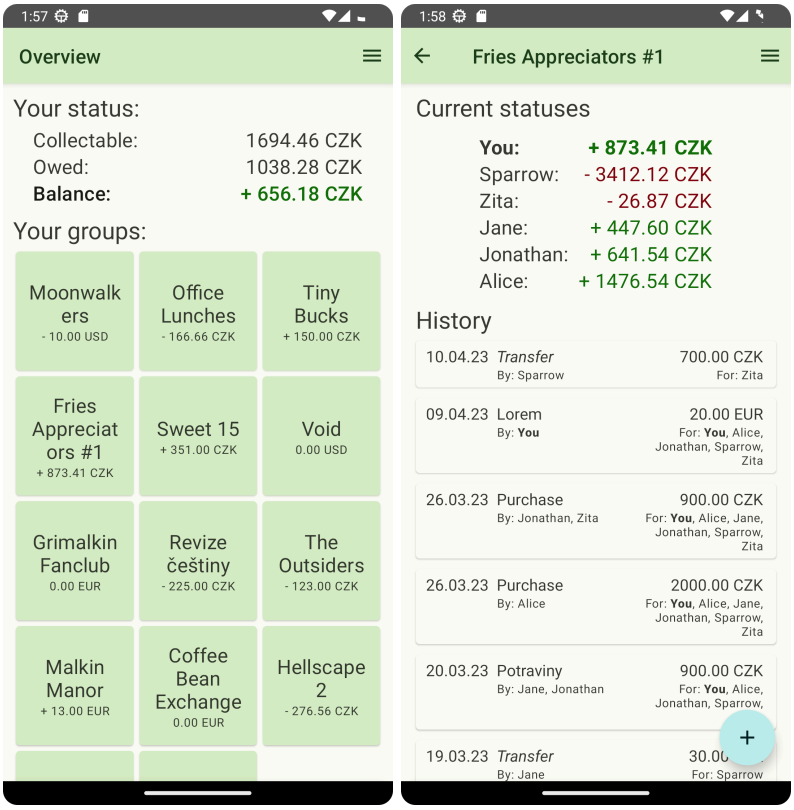
Snímek obrazovky viz 3.6

Spustí-li přihlášený uživatel aplikaci, zobrazí se mu obrazovka Přehledu, která bude (do uživatelova případného odhlášení) vždy „kořenovou obrazovkou v navigačním grafu aplikace“: začne-li přihlášený uživatel na libovolné obrazovce zadávat „zpět“ (přes tlačítko – systémové či zobrazované v levém horním rohu – či pomocí systémových gest), časem skončí na obrazovce Přehledu. Zvolí-li systémovou možnost „zpět“ na obrazovce Přehledu (se zavřeným menu), aplikace se zavře.

V Přehledu jsou karty uživatelových skupin (obsahující jméno skupiny a uživatelův dlužný stav v ní) a součty uživatelových dluhů a pohledávek (nebo text, že může vyplnit svou preferovanou měnu, pokud tak ještě neučinil).

Karty skupin přesměrovávají na obrazovku s detailem dané skupiny.

V menu Přehledu může uživatel provést odhlášení, přejít na informace o účtu, přejít k vytváření nové skupiny, nebo zvolit možnost „připojit se ke skupině“: ta ve spodním panelu schová položky menu a zobrazí textové pole pro zadání ID skupiny, ke které se uživatel chce připojit. Povede-li se připojení, je uživatel přesměrován na detail skupiny.



Obrázek 3.6 Obrazovky Přehled a detail skupiny.

3.6.3 Detail skupiny

Snímek na obrázku 3.6.

Na obrazovce jsou zobrazeny aktuální dlužné stavy členů skupiny (seřazené podle částek) a karty transakcí, seřazené podle data od nejnovější. Jelikož transakcí může být mnoho, je použit `LazyColumn`, který vykresluje vždy jen tu část karet, na které se uživatel kouká, aby zařízení

nebylo zbytečně přetěžováno [40]. Nemá-li skupina ještě členy, zobrazí se místo prázdného prostoru hláška, že uživatel může členy přidat.

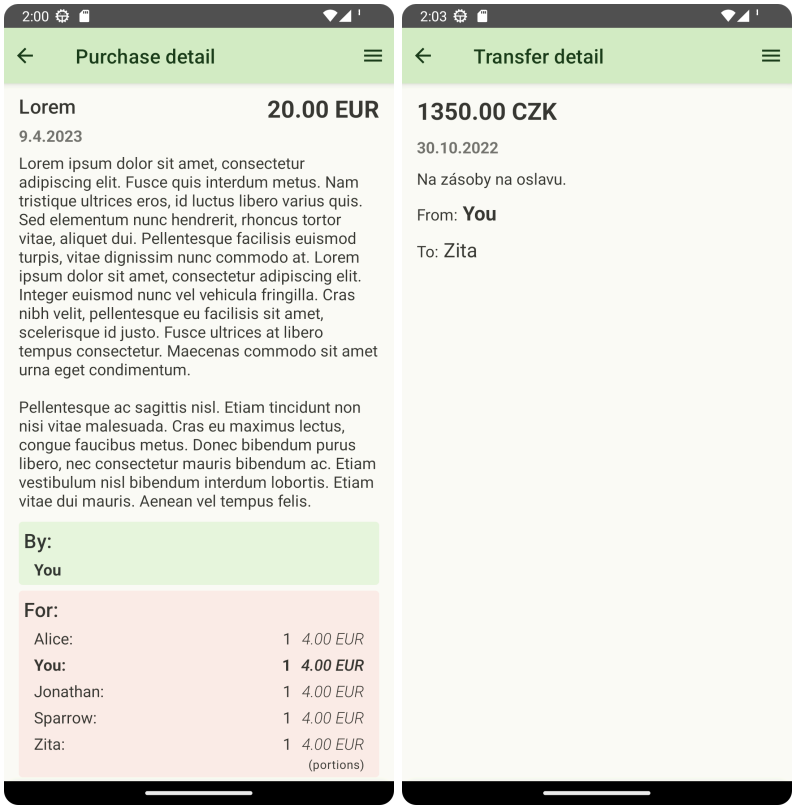
Na kartách transakcí jsou seznamy členů skupiny: platících danou transakci a „přijímající“ danou transakci. Výskyt uživatelova člena je zvýrazněn tučně; karty transakcí jsou jinak relativně minimalistické, takže toto zvýraznění dostatečně upoutává pozornost při prohlížení, hledá-li uživatel své transakce, a zároveň zbytečně vizuálně neruší.

V menu má uživatel možnost přejít na obrazovky „informace o skupině“ a „členové skupiny“.

Proklik z karty transakce vede na její detail.

Floating action button s ikonou plus po kliknutí nabídne uživateli ve spodním panelu možnost vybrat, který typ transakce chce začít vytvářet: převod peněz či výdaj. *Floating action button* se však nezobrazuje, pokud ve skupině není vyplněn alespoň jeden člen.

3.6.4 Detail transakce



Obrázek 3.7 Obrazovky detailu výdaje a převodu.

3.6.4.1 Detail výdaje

Snímek na obrázku 3.7.

Obrazovka detailu výdaje uživateli prezentuje všechny informace o výdaji, ale snaží se ho nezahlcovat opakováním jedné informace vícekrát.

Vždy jsou zobrazeny povinné atributy výdaje: datum, celková částka, informace o plátcích a o členech s podílem. Volitelné atributy název a poznámka jsou zobrazeny, jen jsou-li vyplněné.

Je-li plátce či uživatel s podílem jen jeden, zobrazí se v korespondující sekci jen jeho přezdívka bez opakování informace o částce: je zjevné, že jde o celkovou částku výdaje, která již na obrazovce je.

Je-li plátců více, zobrazuje se u každého zaplacená částka.

Je-li výdaj rozdělení konkrétními částkami, zobrazují se u členů ony. Je-li dělený podílovými koeficienty, zobrazí se u členů koeficienty (normálním fontem) a napravo od nich i výsledné částky podílu (kurzívou v tenčím fontu).

Vyskytuje-li se uživatel mezi plátci či mezi členy s podílem, je jeho řádek tučně zvýrazněn.

Uživatel má v menu možnost přejít k úpravám výdaje, nebo výdaj smazat – což případně potvrzuje v dialogu.

3.6.4.2 Detail převodu

Snímek viz obrázek 3.7.

Styl textu položek společných pro výdaj a převod (celková částka, datum, poznámka) jsou shodné jako na detailu výdaje, ač se liší pozicování elementů: celková částka je zde vlevo, ne vpravo.

Místo barevných karet jsou zde účastnící se členové uvedeni jen s popisky „od“ a „pro“. Případný výskyt uživatelova člena je standardně zvýrazněn.

V menu má uživatel možnost využít obdobných akcí jako na detailu výdaje: přechod k úpravám či smazání.

3.6.5 Vytváření transakce

Případné úpravy transakcí probíhají na stejných obrazovkách – jen jsou předvyplněna data upravované transakce, a je lehce upravena textace (nadpis obrazovky, text na tlačítku pro uložení).

3.6.5.1 Vytváření převodu

Viz obrázek 3.8 (levá část).

Díky malému množství informací nutných pro zaevidování převodu peněz dvěma účastníky je obrazovka velmi jednoduchá.

Uživatel do textového pole vepíše částku (pole se otevře s číselnou klávesnicí, nedovolí vložit nečíselnou, zápornou nebo nesmyslnou hodnotu, a označuje za nevalidní situaci, kdy je vyplněna nulová částka).

Zvolí, od kterého a ke kterému členovi skupiny převod proběhl pomocí rozbalujících se seznamů (výskyt uživatele mezi členy je standardně zvýrazněn). Kdyby se pokusil obě pole nastavit na stejnou hodnotu, hodnota ve dříve upraveném poli zmizí: nelze vyplnit stejného člena do obou kolonek.

Uživatel má možnost vybrat jinou měnu (výchozí hodnota je měna skupiny). Může pomocí nativní kalendářové komponenty upravit datum (které je přednastaveno na aktuální den). Má možnost vyplnit pole „poznámka“.

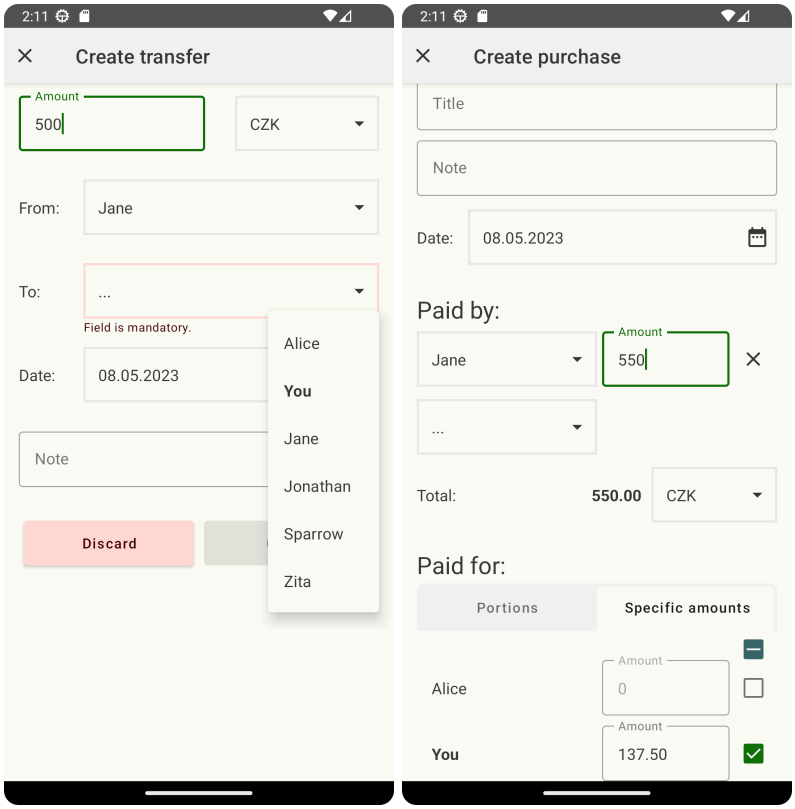
Převod může uložit pomocí tlačítka „vytvořit“.

3.6.5.2 Vytváření výdaje

Obrázky viz 3.8 (napravo) a 3.9.

Na začátku obrazovky výdaje má uživatel možnost vyplnit název a poznámku a upravit datum.

Níže vybírá plátce: z rozbalujícího seznamu musí vybrat alespoň jednoho člena a doplnit k němu, kolik platil. Může plátců zvolit více (maximálně všechny členy skupiny). Plátce může i odebírat křížkem na konci daného řádku.



Obrázek 3.8 Obrazovka vytváření převodu; první část obrazovky vytváření výdaje.

Výdaj má vždy právě jednu měnu, kterou lze upravit vedle celkové zaplacené částky.

Níže je nejkomplexnější část obrazovky: určení dělení výdaje mezi členy. Uživatel může přepínat mezi dělením podílovými koeficienty a konkrétními částkami: při přepnutí se zachovávají vyplněné údaje, a záložky si vzájemně nepřepisují částky. Mezi záložkami je však shodné pořadí členů a jejich zaškrtnutí. Předvybrána je varianta s podílovými koeficienty.

Zaškrtávací políčko slouží k zařazení či vyřazení člena pro dělení: odškrtnutí členové nebudou mít na výdaji podíl. Taktéž ho nebudou mít členové, kterým ho uživatel nastaví na nulový, ale toto manuální nastavování je časově náročné a tudíž nepohodlné. Nad zaškrtávacími políčky pro jednotlivé řádky je zaškrtávací políčko umožňující zaškrtnutí a odškrtnutí všech políček.

Pro dělení konkrétními částkami uživatel může manuálně vyplnit částky, nebo využít tlačítka „rozdělit rovnoměrně“. Tolerance nepřesnosti je zde počet zaškrtnutých členů krát nejnižší možná jednotka měny. Kolik uživateli zbývá – nebo přebývá – k rozdělení ukazuje případně červený text pod seznamem podílů.

Pro dělení podílovými koeficienty jsou předvyplněny koeficienty na jedna. Uživatel je může upravovat na libovolné nezáporné celé číslo do jednoho tisíce včetně. Na řádku se zobrazuje i přepočet aktuálního koeficientu na částku.

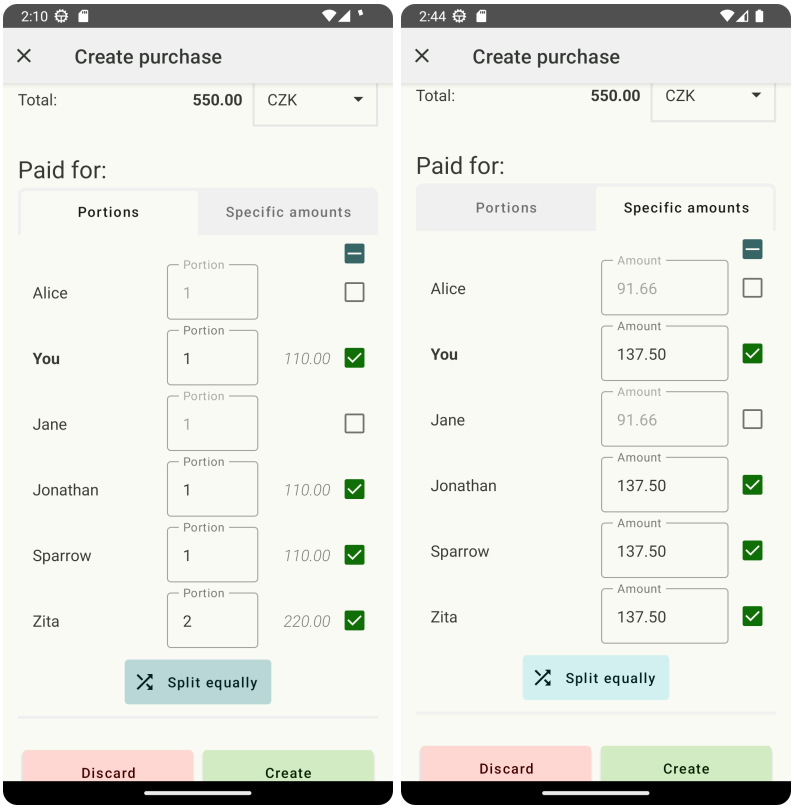
Uživatel může výdaj uložit pomocí tlačítka „vytvořit“.

3.6.6 Členové skupiny

Obrázek viz 3.10.

Obrazovka sloužící pro správu členů skupiny.

Existující členové jsou řazeni abecedně podle přezdívek.



Obrázek 3.9 Pokračování obrazovky vytváření výdaje: verze pro podílové koeficienty a konkrétní částky.

Uživatel může členy přidávat: po kliknutí na *floating action button* s ikonou plus se otevře spodní panel s textovým polem pro přezdívku nového člena a tlačítky „přidat“ a „zrušit“. Přidávací tlačítko je aktivní, jen pokud vyplněná přezdívka není prázdná a není shodná s přezdívkou nějakého již existujícího člena dané skupiny.

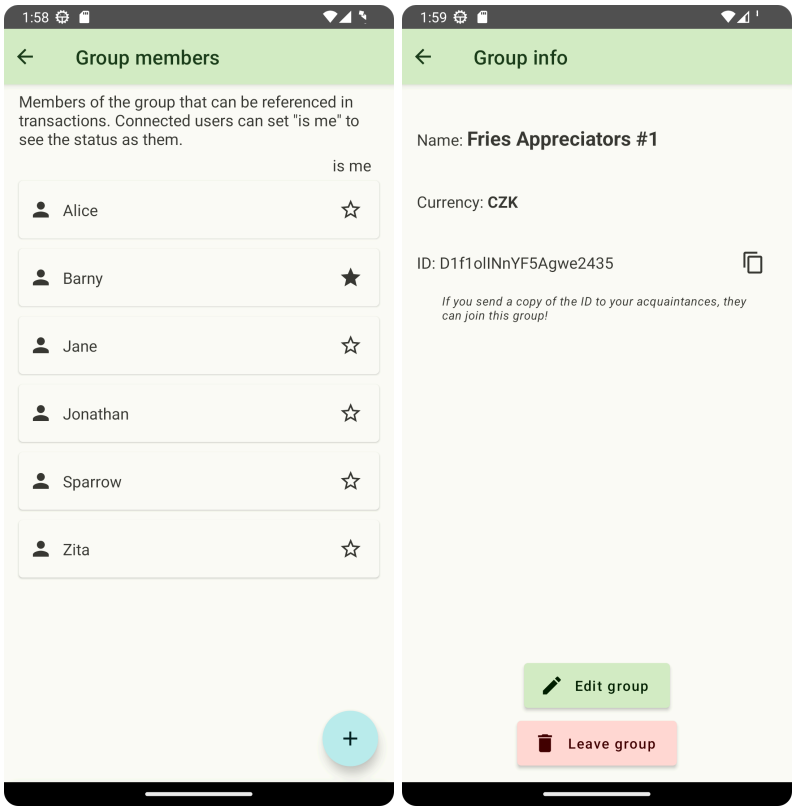
Uživatel také může pomocí ikony hvězdy na kartách členů přepínat, jako který člen vystupuje (také může přiřazení zrušit a nevystupovat jako žádný z členů).

3.6.7 Informace o skupině

Snímek viz 3.10.

Jednoduchá obrazovka agregující možnosti skupiny, u kterých se očekává méně časté využití uživateli, než u funkcionalit dostupných z detailu skupiny. Uživatel zde má:

- Možnost vykopírovat ID skupiny pro sdílení – pomocí kopírovacího tlačítka, či vybráním textu dlouhým podržením a kopírováním přes kontextové menu.
- Možnost přejít k úpravě parametrů skupiny – názvu a hlavní měny.
- Možnost skupinu opustit – uživatel po zmáčknutí daného tlačítka případně záměr potvrzuje v dialogu, podobně jako u mazání transakce.



Obrázek 3.10 Obrazovky se členy skupiny a informacemi o skupině.

Kapitola 4
Testování

4.1 Vývojářské testování

Aplikace byla průběžně manuálně testována autorkou. Byl kladen důraz na ověření korektního fungování celé aplikace. Také byla posuzována přehlednost grafického rozhraní.

Zpočátku byla většinou vývojářská testování prováděna nad *debug* sestavením, s blížícím se ukončením implementační fáze byla častěji testována i *release* verze. *Debug* sestavení aplikace je značně rychlejší, neboť při něm nedochází ke zmenšování a obfuskaci kódu. *Release* varianta sestavení je však to, co bude zpřístupněno uživatelům, bylo tedy nutné i u této varianty ověřit plnou funkčnost.

K testování byla primárně využita zařízení v tabulce 4.1 (jedno fyzické a dvě emulovaná). Nejčastěji byla používána zařízení Galaxy a Pixel, jako zástupci relativně běžných zařízení v době vývoje. Emulovaný Nexus byl využit jednou za čas k ověření funkčnosti aplikace na zařízení, které má jednak nižší verzi Android OS, jednak velmi malý displej (viz tabulka 4.1) – cílem tohoto testování bylo ověření plné funkčnosti aplikace na zařízení s těmito limitujícími faktory.

Unit (jednotkovými) testy byly pokryty části kódu zodpovědné za přepočty dlužných částek, jelikož je málo pravděpodobné, že by byly v blízké době příliš měněny, a manuální testování korektních přepočtů by bylo velmi pracné. Zbylé části nepochybně do budoucna stojí za to také automatizovatelnými testy pokrýt, ale jejich vývoj je v plánu ponechat až po první vlně uživatelského testování (a tedy i po závěru této práce). Při uživatelském testování totiž může být zjištěna nutnost přepisu části aplikace (např. nutnost přesunu funkcionalit mezi obrazovkami), bude tedy výhodnější další testy psát, až bude známé, které části aplikace uživatelům v aktuální podobě vyhovují.

Tabulka 4.1 Zařízení použitá k vývojářskému testování.

Název	Android (API)	Displej [palce]	Rozlišení [pixely]	Fyzické/ emulované
Samsung Galaxy A20e	11 (30)	5,8	720 × 1560	fyzické
Google Pixel 3a	13 (33)	5,6	1080 x 2220	emulované
Google Nexus One	5 (22)	3,7	480 x 800	emulované

4.2 Testování s uživateli

Počet testujících uživatelů nemusí být zvlášť velký na to, aby nalezl většinu problémů s přehledností a použitelností aplikace, obzvlášť u menšího projektu (což vzniklá aplikace zatím rozhodně je) [41]. Testerů bude tedy pět, z nich čtyři uživatelé Androidu (primární cílová skupiny aplikace) a pro kontrolu jeden uživatel iOS.

Čeho naopak nebude málo, jsou testovací kroky. Testeři si projdou všechny běžné uživatelské akce v aplikaci (i pár potenciálně vzácnějších), aby byla otestována použitelnost celé aplikace.

Testující uživatelé nebudou jen upozorňovat na problémy a nepohodlí, na které narazí v průběhu testovacích kroků. Budou mít možnost po testování aplikaci dále zkoumat a budou požádáni i o upozornění na potenciálně problémové aspekty aplikace nespadající do testovacích sad, o vlastní návrhy na vylepšení aplikace, a o celkové hodnocení její aktuální podoby.

4.2.1 Sady testů

Zvolené sady testů sdružují testovací kroky zaměřené na související případy užití.

Kroky sad pokrývají všechny aktuálně podporované uživatelské akce v aplikaci, aby byla otestována pohodlnost užívání všech částí uživatelského rozhraní. Jediným případem užití, který bude v testování lehce ochuzen, bude úprava existující transakce: samotné úpravy odehrávají na stejné obrazovce, jako vytváření transakce, obrazovka bude podrobně otestována při vytváření transakcí.

Dále jsou sepsány kroky testovacích sad.

4.2.1.1 Sada: Management účtu

Kroky:

1. Registrace.
   - Očekávaný průběh: Uživatel z přihlašovací obrazovky přejde na registrační obrazovku pomocí tlačítka „registrovat“, vyplní údaje a klikne na tlačítko „založit účet“.
   - Cíl: Ověřit přehlednost jak samotné registrační obrazovky, tak toho, zda testeři budou mít problém se na ní dostat. Registrační obrazovka navíc předvyplní údaje již zadané na přihlašovací obrazovce: toto mohou testeři potenciálně hodnotit jako nápomocné či matoucí.
2. Vyplnění preferované měny.
   - Očekávaný průběh: Uživatel v menu na obrazovce Přehled vybere „účet“, na obrazovce účtu klikne na tlačítko s ikonou tužky vedle preferované měny a na následující editační obrazovce měnu změní.
   - Cíl: Ověřit, zda uživatel úspěšně zjistí, kde najde detaily o účtu, a zda pro něj zbytek procesu bude přehledný a pohodlný.
3. Odhlášení.
   - Očekávaný průběh: Uživatel v menu na obrazovce Přehled vybere „odhlásit“.
   - Cíl: Ověřit, zda uživatel úspěšně zjistí, kde najde možnost odhlášení.
4. Přihlášení.
   - Očekávaný průběh: Uživatel na přihlašovací obrazovce vyplní e-mail a heslo, ideálně mezi danými poli přepne pomocí tlačítka klávesnice „další“, které by se mělo zobrazit místo tlačítka pro nový řádek. Následně potvrdí (tlačítkem klávesnice nebo tlačítkem „přihlásit“ na obrazovce) a bude přihlášen.
   - Cíl: Ověřit, zda je přihlašovací obrazovka přehledná a pohodlná na užívání.

4.2.1.2 Sada: Management přístupů ke skupinám

Kroky:

1. Vytvoření nové skupiny.
   - Očekávaný průběh: Uživatel v menu na obrazovce Přehled vybere „vytvořit novou skupinu“, na následující obrazovce vyplní název a měnu skupiny a potvrdí. Skončit by měl na detailu nově vytvořené prázdné skupiny.
   - Cíl: Ověřit, zda uživatelé snadno najdou, kde začít skupinu vytvářet, a zda nebudou mít problémy s obrazovkou pro vytvoření skupiny.
2. Připojení ke skupině přes její ID.
   - Očekávaný průběh: Uživatel v menu na obrazovce Přehled vybere „připojit se ke skupině“. Obsah spodního panelu se změní z menu na textové pole a tlačítko „připojit“. Do pole uživatel zadá ID skupiny, a potvrdí přes klávesnici či tlačítko.
   - Cíl: Ověřit, zda uživatelé snadno najdou možnost připojení ke skupině, a zda nebudou mít problém ovládat elementy k tomu určené.
3. Sdílení ID skupiny jinému člověku (přes libovolnou chatovací aplikaci).
   - Očekávaný průběh: Uživatel na detailu dané skupiny z menu vybere „Informace o skupině“. Tím se dostane na obrazovku s mj. ID skupiny, které může vykopírovat pomocí tlačítka s ikonou pro kopírování, či vybráním daného textu (dlouhým podržením) a kliknutím na kontextovou možnost „kopírovat“, která se objeví. Následně přejde do chatovací aplikace a ID vloží do zprávy.
   - Cíl: Ověřit, zda uživatelé nebudou mít problém najít, kde se ID nachází, a zda nebudou mít problém ho zkopírovat.
4. Opuštění skupiny.
   - Očekávaný průběh: Uživatel na detailu dané skupiny z menu vybere „Informace o skupině“. Na následující obrazovce klikne na tlačítko „opustit skupinu“. Zobrazí se potvrzovací dialog, kde uživatel klikne na „ano“, a bude přesměrován na seznam skupin (již bez právě opuštěné skupiny).
   - Cíl: Ověřit, zda uživatelé snadno naleznou tlačítko „opustit skupinu“ a nezmate je potvrzovací dialog.

4.2.1.3 Sada: Management dat a transakcí skupiny

Kroky:

1. Přidání *lokálních členů* skupiny a označení sebe mezi nimi.
   - Očekávaný průběh: Uživatel na detailu skupiny z menu vybere „členové skupiny“. Na následující obrazovce klikne na tlačítko s ikonou plus. Objeví se spodní panel s textovým polem pro přezdívku a tlačítkem „přidat“. Uživatel vyplní přezdívku či jméno do pole a potvrdí tlačítkem. Obrazovka se na okamžik zatmaví s indikátorem zpracovávání[1], následně se nový člen objeví v seznamu. Uživatel postup zopakuje pro další členy. „Svého“ člena vybere kliknutím na ikonku hvězdičky na kartě člena, která se pro potvrzení vyplní barvou.

[1] Všemi oblíbené „točící se kolečko.“

- Cíl: Ověřit, zda uživatelé snadno najdou obrazovku pro úpravy členů skupiny, a zda je pohodlné danou obrazovku používat.

2. Vytvoření výdaje s dělením vahami.
   - Očekávaný průběh: Uživatel na detailu skupiny klikne na tlačítko s ikonou plus. Objeví se spodní panel, ze kterého uživatel vybere typ „nákup“. Na následující obrazovce vyplní parametry nákupu, platícího účastníka a kolik daný zaplatil. V sekci obrazovky „zaplaceno za“ odškrtne pomocí zaškrtávacích políček účastníky, kteří na nákupu podíl mít nemají, a u zbylých vyplní podíly. Poté uživatel klikne na tlačítko „vytvořit nákup“.
   - Cíl: Ověřit, zda uživatel snadno najde možnost přidání výdaje, a zda se mu budou snadno vyplňovat údaje na obrazovce pro vytváření výdaje (s variantou dělením podílem).
3. Vytvoření výdaje s více plátci a dělením konkrétními částkami.
   - Očekávaný průběh: Začátek obdobný jako u předchozího scénáře. Tentokrát ale uživatel vyplní více plátců, včetně částek, které zaplatili. V sekci obrazovky „zaplaceno za“ přepne do varianty zadávání konkrétních částek a částky vyplní (potenciálně za použití tlačítka „rozdělit rovnoměrně“; mezi poli pro podíly jednotlivých účastníků může přepínat pomocí klávesnicové volby „další“). Poté klikne na tlačítko „vytvořit nákup“.
   - Cíl: Ověřit, zda uživatel nebude mít problém přepnout do varianty zadávání dělení konkrétními částkami, a zda se mu budou snadno vyplňovat údaje pro tuto variantu dělení.
4. Vytvoření převodu peněz mezi účastníky.
   - Očekávaný průběh: Uživatel na detailu skupiny klikne na tlačítko s ikonou plus. Objeví se spodní panel, ze kterého uživatel vybere typ „převod peněz“. Na následující obrazovce vyplní parametry převodu a klikne na „vytvořit převod“.
   - Cíl: Ověřit, zda uživatel snadno najde možnost přidání převodu peněz, a zda se mu údaje na obrazovce budou snadno vyplňovat.
5. Úprava dat transakce.
   - Očekávaný průběh: Uživatel na detailu skupiny klikne na kartu dané transakce. Bude přesměrován na detail dané transakce. Zde v menu vybere možnost „upravit“. Bude přesměrován na obrazovku pro úpravu daného typu transakce (výdaje či převodu), kde upraví požadovaný parametr a klikne na „uložit změny“.
   - Cíl: Ověřit, zda uživatelé snadno najdou možnost upravit transakci.
6. Změna skupinové měny.
   - Očekávaný průběh: Uživatel na detailu skupiny z menu vybere „informace o skupině“. Na následující obrazovce klikne na tlačítko „upravit“. Na další obrazovce vybere novou měnu a klikne na „uložit změny“.
   - Cíl: Ověřit, zda uživatel snadno najde, kde upravit parametry skupiny, a zda je obrazovka k tomu určená snadno ovladatelná.
7. Interpretace aktuálního stavu dluhů ve skupině.
   - Očekávaný průběh: Uživatel po pohledu na seznam dlužných stavů na detailu skupiny dojde k závěru, že členové skupiny se záporným stavem mají zaplatit částku, která je u nich uvedená, a že členové s kladným stavem by měli dostat obnos, který je uveden u nich. U menší skupiny (4–5 členů) bude uživatel schopen vyjmenovat seznam plateb, který by vedl ke značnému snížení dluhů ve skupině (bude moci zaokrouhlovat – proto snížení, ne eliminaci dluhů).

Tabulka 4.2 Zařízení a barevný mód testujících uživatelů (a jejich primární OS).

Tester	Zařízení	Verze Android	Barevný mód	Primární OS testera
Tester K	Xiaomi 12X	13	tmavý	Android
Tester J	Xiaomi POCO M3	10	světlý	Android
Tester C	Xiaomi Redmi Note 8 Pro	11	tmavý čtecí	Android
Tester N	Samsung Galaxy A50	11	světlý	Android
Tester Z	Samsung Galaxy A50	11	světlý	iOS

- Cíl: Zjistit, zda je aktuální zjednodušená prezentace dlužných částek pro uživatele srozumitelná a dostatečná. U tohoto kroku je větší šance pro neúspěch, případné neúspěchy budou využity k diskuzi s uživateli nad tím, jaká prezentace dlužných stavů by jim vyhovovala více.

4.2.2 Testující uživatelé

Testování probíhalo odděleně mimo testerů N a Z, které v domluveném termínu testování nebylo možné (v důsledku nedostatku možných alternativních testovacích prostor) oddělit. Důvodem pro preferenci odděleného testování bylo snížení potenciálního vzájemného ovlivňování testerů. V absenci jiných lidí by testeři mohli přinést ostřejší kritiku a více se ponořit do popisu případných možných vylepšení aplikace. Navíc je individuální testování mnohem méně náročné na rozvrh testerů, a tedy nižší šance, že by zkracovali popis nedostatků aplikace, aby ušetřili čas. Nevýhodou oddělení testerů je, že testeři své návrhy na vylepšení aplikace nemohli vzájemně validovat, např. podpořit možnost, která je samotné nenapadla.

Uživatelé testovali aplikaci ve světlém či tmavém režimu dle vlastní preference a zvyku.

Testovali na vlastních zařízeních mimo testera Z. Tester Z primárně využívá zařízení s iOS systémem, je tedy očekávané, že nebude zvyklý na Android komponenty a schémata navigace.

Přehled zařízení a barevných voleb testerů je v tabulce 4.2 (ve všech tabulkách v kapitole jsou testeři uvedení ve stejném pořadí).

Tester C je pravidelným uživatelem Settle Up, tester K příležitostným uživatelem stejné aplikace. Zbytek testerů žádnou obdobnou aplikaci nevyužívá

4.2.3 Výsledky testů

Pro přehledné shrnutí v tabulce jsou výsledky testovacích kroků – z hlediska uživatelské intuitivity – u jednotlivých uživatelů rozděleny do těchto kategorií:

✓ Uživatel krokem plynule prošel očekávanou cestou, nevykazoval žádné nepohodlí.

□ Uživatel zvládl krok splnit po menším zmatení či váhání.

× Uživatel dlouho v rozhraní hledal, váhal, či byl zmatený.

Pro označení technické chyby při testu je případně doplněn symbol ⊗ .

Výsledky jsou shrnuty v tabulce 4.3 – popisky testovacích kroků jsou pro přehlednost zkrácené, pro plný popis případně viz 4.2.1 Sady testů.

Technické chyby nastaly jen ve dvou případech, tedy 2,67 %. Stejný byl výskyt vážných uživatelských problémů. Drobné uživatelské problémy měly častější výskyt: 22 případů, 29,33 %. Bez žádných problémů s uživatelskou přívětivostí bylo 51 provedení testovacích kroků: 68 %.

Níže budou popsány detaily nalezených problémů a podněty testerů.

Tabulka 4.3 Výsledky testů aplikace uživateli.

Testovací krok	Tester K	Tester J	Tester C	Tester N	Tester Z
Registrace	☐	☐	✓	✓	☐
Vyplnění preferované měny	☐	☐	✓	✓	✓
Přihlášení	✓	✓	✓	✓	☐
Odhlášení	✓	✓	✓	✓	✓
Vytvoření skupiny	✓	✓	✓	☐	✓
Připojení ke skupině	✓	☐	✓	✓	✓
Sdílení ID skupiny	✓	☐	✓	✓	✓
Opuštění skupiny	✓	✓	✓	✓	✓
Přidání členů skupiny	☐ ⊗	☐	☐	☐	✓
Výdaj dělený podíly	☐	☐	✓	✓	×
Výdaj dělený konkrétně, více plátců	✓	✓	✓	☐	✓
Vytvoření převodu peněz	✓	✓	✓	✓	✓
Úprava transakce	☐	☐	☐	✓	✓
Změna skupinové měny	✓ ⊗	✓	✓	✓	☐
Interpretace dlužných stavů	☐	☐	×	✓	✓

4.2.3.1 Nalezené technické problémy

Na zařízení testera K byly problémy s chováním navigace.

Při přidávání členů do skupiny se několikrát stalo, že po přidání člena byl uživatel přesměrován na detail skupiny (ač měl zůstat na obrazovce „členové skupiny“). Zde se problém objevoval nekonzistentně: jednou se uživateli povedlo přidat čtyři členy a stále byl na obrazovce „členové skupiny“, jindy byl přesměrován i po přidání jediného člena.

Při úpravě měny skupiny byl tester po uložení přesměrován opět na obrazovku pro úpravy skupiny (ale změna měny se korektně propsala).

Při obou problémech zůstávaly všechny obrazovky i v historii, když tedy tester po výskytu problému klikal na systémové tlačítko „zpět“, prošel například sekvencí „detail skupiny“ – „členové skupiny“ – „detail skupiny“ – „členové skupiny“ – „detail skupiny“, a sekvencí „editace skupiny“ – „informace o skupině“ – „editace skupiny“ – „informace o skupině“.

Na jiném zařízení se žádný podobný navigační problém nepovedlo replikovat. Pro odstranění by bylo vhodné na tento model zařízení nahrát aplikaci obohacenou o specifické logovací výpisy a pokusit se zjistit detaily (včetně toho, zda se chování objevuje i v *debug* verzi sestavení aplikace). Dalším potenciálním faktorem je, že dané zařízení dostalo aktualizaci OS jen jeden den před provedeným testem: je zde i drobná šance, že chování není způsobeno aplikací, ale chybou v nové verzi OS.

Tester K je ochotný v budoucnu své zařízení k této investigaci zapůjčit, ovšem díky jeho časové vytíženosti bude investigaci nutné provést až po skončení této práce.

4.2.3.2 Nalezené problémy s uživatelskou přívětivostí z testovacích sad

Problémy jsou rozděleny podle testovacího kroku, ve kterém se vyskytly.

- Registrace.
  - Tester Z vyplnil informace na přihlašovací obrazovce a zmáčkl tlačítko „přihlásit“. Po přečtení chybové hlášky si však rychle uvědomil, kde nastala chyba, a ocenil, že se vyplněné údaje předvyplnily i na registrační obrazovce.

  - Tester J vyplnil krátké heslo a chvíli musel zkoumat, proč je tlačítko pro vytvoření účtu neaktivní, než si všiml, že pole pro heslo je červené a pod ním je o délce hesla hláška.
  - Tester K prošel první registrací s jedinou stížností: textové pole pro e-mail není tak pro systém označeno, takže klávesnice nezačne uživatelovy e-maily nabízet sama. Dále při zkoumání aplikace po testovacích krocích také zjistil, že chybová hláška při pokusu o vytvoření účtu k již registrovanému e-mailu není příliš nápomocná, jednak si stěžoval, že předvyplnění údajů z přihlašovací obrazovky je matoucí. Ohledně posledního poznatku by však mělo být zváženo, že testeři Z, J i N tuto vlastnost (patrně neplánovaně) využili a ocenili.

- Vyplnění preferované měny.
  - Tester J nejdříve krátkou chvilku hledal ve skupinovém menu, ale relativně rychle přešel na správnou obrazovku a akci bez problémů dokončil.
  - Testerovi K nějakou chvíli trvalo vymyslet, kde by informace o účtu mohl hledat, dle vlastních slov je zvyklý, že má-li v aplikaci účet, vidí v aplikaci neustále nějakého svého avatara/iniciálu, na které lze kliknout. Samotné vyplnění po nalezení položky v menu již proběhlo bez problémů.
- Přihlášení.
  - Tester Z chtěl po vyplnění e-mailu k poli pro heslo doscrollovat, což nešlo.
- Vytvoření skupiny.
  - Tester N nejdříve v menu vybral možnost „připojit se ke skupině“ a byl chvíli zmaten. Pak mu došlo, co se stalo, a s dokončením již neměl problémy.
- Připojení ke skupině.
  - Tester J chtěl po vložení ID skupiny do textového pole zmáčknout tlačítko pro připojení ve spodním panelu (místo využití enteru klávesnice), klikl mimo spodní panel, čímž panel zavřel (a tím pádem přišel o vyplněné ID). Při dalším pokusu již prošel v pořádku. Pro mitigaci tohoto problému by zde mohlo postačit lehké zúžení textového pole a umístění tlačítka „Připojit“ napravo od pole místo pod něj: uživatelé zvyklí interagovat s tlačítky na obrazovce místo klávesnicových by neměli vůbec potřebu klávesnici schovávat.
- Sdílení ID skupiny.
  - Tester J nejdřív využil ke zkopírování kopírovací tlačítko, ale díky tomu, že po kopírování se nezobrazuje žádná hláška, předpokládal, že tlačítko nezafungovalo. Proto poté kopíroval ID dlouhým podržením. Mimo hlášky o povedeném zkopírování by ocenil přímo tlačítko pro sdílení, aby nemusel přepínat aplikace.
- Přidání členů skupiny.
  - Tester J (stejně jako u sdílení ID skupiny) při pokusu o zavření klávesnice zavřel celý dolní panel a přezdívku prvního člena tak musel vyplnit znovu. Další problémy již neměl.
  - Testeři K a N by ocenili, kdyby pro skupinu bez členů byl upraven úvodní text obrazovky „členové skupiny“, a ideálně např. velkou šipku k tlačítku s plusem, aby tlačítko nemuseli hledat.
  - Tester C by ocenil, kdyby byla jeho přezdívka ve skupině předvyplněná, a raději než nutnost vypisování by ocenil např. možnost akumulace „seznamu přátel“ ve skupině, ze kterých by mohl část členů vybrat. „Seznam přátel“ by prý přišel užitečný i testerům N a Z (tito dva tak indikovali až na konkrétní dotaz, po testech a vlastních připomínkách).

  - Při označování vlastního člena skupiny navíc v aplikaci chybí indikace prováděné akce. Na pomalou vizuální reakci označování si postěžovali testeři K, N i C. Testeři K a N uvedli, že při přítomnosti indikace by nejspíš s rychlostí označení problém neměli (vadilo jim, že nebylo jasné, že aplikace jejich požadavek zaregistrovala). Testerovi C by prý označení i tak přišlo pomalé.

- Výdaj dělený podíly.
  - Testeři Z a J si nevšimli možnosti odškrtávat účastníky pomocí zaškrtávacích políček, vyplňovali tedy manuálně nuly pro označení, kteří účastníci na nákupu podíl mít nemají. Tester Z ani nevyužíval možnosti posunout se na další pole přes tlačítko klávesnice, a v průběhu ze zvědavosti klepl na tlačítko „rozdělit rovnoměrně“, díky čemuž nuly následně vyplňovat znovu.
  - Tester K si sice individuálních políček všiml, ale nevšiml si políčka nad nimi, určeného pro o(d)značení všech účastníků, a tak si stěžoval na nemožnost účastníky najednou odznačit.
  - Po testování byli testeři Z, J i K dotázání, zda jim daná zaškrtávací políčka přijdou příliš nevýrazná, přesunuli by je, zvětšili...? Ani jednoho nenapadlo, jaké vizuální zvýraznění by ho na políčka upozornilo, ale shodli se, že když už o políčkách ví, přijdou jim na použití pohodlná. Pro první spuštění obrazovky pro vytváření výdaje by tedy patrně bylo vhodné spustit krátký tutoriál, který by uživatele na oba typy políček upozornil.

- Výdaj dělený konkrétně, více plátců.
  - Tester N podotkl, že zobrazovaná informace „kolik ještě musí rozdělit“, je nápomocná, ale musí zavřít klávesnici, aby jí mohl vidět.

- Úprava transakce.
  - Testeři K a J se nejdříve k možnosti editace pokoušeli dostat podržením karty transakce v detailu skupiny. Následně se však oba bez problémů vydali správnou cestou.
  - Tester C hledal možnost editace relativně dlouho, prý čekal, že „nad transakcí bude stejné menu jako nad detailem skupiny“.

- Změna skupinové měny.
  - Tester Z měl problém nalézt možnost editace skupiny, prý nečekal, že na obrazovce nazvané „informace o skupině“ bude mít možnost něco editovat.

- Interpretace dlužných stavů.
  - Všichni testeři interpretaci zvládli, ale K a J ji považovali za lehce nepohodlnou a C za příliš nepohodlnou. Všichni tři by preferovali zobrazení seznamu plateb nutných k vyrovnání.

4.2.3.3 Problémy s uživatelskou přívětivostí nalezené mimo sady

Zde jsou popsány další problémy, na které testeři narazili, které nešly přiřadit ke konkrétním krokům z testovacích sad (některé platí v celé aplikaci, jiné byly nalezeny po testování při individuálním zkoumání aplikace):

- Obrazovka „členové skupiny“:
  - Testery K i C mátlo, že na kartách členů je vizuální klikací efekt, ale nic jiného se po kliknutí neděje. Dokud nebude kliknutí vyvolávat nějakou akci, měl by být efekt odstraněn.

  - Testerovi K přijde, že ač prvotní označení sebe mezi členy není potřeba komplikovat, změna na jiného člena či odznačení člena by již mohlo být potvrzováno odkliknutím dialogu či vynucením zadání hesla: uživatel by prý jinak mohl snadno přeznačit svého člena nedopatřením, případné útoky malých dětí na telefony členů rodiny by taktéž mohly vyústit ve výměnu uživatelova člena, a chybný stav uživatel snadno přehlédne.

- Obrazovka Přehled: Na kartách skupin s jinou měnou, než je uživatelova měna, by tester K rád viděl i konverzi dlužené částky do uživatelovy měny.

- Obrazovka pro vytváření a úpravu skupiny: Tester C nechce, aby klávesnice automaticky kapitalizovala první písmena všech slov, jen prvního. Toto by patrně bylo vhodné dodat s českou lokalizací – angličtina častěji preferuje v názvech většinu slov uvozených velkým písmenem.

- Obrazovka „detail výdaje“ u výdaje s podíly: Tester C shledal, že číslo podílu je příliš blízko částce, a tak na první dojem (než si všimne, že částky jsou kurzívou) působí jako součást částky. Je nutné zvýšit rozestup mezi těmito čísly.

- Převody peněz:

  - Tester C by u nich rád měl možnost také vyplnit název, jako u výdaje.
  - Obrazovka „detail převodu“: Tester C by ocenil vizuální sjednocení s obrazovkou výdaje, barevné indikace, kdo komu platil mu přijdou přehlednější.

- Nutnost registrace: Testerovi C velmi zkazila dojem z aplikace. Možnost práce úplně bez přihlášení by nevyžadoval, postačila by možnost přihlášení přes Google účet.

- Využití spodních panelů pro menu: uživatel K preferuje využití „šuplíčků“ na levé či pravé straně obrazovky, které lze mimo tlačítka otevírat i potažením.

- Možnost zavření klávesnice: Testeři N a Z se pokoušeli klávesnici zavírat dlouhým podržením mimo textové pole. Aplikace aktuálně klávesnici zavírá jen po (krátkém) kliknutí mimo textové pole, toto chování by bylo vhodné rozšířit i na dlouhé podržení.

- Barvy:

  - Kontrast textu v tmavém režimu: Testerovi K přišel nedostatečný.
  - Primární barva v tmavém režimu: Podle testera C by měla být trochu světlejší.

4.2.3.4 Podněty testerů

Zde jsou podněty testerů na vylepšení aplikace do budoucna (identifikátor testera/testerů majících zájem o dané vylepšení uvedeno v závorkách):

- Možnost barvit skupiny pro lepší orientaci na obrazovce Přehled. [C, K, Z]

- Seznam přátel umožňující rychlejší vyplnění členů skupiny. [C, N, Z]

- Po vytvoření nové skupiny přesměrovat rovnou na vytváření členů. [N, Z]

- Možnost zvaní uživatelů aplikace do skupin přímo v aplikaci (bez nutnosti posílání ID). [C]

- Česká lokalizace. [N]

- Obdobná aplikace pro iOS. [Z]

Tabulka 4.4 Hodnocení aplikace uživateli.

Tester	Použitelnost	Přehlednost	Design	Bezchybovost	Osobní dojem
Tester K	60 %	70 %	90 %	80 %	75 %
Tester J	80 %	90 %	80 %	100 %	80 %
Tester C	69 %	80 %	20 %	86 %	75 %
Tester N	95 %	95 %	90 %	100 %	90 %
Tester Z	100 %	90 %	75 %	95 %	90 %
Průměr	80,8 %	85 %	71 %	92,2 %	82 %

4.2.4 Hodnocení aplikace testujícími uživateli

Mimo zjištění konkrétních problémů uživatelů s aplikací byl také zkoumán jejich celkový dojem z aplikace. Za tímto účelem měli testující uživatelé ohodnotit aplikaci procenty v těchto kategoriích:

- Použitelnost: jak moc by pro ně bylo možné problémovou doménu (udržování přehledů o skupinových financích) řešit pomocí této aplikace.
- Přehlednost: jak snadno se orientují v rozložení grafického rozhraní, *user experience*.
- Design: zvolené barvy, fonty, celkový estetický dojem z grafického rozhraní.
- Bezchybovost.
- Celkový osobní dojem: jak moc se jim aplikace subjektivně (ne)líbí jako celek. Pro tuto kategorii byli testeři požádání, aby hodnotili podle vlastních priorit a preferencí.

100 % znamená bezchybný úspěch v dané kategorii, 0 % znamená naprostý neúspěch v plnění (i základních) parametrů dané kategorie.

Cílem sběru hodnocení bylo zjistit, zda aplikace má budoucnost, a které její aspekty případně potřebují více doladit.

Hodnocení testerů a jejich průměry v jednotlivých kategoriích jsou v tabulce 4.4.

Celkový průměr hodnocení přes všechny kategorie je 82.2 %, což lze u prvního uživatelského testování aplikace považovat za velmi dobrý výsledek. Čísla jsou patrně tažená dolů jednak objevenými problémy (viz výše), jednak designem, na nějž měli testeři nekonzistentní reakce. Ovšem nalezené problémy lze vyřešit, a design lze poupravit (či plně vyměnit) pomocí pár krátkých řádek kódu. Vzhledem k tomu, že se jednalo o úplně první uživatelské testování aplikace, a s ohledem na provedené pozorování reakcí testerů na aplikaci, se autorka domnívá, že po oněch opravách a úpravách by si uživatelé aplikaci snadno mohli oblíbit.

4.2.5 Výsledky testování s uživateli

Uživatelské testování odhalilo problémy s uživatelskou přívětivostí aplikace, které je v plánu v dalších verzích aplikace opravit.

Uživatelé však aplikaci i přes ony problémy hodnotili pozitivně, ač šlo o její vůbec první uživatelské testování. Aplikace se tedy v budoucnu může stát spolehlivým a oceňovaným pomocníkem uživatelů.

4.3 Budoucnost testování aplikace

Před rozšiřováním aplikace o nové funkcionality (potenciálně mimo těch nezbytně nutných pro zpřístupnění aplikace běžným uživatelům, viz 1.3 Analýza požadavků) je v plánu pro udržitelnost

budoucího vývoje pokrýt aplikaci automatickými testy: zvýšit pokrytí jednotkovými testy, pro pár klíčových funkcionalit by přišly vhod i celo-aplikační testy, a spouštění testů by bylo ideální zaintegrovat na GitLab.

Mimo automatických testů budou další verze aplikace stále testovány uživateli kvůli ověření uživatelské přívětivosti. Před případnou adopcí aplikace širší veřejností, která by potenciálně umožnila ponechat část tohoto testování na reálných aktivních uživatelích, půjde patrně o obdobná testování, jako to provedené v rámci této práce. Tato testování budou prováděna po změnách, které mohou znatelně ovlivnit uživatelskou přívětivost aplikace. Pro ušetření času je však možné nesouvisející změny pro testování seskupovat, například změnu barevného schématu s revizí typů ovládacích prvků (tlačítka na obrazovce versus různé vyjíždějící panely apod.).

Kapitola 5
Budoucnost aplikace

Aby bylo možné aplikaci zpřístupnit pro běžné uživatele, je nutné doimplementovat ověření e-mailu, možnosti změny e-mailu a hesla k účtu, resetu hesla, možnost a smazání účtu (potřebné i kvůli GDPR – uživatelé musí mít možnost požádat o výmaz svých dat).

Také je nutné dodat právně korektní podmínky používání aplikace (a dodat link na podmínky na registrační obrazovku).

Je potřeba vyřešit možnost budoucího financování Firebase pro případ, že bezplatná varianta nebude stačit pro uživatelský provoz či pro objem uložených dat.

Bude doplněna možnost automaticky aktualizovat koeficienty pro převody měn. Zdroj dat pro převodní tabulku je možné řešit např. drobným prográmkem, který by periodicky konzumoval převodní koeficienty z důvěryhodného veřejného zdroje a publikoval výtah z ní do Firestore, odkud by si jej stahovaly aplikace uživatelů.

Dočasnou alternativou automatické aktualizace koeficientů pro převod měn by mohla být manuální aktualizace převodní tabulky ve Firestore, nebo naopak dočasné zablokování možností měnit měnu skupiny, vytvářet transakce s měnou odlišnou od měny skupiny, a buďto schování součtů dluhů a pohledávek v přehledu, nebo naopak zobrazování dluhů i pohledávek odděleně ve všech měnách, ve kterých je daná částka nenulová.

Problémy objevené v rámci uživatelského testování (viz 4.2.3) budou opraveny.

Budou vytvořeny automatizované celo-aplikační testy, spouštěné v emulátorech či na reálných zařízeních, které budou ověřovat základní funkcionality aplikace. Bude zvýšeno pokrytí jednotkovými testy. Také je v plánu zavedení částečné automatizace na GitLabu, např. spouštění základních testů pro všechny větve, spouštění komplexních testů pro hlavní větve, možnost vytvořit distribuovatelné APK pomocí tagu apod.

Poté bude na místě nahrát aplikaci do obchodu Google Play.

I po vyřešení těchto věcí by mohlo být dobré počkat se zpřístupněním aplikace veřejnosti až po dodání pár z níže vyjmenovaných funkcionalit (pro zjištění tohoto by byl patrně potřeba další průzkum mezi potenciálními uživateli).

5.1 Plánovaná vylepšení a rozšíření aplikace

Níže vyjmenované funkcionality by již patrně nebyly pro běh aplikace překážkou, neměl by být problém uživatelům zpřístupnit aplikaci i bez nich (což by umožnilo sledovat uživatelský provoz, a podle dat aplikaci vylepšovat). Pro nemálo z nich by také bylo vhodné provést další průzkum mezi uživateli, jaká varianta by jim pro danou funkcionalitu vyhovovala. Taktéž je potřeba průzkum pro zjištění, jak jsou jednotlivé funkcionality jsou pro uživatele podstatné, aby je bylo možné zprioritizovat.

- Možnost upravit přezdívku účastníka skupiny. Ideálně nejspíš s historií původních přezdívek, aby bylo snadné účastníky zpětně identifikovat při úpravě na nevypovídající přezdívku.
- Možnost mazat účastníky, kteří nemají transakce (např. pro účastníky přidané do skupiny omylem) či skrývat účastníky v přehledu dlužných stavů (např. pro nekooperativní účastníky či pro skupiny, jejichž složení se průběžně mění).
- „Seznam přátel“ pro urychlení přidávání členů do skupin (přátelé by mohli mít např. přednastavenou defaultní přezdívku) a umožnění zvaní uživatelů do skupin v rámci samotné aplikace.
- Upravená posloupnost obrazovek pro vytváření skupiny, uživateli by po vytvoření mohlo být navrženo přidat svou preferovanou přezdívku (včetně automatického prolinkování vzniklého *lokálního člena skupiny* k uživateli) a následně by mohl napřímo přejít k vytváření dalších členů (včetně případného využití výše zmíněného seznamu přátel).
- Umožnit na skupině nastavovat, zda se k ní v daný moment smí připojovat další uživatelé. Pokud by uživatelé skupiny stáli o toto zabezpečení, mohou ho přenastavovat podle toho, zda v daném období zvou nového uživatele. (Pokud by o zabezpečení nestáli, mohli by ponechat nastavení permanentně aktivní.)
- Možnost vykázat uživatele ze skupiny. Pro začátek může být realizovaná i částečně manuálně, kdy by uživatelé o toto žádali e-maily či v ticketovacím systému od podpory. V delším časovém horizontu by však bylo vhodné toto přidat k následujícímu bodu.
- Ve skupině oddělit „běžné uživatele“ a „administrátory“ či zavést pro některé akce na skupině možnost hlasování. Zde by jistě byl nutný průzkum mezi uživateli ohledně jejich preferencí pro způsob fungování. Některé funkcionality (existující i plánované) ve skupinách by bylo vhodné omezit: např. mazání transakcí (alespoň těch vzniklých např. před více než 15 minutami – nemusí být nutné omezovat mazání z důvodů dvojitého vložení záznamu dvěma uživateli apod.), přepínání možnosti pro přijímání nových uživatelů, potenciálně editace starých transakcí (mimo např. pole „poznámka“) apod. Tyto funkcionality by mohly být přístupné jen administrátorům skupiny (kteří by mohli i z ostatních uživatelů dělat administrátory), mohly by být podmíněné hlasováním (že pro danou věc musí „hlasovat“ daný počet / dané procento uživatelů ve skupině), či by skupiny měly možnost volit mezi těmito systémy.
- Možnost zpřístupnit skupinu některým uživatelům jen pro čtení.
- Možnost k transakci připojit účtenku či fakturu (jako obrázek či PDF).
- Ukázat u transakcí převod do primární měny skupiny, pokud transakce v primární měně není. Převody by bylo možné u výdajů zobrazit i pro zaplacené částky a podíly, pokud by uživatelé o takový detail měli zájem.
- Možnost označovat transakce, které chce uživatel s ostatními prodiskutovat či ke kterým chce např. sám doplnit detaily (třeba onu fotku účtenky). Zajímavou variantou by byla např. možnost připojit komentář (uživatel připíše vlastní poznámku, na rozdíl od pouhého označení nevypovídající vlaječkou), na základě přítomnosti komentářů by byly transakce v přehledu skupiny barevně zvýrazňovány.
- Možnost ukládat čísla bankovních účtů a na jejich základě generovat platební QR kódy pro zobrazení, uložení či sdílení. Účty by bylo možné přidat k uživatelskému účtu či k lokálnímu členovi skupiny (zde potenciálně opět s historií, či s možností úpravy jen administrátorem). Z obrazovky pro generování QR kódy by uživatelé však mohli mít i možnost následně pokračovat na zaevidování daného převodu peněz (byl-li by bankovní účet na uživatelském účtu, bylo by nutné zvolit skupinu; byl-li by na členovi skupiny, nabídlo by se zaevidování převodu do aktuální skupiny).

- Možnost vygenerovat seznam plateb nutných k vyrovnání dluhů v rámci dané skupiny. Uživatel by ideálně mohl zadat „toleranci“ – částku, která je považována za přijatelný zbytkový dluh po vyrovnání (mohl by nastavit i nulu). Toto by při použití správného algoritmu umožnilo vygenerovat skutečně co nejmenší seznam plateb, neboť bez tolerance se nutné platby díky drobným zbytkovým částkám často blíží kartézskému součinu členů skupiny.
- Možnosti řazení seznamu skupin v přehledu. Např. abecedně dle jména skupiny, podle data poslední přidané transakce, či např. uživatelům umožnit definovat vlastní řazení. Také by bylo možné nechat uživatele označovat jednotlivé skupiny jako „méně podstatné“ pro případ, kdy si detail dané skupiny neplánují často zobrazovat, ale ani nechtějí skupinu opustit: seznam skupin by v takovém případě byl rozdělen na dva, nejdříve standardní skupiny, až po nich skupiny označené jako méně podstatné.
- Možnost přiřazovat skupinám barvy pro karty skupin v přehledu (pro usnadnění nalezení hledané skupiny). Bylo by možné např. definovat množinu přístupných barev, a identifikátor uživatelem vybrané barvy ukládat k uživatelově odkazu na skupinu (uživatelé by si tedy barvy vzájemně nepřepisovali).
- Možnost mazat skupiny. Právo na mazání by mohlo být zpřístupněno administrátorům, vycházet z hlasování, nebo být přidělováno systémem „poslední zhasne“, kdy by poslední uživatel opouštějící skupinu měl možnost či byl nucen skupinu smazat.
- Lokalizace do dalších jazyků.
- Více responzivní obrazovky (pro uživatele tabletů a velmi velkých telefonů).
- Zrychlení načítání obrazovky Přehledu: je-li uživatel ve více skupinách (na testovacím zařízení od cca 10), je při spuštění aplikace po chvíli její nečinnosti potřeba krátká, ale znatelná chvilka na načtení dat. Již nyní se data pro obrazovku Přehledu připravují ve dvou vlnách: načtení skupin a načtení celkového uživatelova stavu. Bylo by lepší načtení skupin samotné rozdělit do dvou vln: nejdříve načtení jmen (a na pozadí i měn) skupin pro zobrazení karet bez dlužných stavů, a až následně donačtení transakcí skupin a dopočet dlužných stavů.
- Agregace částek pro výpočty dlužných stavů ve více měnách: aktuálně jsou při agregaci částky nejdříve převedeny do cílové měny. Pro lehce zvýšenou přesnost by mohly být agregovány ve všech měnách odděleně a převedeny do cílové až na konci výpočtu.
- Podpora pro více měn.
- Pokud bude podporováno tolik měn, že přestane být uživatelsky pohodlné je vybírat ze seznamů při vytváření a úpravách transakcí a skupin, pak vytváření a udržování seznamů měn, které uživatel nedávno použil, a výměna stávajících dialogů pro výběr měny v aplikaci na variantu, kdy se uživateli zobrazí nejdříve onen kratší personalizovaný seznam s možností „jiná měna“, a až tato možnost ho navede na kompletní seznam podporovaných měn.
- Možnost nastavení dvoufaktorového ověření přihlášení.
- Možnost zpřístupnění dat pro ne-uživatele Android systému. Pro toto by byla pro začátek patrně nejlepší responzivní webová aplikace, která by zajistila přístup pro uživatele jiných mobilních operačních systémů i pro oněch pár potenciálních uživatelů, kteří raději používají osobní počítače či tablety.

Aplikaci jistě lze dále vylepšovat. Ovšem již po pár úpravách zmíněných v počátku kapitoly by měla být zpřístupnitelná veřejnosti, a mohla by tak začít uživatelům pomáhat s udržováním přehledu o skupinových financích.

Závěr

Cílem práce bylo vytvořit Android aplikaci, která by uživatelům pomohla v udržování přehledu o společných financích.

Za tímto účelem byla provedena analýza potenciálních uživatelů a existující konkurence. Byly specifikovány konkrétní požadavky na aplikaci.

S ohledem na požadavky bylo navrženo technické řešení: využité technologie, schéma ukládaných dat, odpovědnosti jednotlivých obrazovek, jejich rozložení a možnosti navigace mezi nimi.

Na základě návrhu byla implementována aplikace v jazyce Kotlin, využívající služby Firebase. Implementace je do budoucna snadno upravitelná a rozšiřitelná, umožňuje i rozdělení do modulů. Případné pády aplikace jsou zaznamenávány pomocí Firebase Crashlytics.

Uživatelská přívětivost aplikace byla otestována potenciálními uživateli. Aplikace v testování obstála, získala dobré hodnocení, a byly identifikovány konkrétní možnosti pro její vylepšení.

Budoucnost aplikace a její možné rozšiřování bylo popsáno v kapitole 5.

Cíle práce byly splněny.

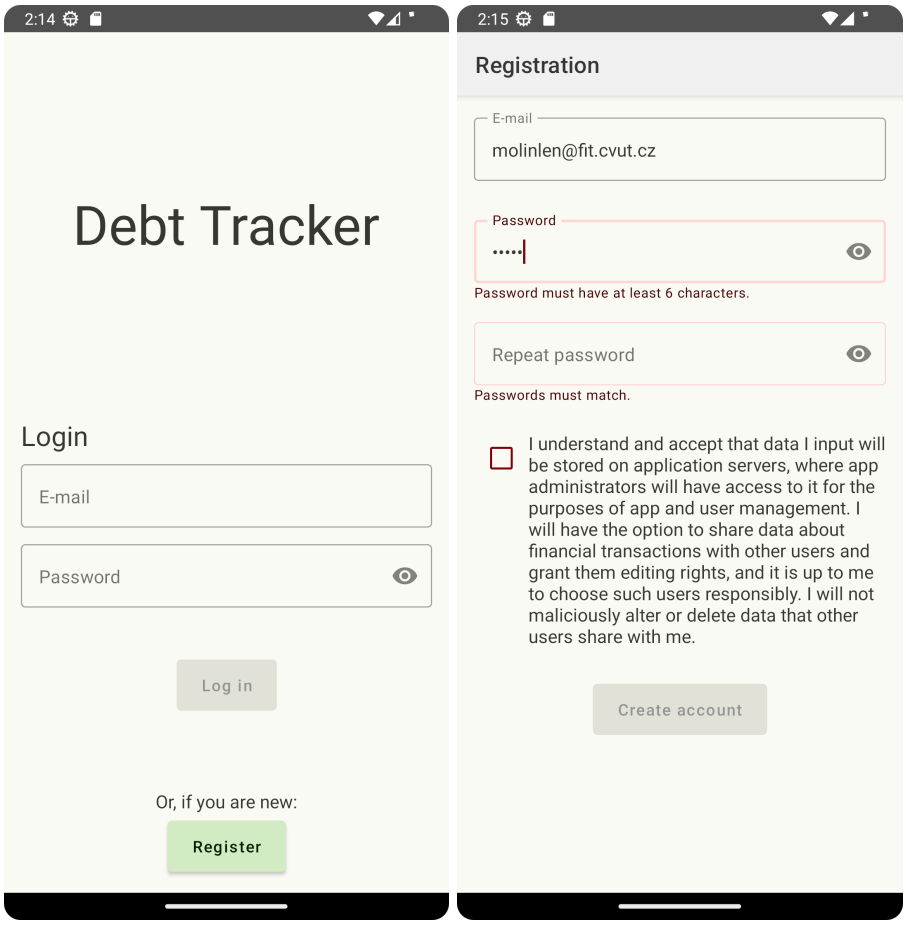
Tabulka 5.1 Pokrytí funkčních požadavků případy užití.

Případ užití	FN1	FN2	FN3	FN4	FN5	FN6	FN7	FN8	FN9
UC01	✓								
UC02		✓							
UC03			✓						
UC04									✓
UC05									✓
UC06			✓						
UC07			✓						
UC08			✓	✓		✓		✓	
UC09				✓				✓	
UC10				✓					
UC11						✓			
UC12				✓					
UC13				✓					
UC14			✓						
UC15					✓				
UC16						✓			
UC17							✓		
UC18							✓		
UC19			✓						

Příloha A

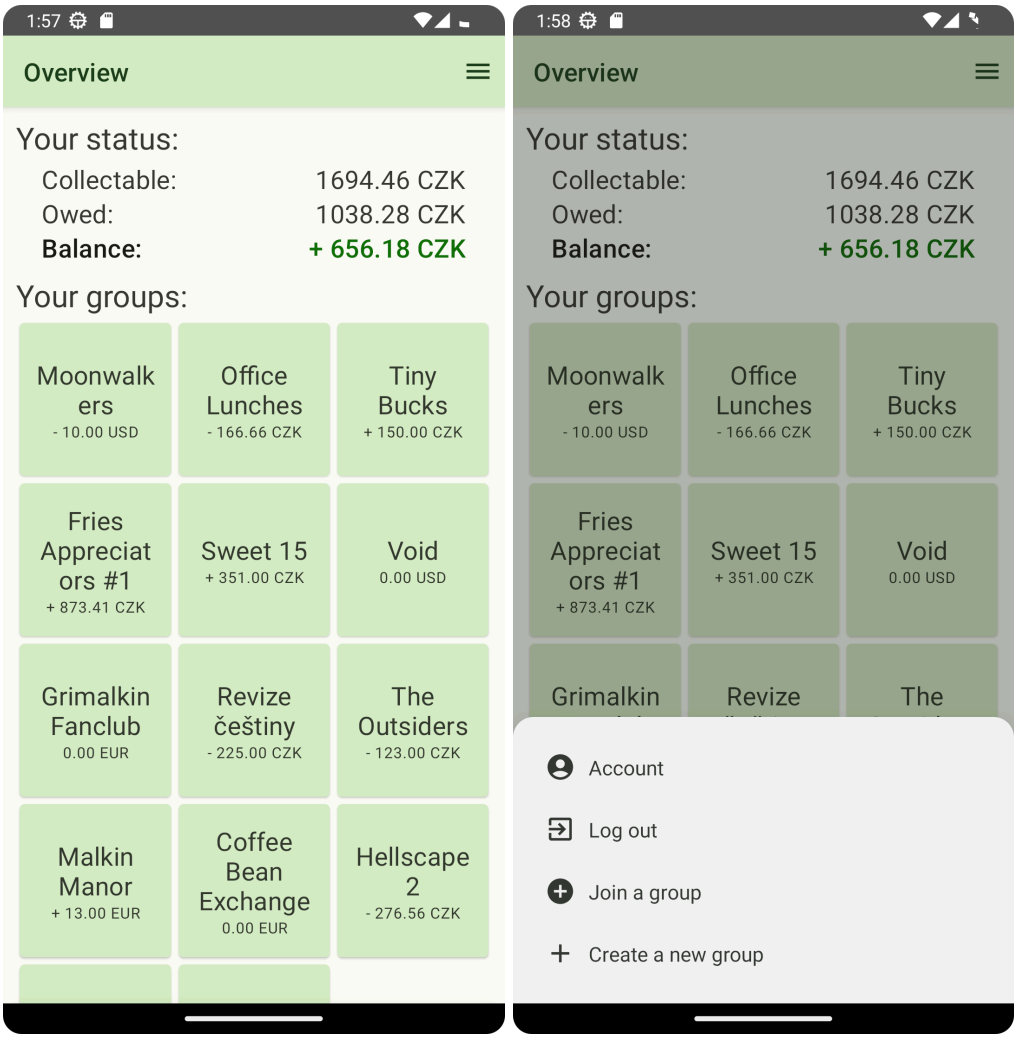
Vzhled výsledné aplikace

Snímky obrazovky byly pořízeny na emulovaném Google Pixel 3a (parametry viz tabulka 4.1).

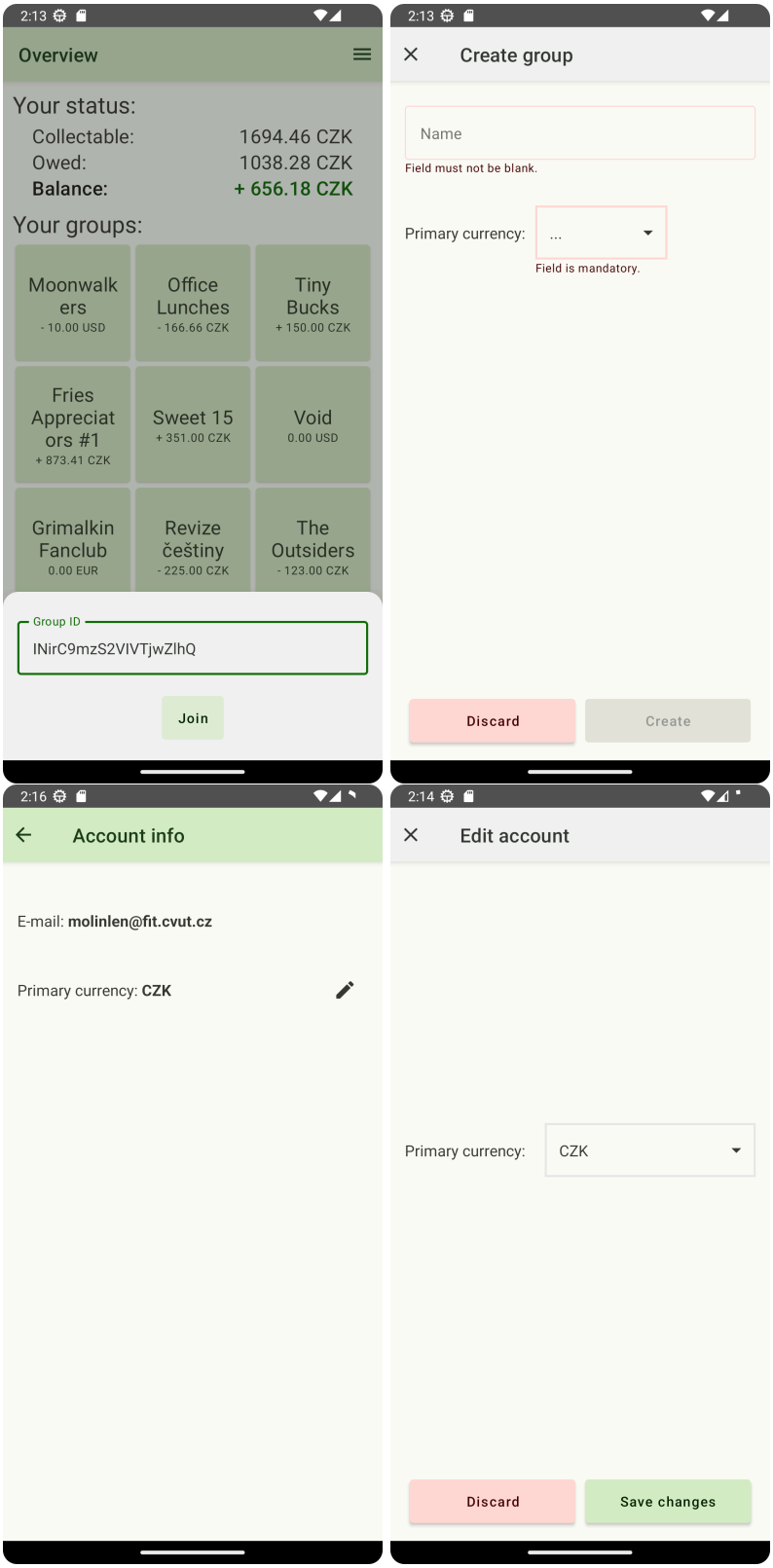


Obrázek A.1 Přihlašovací a registrační obrazovka.

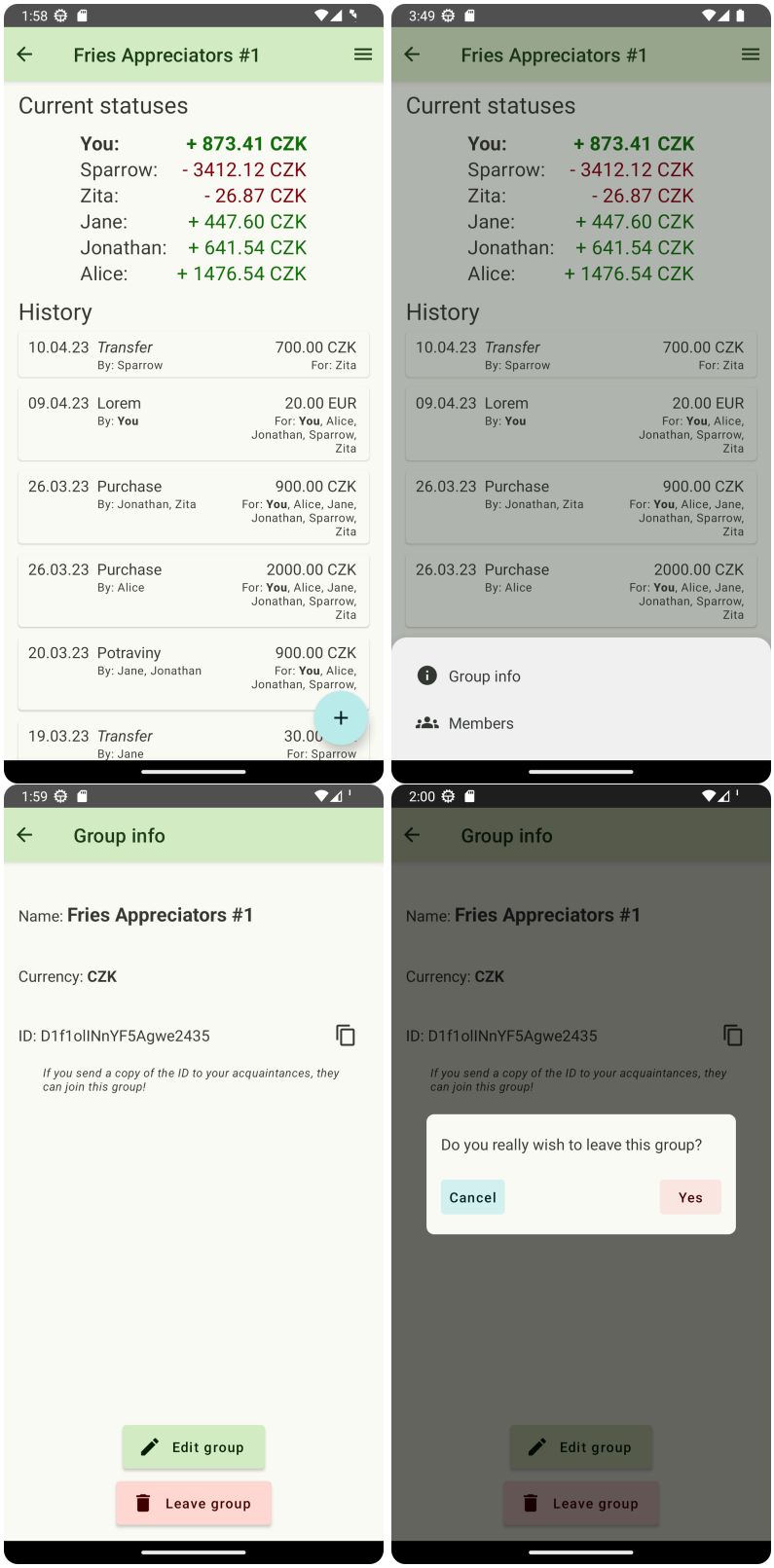
Obrázek A.2 Ikona aplikace. Zleva: čtvercová a kulatá verze ikony pro *release* variantu sestavení aplikace; alternativní verze ikony pro *debug* sestavení – umožňuje na testovacích zařízeních snadno odlišit varianty aplikace. Za SVG grafiku pro ikony moc děkuji Danny.



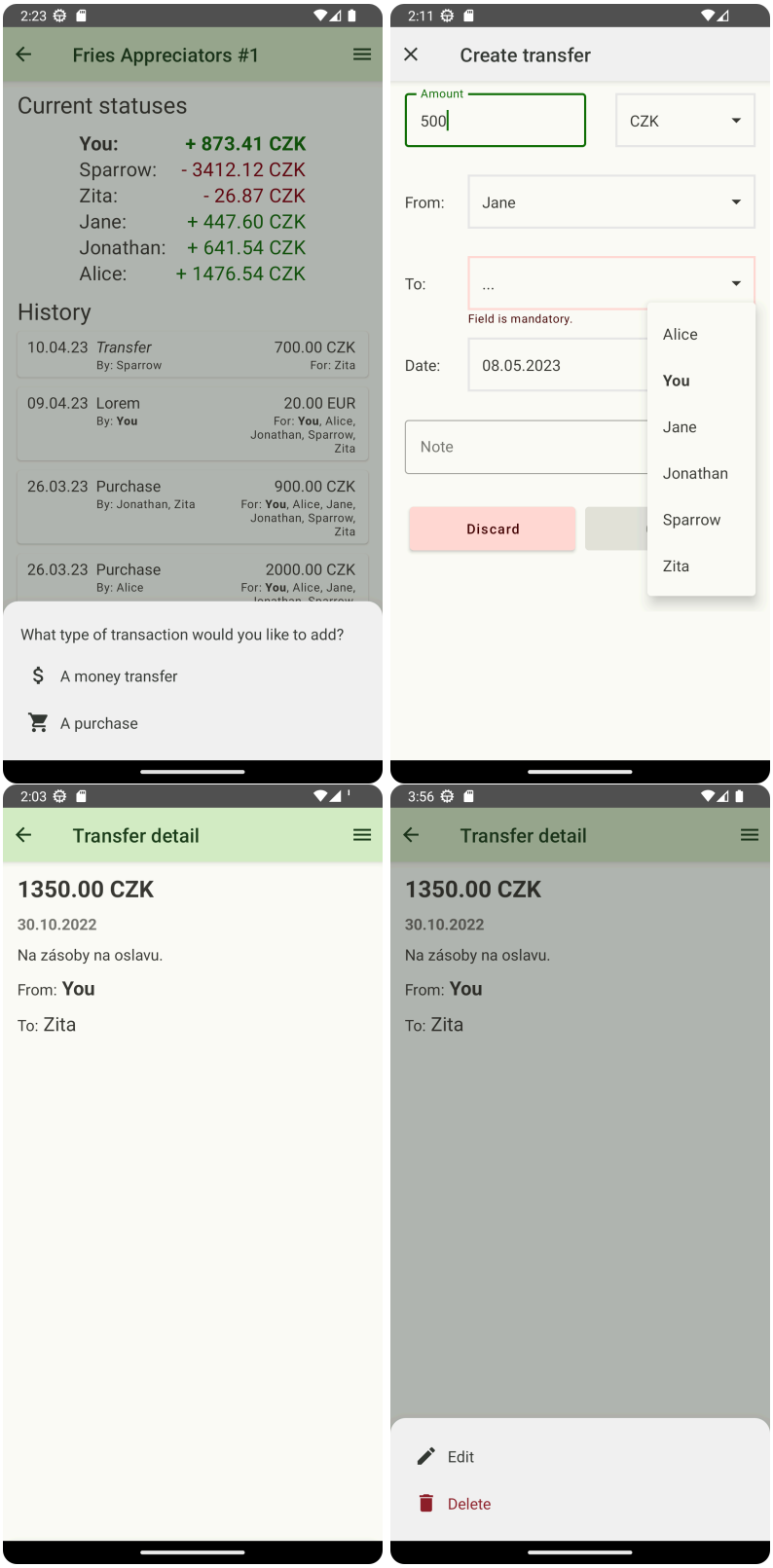
Obrázek A.3 Obrazovka Přehled a její menu.



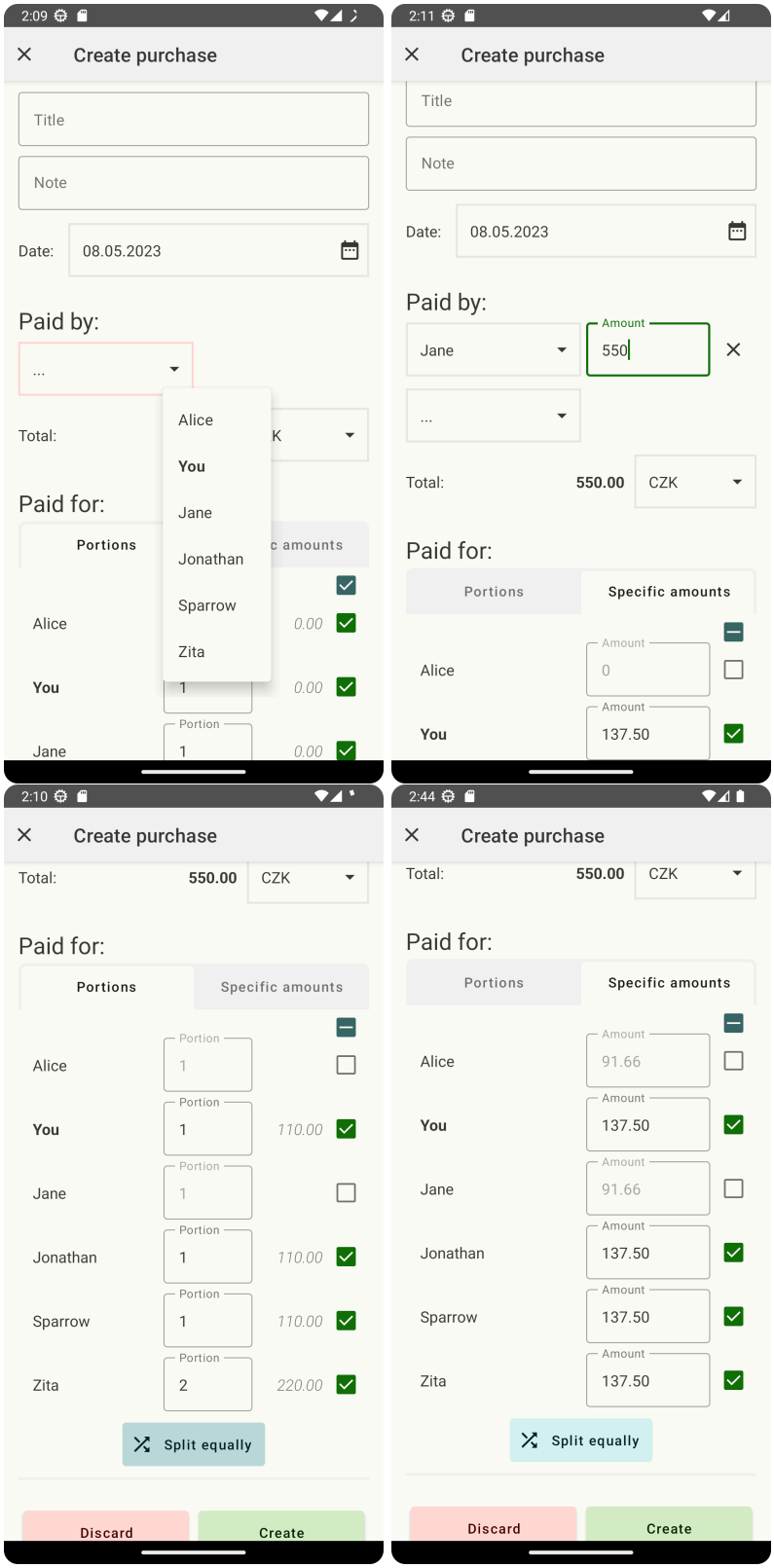
Obrázek A.4 Připojení ke skupině; vytvoření nové skupiny; informace o účtu; úprava preferované měny.



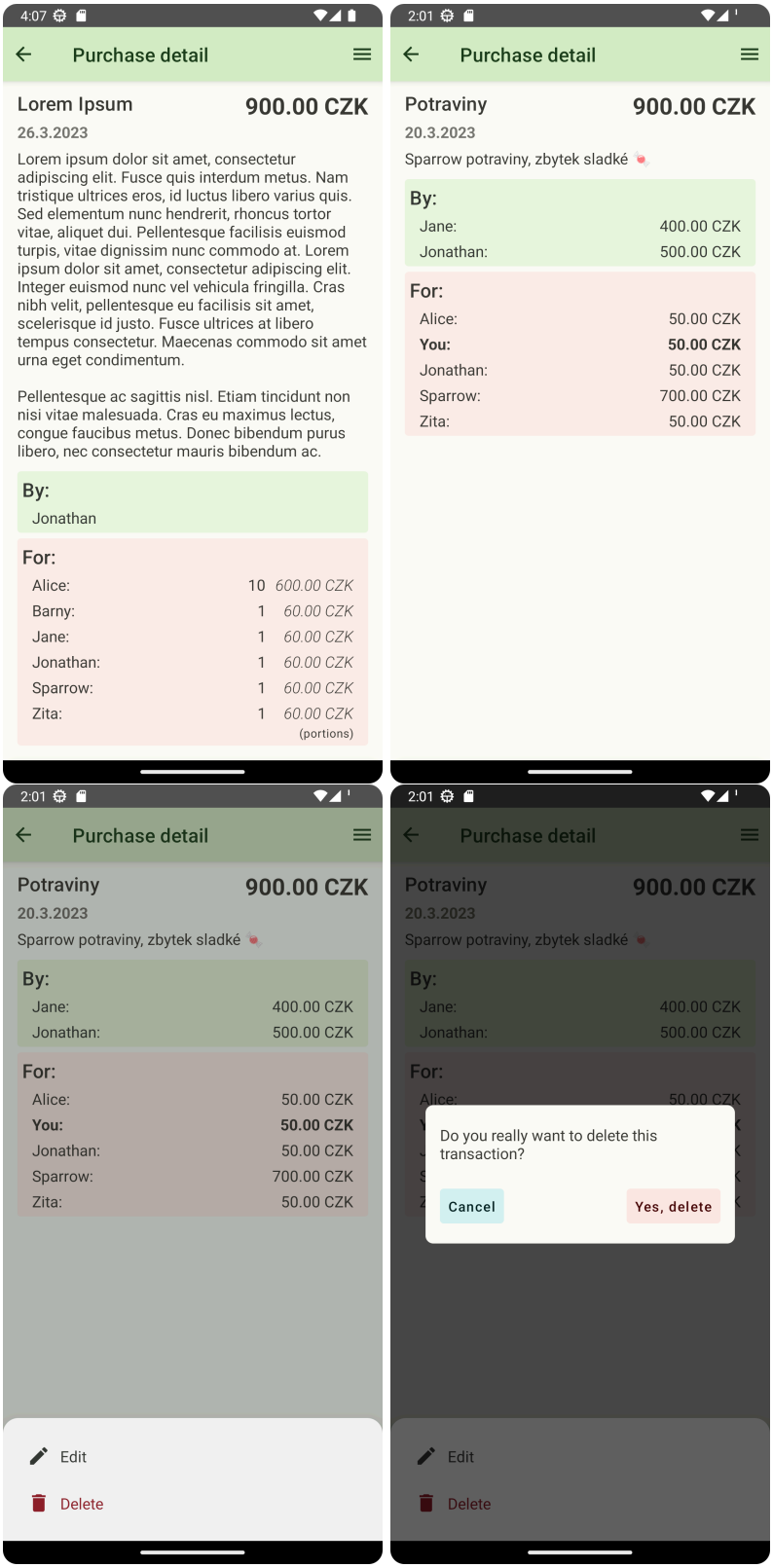
Obrázek A.5 Detail skupiny; jeho menu; informace o skupině; dialog potvrzení opuštění skupiny.



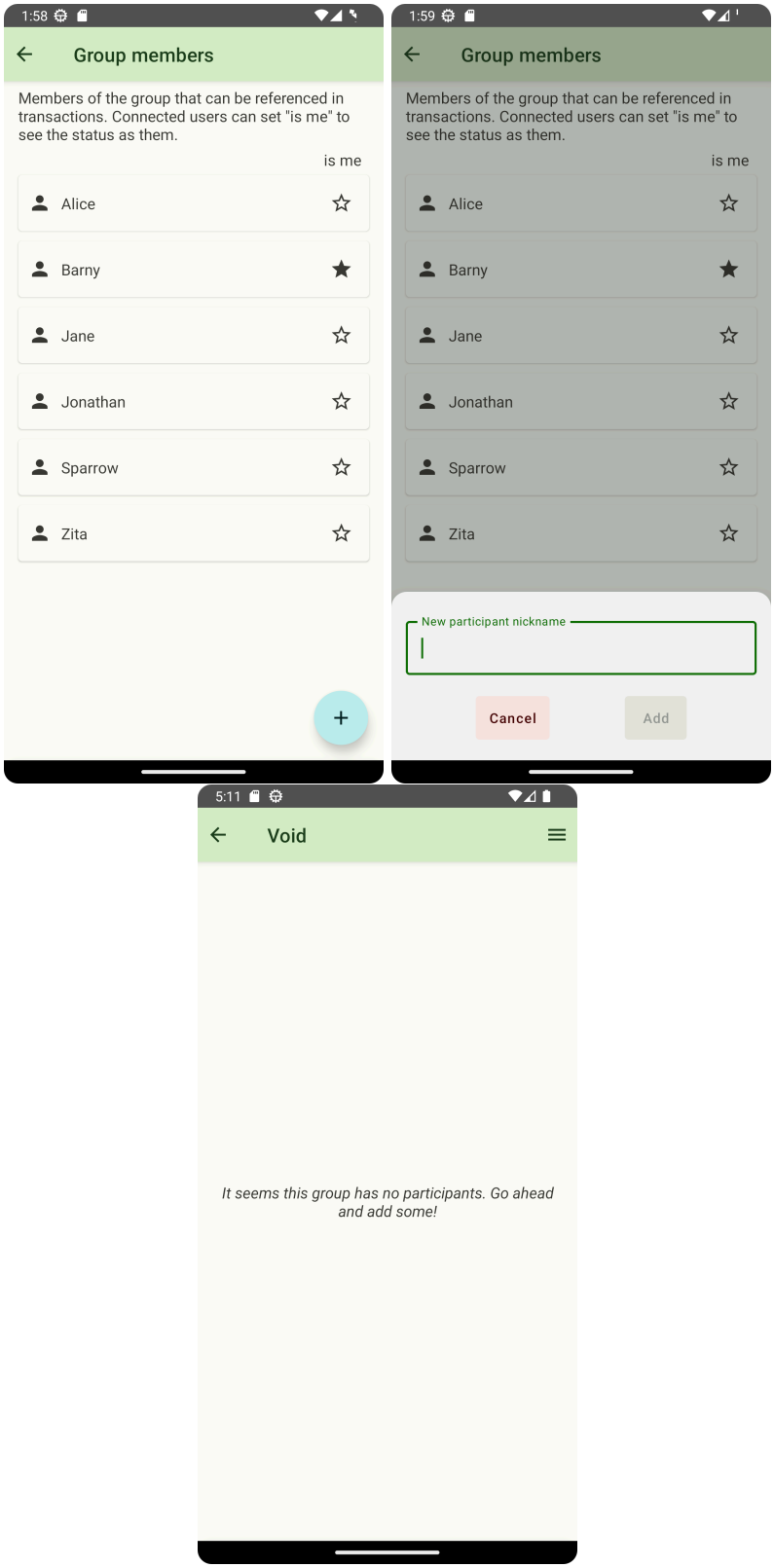
Obrázek A.6 Zvolení typu vytvářené transakce; vytváření převodu; detail převodu; a jeho menu.



Obrázek A.7 Vytváření nákupu: volba plátce, jím zaplacené částky, určení podílů pomocí podílových koeficientů či konkrétních částek.



Obrázek A.8 Detail výdaje: děleného koeficienty; děleného částkami; s menu; s dialogem pro potvrzení smazání.



Obrázek A.9 Seznam členů skupiny; možnost přidání člena skupiny; detail skupiny bez členů.

Literatura

1. STATCOUNTER. *Mobile Operating System Market Share Czech Republic* [online]. Apr 2022 [cit. 2022-05-30]. Dostupné z: `https://gs.statcounter.com/os-market-share/mobile/czech-republic`.
2. STATCOUNTER. *Mobile Operating System Market Share Worldwide* [online]. Apr 2022 [cit. 2022-06-02]. Dostupné z: `https://gs.statcounter.com/os-market-share/mobile`.
3. SPLITWISE. *Splitwise* [online] [cit. 2022-05-31]. Dostupné z: `https://play.google.com/store/apps/details?id=com.Splitwise.SplitwiseMobile`. Splitwise app on Google Play.
4. SPLITWISE. *Merge accounts* [online] [cit. 2022-06-01]. Dostupné z: `https://secure.splitwise.com/users/merge`. Možnost sloučit Splitwise účty; přístupné jen po přihlášení do služby.
5. TRICOUNT. *Tricount - Split bills & manage group expenses* [online] [cit. 2022-05-31]. Dostupné z: `https://play.google.com/store/apps/details?id=com.tribab.tricount.android`. Tricount app on Google Play.
6. STEP UP LABS. *Settle Up – Group Expenses* [online] [cit. 2022-05-31]. Dostupné z: `https://play.google.com/store/apps/details?id=cz.destil.settleup`. Settle Up app on Google Play.
7. GOOGLE. *Firebase* [online] [cit. 2023-03-13]. Dostupné z: `https://firebase.google.com/`.
8. GOOGLE. *Firebase Firestore* [online] [cit. 2023-03-23]. Dostupné z: `https://firebase.google.com/docs/firestore`.
9. GOOGLE. *Understand Cloud Firestore billing* [online] [cit. 2023-05-06]. Dostupné z: `https://firebase.google.com/docs/firestore/pricing`.
10. GOOGLE. *Identity Platform pricing* [online] [cit. 2023-05-06]. Dostupné z: `https://cloud.google.com/identity-platform/pricing`.
11. GOOGLE. *Firebase Crashlytics* [online] [cit. 2023-05-06]. Dostupné z: `https://firebase.google.com/docs/crashlytics`.
12. GOOGLE. *Google Analytics for Firebase* [online] [cit. 2023-05-06]. Dostupné z: `https://firebase.google.com/docs/analytics`.
13. GOOGLE. *Firebase Authentication* [online] [cit. 2023-05-06]. Dostupné z: `https://firebase.google.com/docs/auth`.
14. GOOGLE. *Writing conditions for Cloud Firestore Security Rules* [online] [cit. 2023-03-23]. Dostupné z: `https://firebase.google.com/docs/firestore/security/rules-conditions`.

15. GOOGLE. *Firebase App Distribution* [online] [cit. 2023-05-10]. Dostupné z: https://firebase.google.com/docs/app-distribution.
16. *PlantUML* [online] [cit. 2023-03-15]. Dostupné z: https://plantuml.com/. Open-source jazyk pro definici diagramů a nástroj pro jejich překlad.
17. *PlantUML Web Server* [online] [cit. 2023-03-15]. Dostupné z: http://www.plantuml.com/plantuml/. Online nástroj pro překlad PlantUML do obrázků, umožňuje export nejen do nepříliš kvalitních PNG, ale i do SVG.
18. GOOGLE. *Cloud Firestore Data model* [online] [cit. 2023-03-23]. Dostupné z: https://firebase.google.com/docs/firestore/data-model.
19. SCHAEFER, Lauren. *What is NoSQL?* [Online] [cit. 2023-05-06]. Dostupné z: https://www.mongodb.com/nosql-explained.
20. GOOGLE. *Supported data types* [online] [cit. 2023-03-23]. Dostupné z: https://firebase.google.com/docs/firestore/manage-data/data-types. Data types supported by Firebase Firestore.
21. GOOGLE. *Understanding layout - Material Design* [online] [cit. 2023-05-07]. Dostupné z: https://m2.material.io/design/layout/understanding-layout.html. Material 2 verze doporučení pro rozložení prvků GUI, koncentrující primární akce uživatele do horního "App bar".
22. GOOGLE. *Layout – Material Design 3* [online] [cit. 2023-05-07]. Dostupné z: https://m3.material.io/foundations/layout/understanding-layout/parts-of-layout#56458f83-51a7-464b-9d77-c2a07bd1fcc1. Sekce "Navigation region"ukazuje doporučené uístění navigačních prvků na různých typech zařízení.
23. MARVEL. *Marvel App* [online] [cit. 2023-03-13]. Dostupné z: https://marvelapp.com/.
24. GOOGLE. *Material Design* [online] [cit. 2023-03-13]. Dostupné z: https://m2.material.io/.
25. GOOGLE. *Download Android Studio and App Tools - Android Developers* [online] [cit. 2023-03-16]. Dostupné z: https://developer.android.com/studio.
26. GOOGLE. *Domain layer* [online] [cit. 2023-04-28]. Dostupné z: https://developer.android.com/topic/architecture/domain-layer.
27. GOOGLE. *ViewModel overview* [online] [cit. 2023-05-03]. Dostupné z: https://developer.android.com/topic/libraries/architecture/viewmodel.
28. GOOGLE. *Navigating with Compose* [online] [cit. 2023-04-27]. Dostupné z: https://developer.android.com/jetpack/compose/navigation.
29. JETBRAINS. *Flow* [online] [cit. 2023-04-27]. Dostupné z: https://kotlinlang.org/api/kotlinx.coroutines/kotlinx-coroutines-core/kotlinx.coroutines.flow/-flow/.
30. JETBRAINS. *MutableStateFlow* [online] [cit. 2023-04-27]. Dostupné z: https://kotlinlang.org/api/kotlinx.coroutines/kotlinx-coroutines-core/kotlinx.coroutines.flow/-mutable-state-flow/.
31. JETBRAINS. *update* [online] [cit. 2023-04-27]. Dostupné z: https://kotlinlang.org/api/kotlinx.coroutines/kotlinx-coroutines-core/kotlinx.coroutines.flow/update.html. Funkce pro aktualizaci hodnoty držené v MutableStateFlow.
32. GOOGLE. *State and Jetpack Compose* [online] [cit. 2023-04-27]. Dostupné z: https://developer.android.com/jetpack/compose/state.
33. ΛRROW. *Arrow Core. Functional companion to Kotlin's Standard Library* [online] [cit. 2023-03-30]. Dostupné z: https://arrow-kt.io/docs/core/.
34. ΛRROW. *Either* [online] [cit. 2023-03-30]. Dostupné z: https://arrow-kt.io/docs/apidocs/arrow-core/arrow.core/-either/.

35. JETBRAINS. *IntelliJ IDEA – the Leading Java and Kotlin IDE* [online] [cit. 2023-03-16]. Dostupné z: https://www.jetbrains.com/idea/.

36. JETBRAINS. *Sealed classes and interfaces* [online] [cit. 2023-03-30]. Dostupné z: https://kotlinlang.org/docs/sealed-classes.html.

37. SIX. *Financial Data Standards* [online] [cit. 2023-04-03]. Dostupné z: https://www.six-group.com/en/products-services/financial-information/data-standards.html. Pro seznam měn s jejich desetinnou přesností: ISO 4217 - Currency Code Maintenance – Current and Historical Lists – Current Currency & Funds – List One.

38. GOOGLE. *Shrink, obfuscate, and optimize your app* [online] [cit. 2023-05-05]. Dostupné z: https://developer.android.com/build/shrink-code.

39. GOOGLE. *Add data to Cloud Firestore* [online] [cit. 2023-03-13]. Dostupné z: https://firebase.google.com/docs/firestore/manage-data/add-data#kotlin+ktx. Možnosti vkládání dat do Firebase Firestore (zmiňuje i Timestamp).

40. GOOGLE. *Lists and grids* [online] [cit. 2023-05-05]. Dostupné z: https://developer.android.com/jetpack/compose/lists#lazy.

41. NIELSEN, Jakob; LANDAUER, Thomas K. A mathematical model of the finding of usability problems. 1993. Dostupné z DOI: 10.1145/169059.169166.