VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ APLIKACE PRO SDÍLENÉ NAKUPOVÁNÍ

MOBILE APP FOR SHARED SHOPPING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **BARBORA ŠŤASTNÁ**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání diplomové práce

Studentka: **Šťastná Barbora, Bc.**
Program: Informační technologie Obor: Informační systémy
Název: **Mobilní aplikace pro sdílené nakupování**
Mobile App for Shared Shopping
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s problematikou sdíleného nakupování, identifikujte příležitosti pro podporu této činnosti pomocí chytrého telefonu.
2. Seznamte se s problematikou vývoje aplikací pro chytré mobilní telefony.
3. Navrhněte aplikaci pro podporu sdíleného nakupování. Zaměřte se na uživatelskou zkušenost, bezpečnost, soukromí uživatelů a snadnost použití.
4. Implementujte navrženou aplikaci.
5. Testujte vytvořenou aplikaci s vhodnou skupinou uživatelů. Iterativně vylepšujte uživatelské rozhraní i funkčnost aplikace.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Šimlová, D.: Nakupujeme společně: Objevte 6 výhod komunitního nakupování
- Berkman, E., Hoober, S.: Designing Mobile Interfaces, O'Reilly Media, Inc. 2011
- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3, značné rozpracování bodů 4 a 5.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 27. listopadu 2019

Abstrakt

Cílem této práce je vytvořit mobilní aplikaci pro operační systém Android, jež lidem provozujícím komunitní nakupování se svými známými usnadní jeho organizaci. Každý uživatel si tedy může vytvářet svoji vlastní komunitu, které nabízí společné nákupy či nějaké jeho produkty ke koupi. Tyto nabídky kategorizuje do jím spravovaných kanálů, z nichž si jeho přátelé vybírají jen ty, jejichž nabídky jim mají být zobrazeny. Je-li pak některými nějaká akceptovaná, zobrazí se mu potřebné informace o jejich dílčích nákupech. Aplikace tak uživatelům umožňuje nabízet svým známým sdílené nákupy např. přímo u nějakého pěstitele, od něhož by jako jednotlivci nemohli nakupovat. Tím si mohou společně obstarat kvalitní potraviny za příznivější cenu a rovněž kolektivním nákupem mohou redukovat produkci plastových obalů.

Abstract

The goal of this thesis is to create a mobile application for the Android operating system, which makes organizing a community shopping easier for its users and their friends. Each user can build a community of their own that offers collective purchasing of products, or can offer their products for purchase. They categorize these offers to their channels and their friends choose only those channels that they find interesting. If their friends accept their offer, they see the necessary information about their individual purchases. Thus, this application enables users to offer shared purchasing to their friends e.g. at a grower from whom these people could not buy individually. They can procure quality produce together at a competitive price and they can also reduce production of plastic packaging by purchasing as a group.

Klíčová slova

komunitní nakupování, mobilní aplikace, Android, komponenty aplikace, záměry, Model-View-ViewModel, Room, Retrofit, LiveData

Keywords

community shopping, mobile application, Android, app components, intents, Model-View-ViewModel, Room, Retrofit, LiveData

Citace

ŠŤASTNÁ, Barbora. *Mobilní aplikace pro sdílené nakupování.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Mobilní aplikace pro sdílené nakupování

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením pana prof. Ing. Adama Herouta, Ph.D. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Barbora Šťastná
3. června 2020

Poděkování

Děkuji panu prof. Ing. Adamovi Heroutovi, Ph.D. za příležitost, díky níž mohla vzniknout tato práce, za odborné vedení, pravidelné konzultace, na nichž mi trpělivě zodpovídal mé dotazy, i za cenné rady.

Obsah

Kapitola 1

Úvod

V posledních několika letech se začíná mnoho českých spotřebitelů vymezovat vůči masové produkci jednorázových plastových obalů, vůči používání průmyslových hnojiv i chemických postřiků a někteří i vůči nízkým a neadekvátním výkupním cenám vypěstovaných plodin drobnými zemědělci. Tito lidé tak hledají způsoby, jak se alespoň v určité míře přestat na těchto počinech podílet i jak si opatřit za nižší ceny pro některé běžně finančně nedostupné kvalitní a zdravotně nezávadné potraviny.

Jedním z několika existujících způsobů je tzv. komunitní nakupování. To lidem nabízí možnost si zakoupit kupř. kosmetické, mycí i prací prostředky šetrné k přírodě za příznivější cenu, nebo si obstarat např. různá semínka, ořechy či sušené ovoce i ve větším množství, jež nebude baleno jen po několika desítkách gramů v malých plastových pytlících, jak je tomu mnohdy v běžných prodejnách. Jelikož však organizace takovýchto nákupů může být časově náročná a jelikož i mě pálí, jaký dopad má naše jednání na životní prostředí, rozhodla jsem se vytvořit mobilní aplikaci pro operační systém Android, která by mohla tento současný fenomén podpořit. Cílem této práce je tedy vytvořit nástroj, jenž lidem provozujícím komunitní nakupování se svými blízkými i kamarády usnadní jeho organizaci a rovněž jenž jim na jednom místě poskytne přehled potřebných informací o dílčích nákupech jednotlivých jeho participantů.

V následujícím textu bude čtenáři nejdříve vysvětleno, co pojem komunitní nakupování znamená a proč se někteří lidé k tomuto způsobu nakupování uchylují. Taktéž se dozví, v jakých podobách se vyskytuje i jak probíhá. Další část práce, tedy 3. kapitola je zaměřena na vývoj mobilních aplikací pro operační systém Android. V ní bude čtenáři sděleno, z jakých základních komponent jsou jednotlivé mobilní aplikace tvořeny i jak jsou spouštěny, a rovněž v ní bude seznámen s jejich doporučenou architekturou. V kapitole 4 mu pak bude přiblížena základní koncepce mobilní aplikace, která je předmětem této práce, a také v ní budou vylíčeny její jednotlivé části doplněné o snímky dílčích obrazovek výsledné produktu. A konečně v posledních dvou kapitolách mu bude popsáno, jakým způsobem byla výsledná aplikace realizovaná i jak byla testovaná.

Kapitola 2

Komunitní nakupování a jeho formy

V posledních několika letech se začíná nezanedbatelná řada českých konzumentů zajímat o to, jaké zboží nakupují. Zaobírají se tedy např. tím, jaká je jeho kvalita i jestli je zdravotně nezávadné, zda bylo vypěstováno či vyrobeno udržitelným způsobem, jaký vliv má nejen jeho výroba, ale i spotřeba na životní prostředí atp. Ve společnosti tak postupně vznikají všelijaké aktivity pro opatřování různých produktů, zejména však potravin, jinými než konvenčními metodami. Jednou z nich je tzv. komunitní nakupování. Tento fenomén však nejen v reakci na poukazování na globální environmentální problémy médii, ale i z několika dalších důvodů provozuje v běžném životě nemalý počet lidí, zejména však mladší generace.

V následujícím textu se budu zabývat právě komunitním nakupováním. Vysvětlím tedy, co tento pojem znamená a proč někteří lidé takto nakupují. Jelikož jsem však nenašla žádnou existující mobilní aplikaci podporující tento způsob nakupování, bude se dále věnovat jen tomu, v jakých formách se komunitní nakupování vyskytuje a jak probíhá.

2.1 Pojem komunitní nakupování a lidská motivace pro jeho provozování

Komunitní neboli *sdílené nakupování* [33] je způsob nakupování, jehož základní podstata spočívá v tom, že se několik lidí z určité lokality dohodne na společném nákupu, který následně realizuje zpravidla jen jedna osoba v předem sjednaném ať již kamenném, nebo elektronickém obchodu či u domluveného dodavatele. Z pohledu daného prodejce se tedy uskutečněný obchod jeví jako jeden nákup od jednoho člověka, ten však obsahuje poptávané zboží všemi účastníky.

Důvodů, proč se někteří lidé uchylují k tomuto způsobu nakupování, je hned několik, nicméně každý může být motivován jinou pohnutkou. Nejčastějším impulzem jsou peníze. Pakliže se totiž sejde kupř. desetičlenná skupina lidí se záměrem si zakoupit zásobu brambor na několik měsíců, může si ji objednat přímo od českého pěstitele, neboť např. 200 kg brambor pro každého participanta je pro zemědělce atraktivní poptávkou než pouhých 200 kg, které by požadoval jednotlivec. Zatímco si tak může stanovit vyšší cenu za kilogram, než mu nabídne výkupčí, pro komunitu může být nákup výhodnější, poněvadž v ceně brambor nebude zohledněna marže jednotlivých článků distribučního řetězce. Sloučení několika objednávek do jedné tedy pěstiteli zajišťuje vetší odbyt, čímž může kolektivu dát i množstevní slevu, a taktéž umožňuje, že se participanti sdíleného nákupu mohou rozdělit o náklady

za balné a dopravu. Ty však mnohdy ani neplatí, neboť je při dostatečně velké objednávce hradí obchodník.

Díky komunitnímu nakupování se lidem rovněž nabízí možnost si zvolit a i podpořit konkrétního výrobce či pěstitele. Záleží tedy pouze na úsudku nákupčích, jaké faktory budou při jeho výběru zohledňovat. S jeho volbou však úzce souvisí i další dvě motivace, které některé vedou k tomuto způsobu nakupování. První je možnost výběru kvalitnějších potravin pěstovaných ekologicky a udržitelně bez používání syntetických pesticidů a průmyslových hnojiv, protože se tak i běžnému spotřebiteli otevírají vrátka k biopotravinám, ale i k celozrnným či bezlepkovým moukám apod., jejichž cenu překupníci velmi často uměle nadsazují. Druhým podnětem je pro některé i snaha o redukci množství obalových materiálů, zejména plastových, neboť je možné producentovi na rozdíl u běžných nákupů v obchodních řetězcích sdělit, že si např. přejí zabalit jimi požadované zboží do znovupoužitelných papírových pytlů, z nichž si pak každý participant skupinového nákupu může později odsypat jím zakoupené množství.

Jako jedním z přínosů kolektivních nákupů je mnohými účastníky vnímáno i to, že šetří celkový čas všech participantů, neboť vše zařizuje jediný člověk. Ostatní členové komunity jsou zodpovědní pouze za to, aby mu sdělili, co chtějí zakoupit, následně si od něj svůj nákup převzali a zaplatili jej. Svůj čas si pak komunita může šetřit i tím, když si jednotlivci zakoupí větší množství jimi pravidelně používaných trvanlivých potravin. Tím si tedy vytvoří jejich zásobu na delší období, čímž pak nemusí příliš často nakupovat.

Za zmínku rovněž stojí i fakt, že předávání doručeného zboží jednotlivým účastníkům sdíleného nákupu zavdává důvod k tomu, aby lidé spolu častěji komunikovali i aby se častěji vídali. Tím tak vzájemně utužují staré a v případě přibyvších členů mohou i budovat nové vztahy.

2.2 Formy komunitního nakupování

Komunitní nakupování má několik podob. Může se vyskytovat kupř. ve vícegeneračních domácnostech, a to např. v situaci, kdy dospělý jedinec během svého nákupu vkládá do košíku i zboží, jež požadují jeho či partnerovi rodiče. V rámci rodinných příslušníků, ale i blízkých přátel se však mohou tvořit i neformální, dynamicky se měnící nákupní skupinky. Jejich vznik může být podnícen potřebou jedné osoby po nějakém produktu či produktech, které nabízí prodejce, u něhož má v plánu nakoupit jeho známý. Tyto skupinové nákupy jsou však často nepravidelné a mnohdy vyvolané jen zmínkou o tom, že někdo bude něco, případně někde kupovat.

V současné době však vznikají i skupiny lidí, jež jsou tvořeny relativně stálými členy, ovšem ne nutně vždy se jednotlivci při jejich ustavování navzájem znají. Podle toho se pak i liší chod a celá organizace jednotlivých komunit sestavených z náhodných, nicméně podobně smýšlejících lidí, kteří jsou si navzájem cizí, od skupin složených z rodinných příslušníků, přátel, kolegů z práce či sousedů. Základní podstata jejich vzniku u obou typů společenství však zůstává stejná. Tito lidé si totiž zakládají na více či méně pravidelných společných nákupech spotřebního zboží, tedy především potravin, někdy však i přírodních kosmetických, mycích a pracích prostředků, nebo na jiném způsobu obstarávání si konzumních předmětů než tradičními metodami.

Provozování sdílených nákupů komunitami tvořenými přáteli není příliš podrobeno specifickými pravidly, jejich fungování je tedy spíše neformální. Např. podle [33] se skupina Denisy Šimlové, což je česká blogerka píšící nejen o skupinovém nakupování, domlouvá na kolektivním nákupu až tehdy, když nějakému jejímu kamarádovi či kamarádce začne

docházet jistý produkt, který objednávají hromadně. V tom případě se dotyčná osoba ozve tzv. organizátorovi, jenž pak pomocí elektronické pošty kontaktuje ostatní, zda se chtějí ke společnému nákupu daného produktu přidat. Následně po nějakou dobu od jednotlivých zájemců shromažďuje, v jakém množství si jej chtějí zakoupit, a až poté u daného dodavatele jimi poptávané zboží objedná. Pokud prodejce nezajišťuje rozvoz, organizátor buď u něj zboží vyzvedne, nebo se domluví s jiným členem komunity, aby objednávku přivezl. Ta se vyloží u organizátora doma a ten posléze informuje participanty kolektivního nákupu, že si mohou svoji část ze společné objednávky vyzvednout. K informaci též připojí cenu jejich podílu včetně poměrné hodnoty za sdílenou dopravu.

Organizátorem takovýchto skupin však nutně nemusí být pouze jedna osoba. V této roli se mohou jednotliví členové střídat, neboť samotná organizace hromadných nákupů může být časově náročná. Organizátor si totiž musí udržovat přehled o tom, co kdo chce zakoupit, v jakém množství jednotlivé produkty participanti sdíleného nákupu požadují, jestli mu již jimi objednané zboží zaplatili atp. Dále může sbírat i požadavky na nákup (kupř. zda mají být poptávané produkty lokální, jestli mají být pěstované bez používání průmyslových hnojiv a chemických postřiků, zda prodejce pečuje o hospodářská zvířata v souladu se zákonem na ochranu zvířat proti týrání atp.), dle nichž pak vybírá vhodného dodavatele. Taktéž se může s jednotlivými účastníky domlouvat zvlášť na místě a čase vyzvednutí jimi objednaného zboží, nicméně může i všem nabídnout pouze jeden společný či několik termínů předání.

Společenství tvořené náhodně seskupenými lidmi jsou obvykle tzv. alternativní potravinovou sítí, jíž je komunitou podporované zemědělství (viz podkapitola 2.2.1). V rámci této iniciativy spolu určitým způsobem provozují skupinové nákupy, přičemž fungování takovéhoto kolektivu je podřízeno určitými pravidly, ovšem ta se mohou u každé komunity lišit. Např. dle nasbíraných odpovědí na jednotlivé otázky dotazníku (viz příloha A), jenž byl rozeslán mezi několik lidí a jejž vyplnilo osmnáct osob, kteří komunitně nakupují, některá seskupení po žadateli o členství vyžadují, aby uvedl jméno člověka, jenž je členem kolektivu a jenž by se za něj mohl jistým způsobem zaručit. Tito lidé pak spolu nejčastěji komunikují v uzavřených skupinách na sociálních sítích a některé aktivnější komunity zde plánují kolektivní nákupy. Jednotlivé produkty, které si pak chtějí objednat, často vepisují do sdíleného tabulkového souboru. Objednané zboží pak leckdy jednotlivci organizátorovi platí při jeho převzetí, je-li však dražší, platí mu jej předem, a to prostřednictvím bankovního převodu.

Speciálními komunitami, jež jsou tvořeny různými, sobě navzájem cizími lidmi a jež nezapadají do kontextu iniciativy komunitou podporované zemědělství, jsou skupiny zakládané osobami, které chtějí s dalšími zájemci nakupovat na internetovém komunitním farmářském tržišti, o němž pojednává samostatná podkapitola 2.2.2.

2.2.1 Komunitou podporované zemědělství

Komunitou podporované zemědělství (dále jen KPZ) je systém produkce a konzumu potravin [21], který je postavený na úzkém vztahu mezi skupinou spotřebitelů a jedním či několika producenty [13] bez dalších zprostředkovatelů [14]. Jinak řečeno, jedná se o oboustranně výhodné partnerství [21] založené na vzájemné důvěře obou stran [15], jež zemědělci po určitou dobu zajišťuje odbyt vypěstovaných či vyrobených produktů a odběratelům dává jistotu v pravidelné dodávce čerstvých [21], lokálních a zpravidla ekologicky šetrně pěstovaných potravin [14].

KPZ není postavené jen na pasivní spotřebě poživatin, klade totiž důraz na demokratický potravinový systém řízený nejen producentem, ale i jeho zákazníky. Ti se stávají

částečně či zcela aktéry, nebo podílníky produkce konkrétního zemědělce, případně zemědělců [15], přičemž, mají-li zájem, mohou si část svého podílu ze sklizně odpracovat [21]. Díky tomuto systému tak mohou spotřebitelé aktivně ovlivňovat i kontrolovat samotné hospodaření [15], mohou tedy určit, jaké potraviny a jakým způsobem se budou pěstovat [21], čímž pak konzumenti nepotřebují a ani po dodavateli nevyžadují žádnou formální BIO certifikaci.

Tento potravinový systém si tedy zakládá na kooperaci všech zúčastněných a na jejich solidaritě [14]. Ta se projevuje tak, že se aktéři, eventuálně podílníci dělí společně s daným pěstitelem nejen o bohatství sklizně, ale rovněž s ním sdílí riziko neúrody [21]. Často je solidarita spotřebiteli vyjádřena i tím, že zemědělcům zaplatí svůj podíl před danou sezónou, přičemž výše platby je nastavena tak, aby jim zajistila důstojné žití. Tím tedy mají zajištěný nejen odbyt na celou sezónu, ale i výdělek [14].

V praxi je možné se setkat s různými typy KPZ, ty však obvykle vychází z jednoho z následujících dvou modelů. Prvním z nich je tzv. *Formální neziskový model*, jehož podstata spočívá v tom, že si skupina lidí, tj. komunita, založí spolek, případně družstvo, za účelem společného opatření potravin. Spolek pak následně může sám jídlo produkovat, a to tak, že za společné peníze získané z členských příspěvků zaměstnává zemědělce, který pro něj hospodaří, nebo jej nakupuje od předem domluveného producenta. Společně nabyté jídlo si pak členové komunity přerozdělují, čímž tedy v rámci spolku nejde o obchod, poněvadž zboží není předáváno žádné třetí straně.

Druhý a i rozšířenější model KPZ je tzv. *Neformální informační model*, ve kterém skupina lidí nevystupuje jako jedna osoba, tedy nezakládá žádný spolek, nýbrž obchodní vztah mezi každým konzumentem a daným zemědělcem probíhá na předem sjednaném místě zvlášť, formálně bez zásahu žádné externí třetí strany. Na toto místo, často označované jako výdejní, pravidelně po celou sezónu vozí zemědělec svoji produkci, již předává jednotlivcům neformální skupiny. Do obchodního vztahu mezi spotřebitelem a pěstitelem však mohou vstoupit jiní členové KPZ, kteří danému prodejci vypomáhají s výdejem zboží, tento zásah je ovšem chápán jako sousedská výpomoc.

KPZ tedy není forma podnikání, jedná se o komunitu, tedy skupinu lidí, již může koordinovat nějaká osoba. Jelikož je však tato činnost časově náročná, může být za ni jistým způsobem ohodnocena – v případě *Formálního neziskového modelu* může být zaměstnancem spolku, jejíž mzda je vyplácena z obdržených členských příspěvků, kdežto u *Neformálního informačního modelu* může od daného zemědělce obdržet odměnu kupř. ve formě naturálií, na niž se skládají jednotliví členové KPZ [15].

Pro lidi, kteří by se chtěli stát členy nějaké KPZ, však nemusí být lehké nalézt nějakou fungující komunitu, neboť u některých je již kapacita pro přerozdělení objemu produkce mezi jednotlivé participanty naplněna a rovněž ne vždy se v jejich lokalitě nachází nějaká již zaběhlá skupina. Tato společenství se totiž často vyskytují ve větších městech, nebo v jejich okolí a vytvoření nové může vyžadovat mnoho času i úsilí. Pokud pak komunita odebírá výrobky od více producentů, každý si může určit jiný den výdeje potravin, čímž si lidé nemusejí přebírat své jednotlivé podíly najednou. Je-li naopak nějaká skupina soustředěna pouze kolem jednoho výrobce či pěstitele a společně tedy neprovozují hromadné nákupy, získávají od něj jen určitý sortiment potravin, který daný výrobce či zemědělec produkuje, takže jiné jimi poptávané zboží musí každý shánět jiným způsobem. Celkové množství jimi odebíraných výrobků se pak odvíjí od velikosti sklizně a od počtu podílů, jež si od něj jednotlivci na začátku sezóny zakoupí, nicméně ne všichni si mohou dovolit pěstiteli naráz uhradit větší finanční částku. A konečně při výdeji potravin všichni podílníci obdrží

stejnou skladbu produktů, nemohou si tedy vybrat jen ty potraviny, o nichž vědí, že opravdu spotřebují.

2.2.2 Internetové komunitní farmářské tržiště

V roce 2017 byla Kamilem Demuthem založena pilotní verze internetového komunitního farmářského tržiště [25], jehož cílem bylo zlepšit dostupnost kvalitních, čerstvých, lokálních potravin za přívětivou pořizovací cenu, v níž nebudou promítnuty náklady velkoobchodů a distributorů. Rovněž měla podpořit malé i začínající farmáře a výrobce [26] hospodařící udržitelným, ideálně ekologickým způsobem [16], kteří často při klasickém obchodu za jejich zboží nezískají jim odpovídající finanční částku [17], a omezit plýtvání potravin, jež prodejci budou sklízet, případně vyrábět až v okamžiku přijetí objednávky [26]. Tato pilotní verze byla v březnu roku 2018 ukončena, nicméně v tutéž dobu byla nahrazena ostrou verzí s názvem *Scuk.cz* a již po roce jejího fungování se Kamil Demuth umístil na třetím místě v soutěži *Startupper roku 2019* [25].

Jak je výše uvedeno, internetové tržiště podporuje předem prověřené malé, ale i střední farmáře a výrobce hospodařící udržitelným způsobem [16], nicméně není po nich vyžadována BIO certifikace, avšak snaží se o to, aby vlastnili alespoň tzv. IPZ certifikát[1] [25]. Těmto prodejcům pomáhá tím, že jim umožňuje nabízet produkty přímo koncovým zákazníkům bez dalších prostředníků [16], tedy bez žádných obchodních řetězců a distributorů [26]. Sami si tedy mohou spravovat své nabídky, přičemž určení cen jimi nabízeného zboží je plně v jejich režii [16], čímž za ně získají adekvátní finanční ohodnocení [17]. Za jeho používání pak neplatí žádný mnohdy jim cenově nedostupný pronájem místa, jak je tomu u běžných farmářských trhů, přestože nemají jistotu určitého zisku [25], nicméně z každého uskutečněného obchodu si *Scuk.cz* účtuje provizi [18].

Komunitní princip internetového tržiště se v nákupním procesu projevuje tak, že jsou do něj zapojeni i samotní spotřebitelé. Ti se seskupují dle místa bydliště do komunit [25], tj. do nákupních skupin, jež zakládá a spravuje jedna osoba, jíž je organizátor. Ten po získání alespoň třiceti členů [16] v pravidelných intervalech, obvykle jednou za týden či čtrnáct dní, spouští kolektivní nákupy [25] a vybírá pouze lokální prodejce, od nichž bude komunita nakupovat. Společný nákup pak zpravidla trvá tři až sedm dní. Po tuto dobu si každý člen může vybrat jakékoliv zboží od organizátorem zvolených farmářů a výrobců, které následně sám za sebe předem prostřednictvím internetu zaplatí [16]. Po ukončení sdíleného nákupu je zakoupené zboží v předem sjednaný den doručeno zdarma prodejci, případně dopravní logistikou internetového tržiště, na pevně zvolené výdejní místo, na kterém zakladatel skupiny předává jednotlivcům zboží, jež si objednali [18].

Výdejní místo zajišťuje již v době vytváření nákupní skupiny organizátor a je zcela na něm, kde jej zřídí. Musí však zajistit čistý, dobře přístupný i dostatečně velký prostor vybavený chladicím zařízením pro uchování chlazených potravin. Zboží na něj bývá dodáno buď den před jeho vydáváním, nebo v den předávání objednaných produktů, a tak organizátor musí být časově flexibilní. Taktéž musí počítat s tím, že je třeba zboží rozdělit dle jednotlivých objednávek členů nákupní skupiny před tím, než si ho půjdou vyzvednout. Tato činnost je tedy časově náročná, a tak ji nejčastěji vykonávají matky na mateřské či rodičovské dovolené, lidé pracující z domova, nebo podnikatelé [16] (např. majitelé zdravé výživy, kavárny s komunitním přesahem atp. [25]), kteří chtějí kolektivními nákupy nalákat nové

[1] Ochranná známka IPZ, jež značí integrovaný systém produkce zeleniny, zajišťuje, že vypěstovaná zelenina je produkována výhradně českými zemědělci za výrazně omezeného používání průmyslových hnojiv a pesticidů [12].

zákazníky [16], nebo rozšířit ve svých obchodech sortiment o čerstvé potraviny. Tito lidé ji však mohou provozovat i jako přivýdělek [25], poněvadž jsou za ni odměněni 7,5 % z hodnoty každé kolektivní objednávky. Ta získávají v kreditech, jež mohou použít pro další nákup, nebo jsou jim vyplacena, ovšem pouze za podmínky, že mají platné živnostenské oprávnění [16].

Internetové tržiště je úspěšné zejména u KPZ, komunitních zahrad a u lidí z malých měst, jimž umožňuje získat v malých obcích často nedostupné čerstvé potraviny (jako např. kozí sýr) i ekodrogerii [25]. Navíc díky principu společných nákupů prodejcům zajišťuje větší odbyt, ovšem aby se jim prodej vyplatil [16], mohou si určit, jakou mají mít jednotlivé objednávky minimální hodnotu a do jaké maximální vzdálenosti budou zboží komunitám dovážet [18]. Lidé tedy mohou nakupovat jen lokální produkty [25], jež neputují přes celou republiku, a rovněž tímto obchodem podpoří místní ekonomiku [17].

Přestože *Scuk.cz* zobrazí potencionálnímu zájemci dle místa jeho bydliště nejbližší skupiny, do nichž se může přidat, nemá zaručeno, že je daná skupina aktivní. Taktéž může mít problém s její lokalitou a i se stanoveným termínem vyzvednutí zakoupeného zboží, neboť výdejní místa jednotlivých seskupení se mohou nacházet v obcích, přes která zájemce nejezdí z práce domů a rovněž která nejsou cílem jeho obvyklých cest. Sice je možné na internetovém tržišti zakoupit zboží ze širší palety druhů a i od různých výrobců, nicméně záleží i na tom, v jaké lokalitě se daný člověk nachází. Mezi prodejci pak nemusí být ti, jež upřednostňuje a u nichž i pravidelně s přáteli nakupuje, a taktéž si jednotliví členové nákupní skupiny mohou vybírat produkty jen od těch výrobců, které vybere organizátor. Ten pak spouštěním jednotlivých nákupů určuje, jak pravidelně budou probíhat. Ty se však nemusí realizovat vždy, neboť se k nákupu nemusí připojit všichni členové kolektivu, čímž tak nemusejí splnit podmínky jednotlivých prodejců.

Kapitola 3

Vývoj mobilních aplikací pro operační systém Android

Mobilní operační systém Android je platforma, jejíž historie se započala psát v Kalifornii v říjnu roku 2003, kdy Andy Rubin, Rich Miner, Nick Sears a Chris White založili společnost Android Inc., jejímž cílem bylo vyvinout chytřejší mobilní zařízení. O dva roky později ji zakoupila společnost Google, nicméně Rubin se svým týmem dál pokračoval pod novým vedením ve vývoji nově vznikající platformy. Ta vychází z linuxového jádra [27], jež je volně dostupné jako otevřený software [22], což společnosti Google umožnilo operační systém nabízet různým výrobcům mobilních telefonů zdarma.

První chytrý telefon s tímto operačním systémem byl veřejnosti představen v září roku 2008 pod názvem *T-Mobile G1*, v některých částech světa byl však známý pod jménem *HTC Dream* [27]. O tři roky později bylo dle [24] koncovým uživatelům prodáno ze všech mobilních zařízení 36,4 % zařízení s operačním systém Android, což byl nejvyšší globální tržní podíl z celkového prodeje přenosných zařízení s jednotlivými tehdy dostupnými operačními systémy. Od té doby Android na globálním trhu s mobilními operačními systémy dominuje, a dokonce ve druhém kvartálu roku 2018 podíl v počtu prodaných kusů přenosných zařízení s Androidem dosáhl již 88 %.

Následující kapitola se zabývá vývojem mobilních aplikací právě pro operační systém Android. Nejdříve čtenáři přiblíží, z jakých základních komponent jsou aplikace tvořeny, posléze popíše životní cyklus jedné z nich, tedy aktivity, i tzv. fragmentů a taktéž bude vysvětleno, jakým způsobem jsou tyto komponenty spouštěny. V poslední části kapitoly bude čtenář obeznámen s doporučenou architekturou mobilních aplikací, v níž bude vylíčeno, jakým způsobem zjednodušuje knihovna *Room* práci s lokání perzistentní databází zařízení pro ukládání uživatelských dat aplikace, dále jak třída `LiveData` usnadňuje monitorování změn provedených nad těmito daty a nakonec jak knihovna *Retrofit* zajišťuje přístup aplikací k webovým službám za účelem výměny dat se vzdáleným serverem.

3.1 Základní komponenty aplikace

Mobilní aplikace pro Android mají na rozdíl od běžných aplikací pro stolní počítače komplexnější strukturu. Jejich výpočet nezačíná na začátku hlavní funkce `main()` a neběží jako jeden monolitický proces, jak je tomu u většiny desktopových aplikací. Typická Android aplikace je tvořena ze základních [1] a nepostradatelných komponent, jež jsou vstupními body, přes které systém či uživatel může do dané aplikace vstoupit. Ty běží v jejím pro-

cesu [2] a mohou být kdykoliv systémem ukončeny, neboť mobilní zařízení mají omezené systémové prostředky [1].

Celkem existují čtyři základní komponenty, přičemž každá se používá za jiným účelem a každá má svůj specifický životní cyklus, jenž definuje, jak se daná komponenta vytváří a jak zaniká. Těmito komponentami, jež jsou popsány v následujících podkapitolách, jsou *aktivity*, *služby*, *přijímače* a *poskytovatelé obsahu* [2].

3.1.1 Aktivity

Aktivita (angl. activity) je vizuální komponenta [2], jež poskytuje okno, do něhož se vykreslí uživatelské rozhraní. Toto okno typicky vyplňuje celou obrazovku mobilního zařízení, avšak může být i menší, nebo může překrývat jiná okna. Obecně tedy platí, že jedna aktivita reprezentuje právě jednu obrazovku [3], s níž může uživatel interagovat [2]. Aplikace však může obsahovat hned několik obrazovek, a tak je typicky jedna aktivita označena jako hlavní. Ta se pak objeví při spuštění aplikace a z ní, ale i z ostatních je možné zahájit činnost jiných aktivit [3].

Ačkoliv aktivity vytvářejí jednotné grafické uživatelské rozhraní dané aplikace, každá z nich je nezávislá na ostatních. Jednotlivé aktivity lze tak spustit i z jiných aplikací [2], čímž programátoři nemusí opakovaně vytvářet stejné funkce. Této skutečnosti lze tedy využít kupř. v situaci, když by chtěl uživatel do rozepsané textové zprávy přiložit nějakou fotografii, již ale potřebuje nejdříve pořídit. Z právě běžící aplikace tedy můžeme spustit aktivitu aplikace obsluhující fotoaparát, jež vykoná činnost odpovídající potřebě uživatele.

3.1.2 Služby

Služby (angl. services) [2] jsou nevizuální komponenty, které běží na pozadí a zpracovávají dlouhotrvající operace – kupř. zajišťují stahování dat z internetu, aniž by došlo k blokování interakce uživatele s aktivitou, nebo přehrávání hudby na pozadí, zatímco uživatel používá jinou aplikaci. Jak je uvedeno v [30], přestože služby postrádají grafické uživatelské rozhraní, mohou uživatele o svém aktuálním stavu informovat, a to pomocí oznámení (angl. notification) či bublinového dialogu (angl. toast) obsahujícího krátkou textovou zprávu, jež je uživateli zobrazena na několik vteřin, čímž nenarušuje jeho stávající práci.

3.1.3 Přijímače

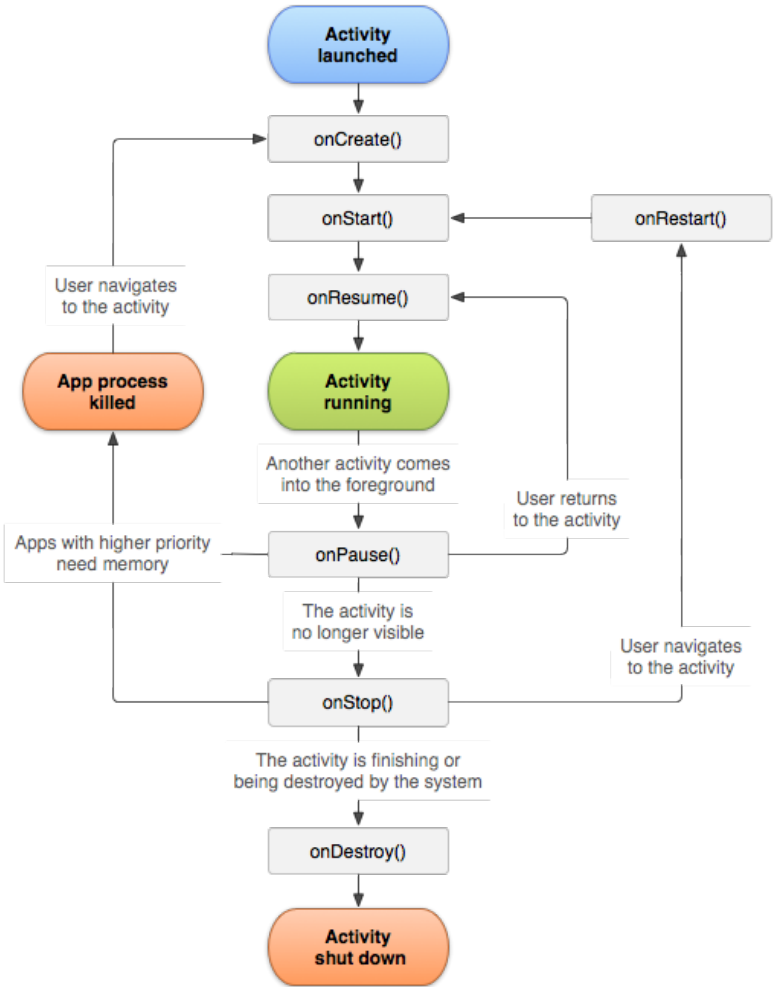
Přijímače (angl. broadcast receivers) [2] umožňují jednotlivým aplikacím mimo regulární uživatelský tok přijímat systémové události, ale i informace od jiných aplikací, popřípadě i od sebe samotné, na které pak mohou reagovat. Události, jimiž může být např. upozornění, že dojde k vypnutí obrazovky, nebo že je vybitá baterie, jsou doručovány i aplikacím, jež právě neběží. Tím si tedy mohou dopředu plánovat zobrazení notifikací, které uživatele upozorní na to, že se blíží nějaká událost.

3.1.4 Poskytovatelé obsahu

Poskytovatelé obsahu (angl. content providers) [2] spravují množinu dat aplikace, jež jsou uložena např. v souborovém systému či v lokálním perzistentním úložišti. Přes tyto komponenty mohou další aplikace k těmto datům přistupovat a měnit je, pakliže s nimi mají být sdílena.

3.2 Životní cyklus aktivity

Uživatel se může volně pohybovat mezi jednotlivými obrazovkami a taktéž může přecházet z jedné aplikace do druhé [4]. Tím tak dochází ke spuštění nových aktivit, přičemž předchozí jsou pozastaveny [32]. Aby bylo možné jednoznačně definovat, jak se má daná aktivita v okamžiku jejího vytvoření, pozastavení atp. chovat, každá má svůj vlastní životní cyklus, který reprezentuje všechny možné stavy, v nichž se může nacházet. Jakmile tedy nastane nějaká událost, resp. dojde-li ke změně stavu dané aktivity, systém spustí jemu odpovídající metodu. Ta zajistí, že aktivita na daný stav bude správně reagovat. Po jejím dokončení pak aktivita obvykle přejde do nového stavu. Jednotlivé přechody mezi stavy jsou znázorněné v diagramu, jenž je možné zhlédnout na obrázku 3.1, a metody, jež jsou po změně stavu systémem volány, jsou popsány v následujících odrážkách:



Obrázek 3.1: Obrázek ilustruje životní cyklus aktivity, jenž představuje všechny možné stavy, ve kterých se může daná aktivita vyskytovat. Aktivita tedy může být buď vytvořená, zahájená, běžící (tj. obnovená), pozastavená, zastavená či zničená. Převzato z [4].

- `onCreate()` – Metoda je spuštěna, jakmile systém poprvé vytvoří danou aktivitu [4]. Ta se nachází na pozadí, je tedy stále zastavená a pro uživatele neviditelná. V metodě tak programátor obvykle definuje její uživatelské rozhraní a vytváří instance tříd, které jsou pro její běh nezbytné [28].
- `onStart()` [4] – Tato metoda slouží pro finální nastavení aktivity. Volá se tedy před tím, než se stane pro uživatele aplikace viditelnou.
- `onResume()` [4] – V tomto stavu je aktivita v popředí a interaguje s uživatelem aplikace. Zůstává v něm tak dlouho, dokud nenastane nějaká událost, jíž může být např. přijetí telefonního hovoru, vypnutí obrazovky mobilního zařízení či spuštění nové aktivity.

- `onPause()` – Aktivita se nachází v pozadí [4], avšak může být stále viditelná. V tomto okamžiku může probíhat aktualizace uživatelského rozhraní, nicméně tento stav častěji indikuje, že uživatel danou aktivitu opouští [3]. Tato metoda se tedy používá k pozastavení operací, které by zde neměly pokračovat v činnosti [4].
- `onStop()` [4] – Metoda je volána, pakliže není aktivita uživateli delší dobu zobrazena. Používá se tedy pro uložení dat do perzistentního úložiště a k uvolnění zdrojů, jež nejsou potřebné a jež mohou být výpočetně náročné.
- `onRestart()` [3] – Volá se, jakmile má být znovu obnovena činnost již zastavené aktivity.
- `onDestroy()` [4] – Metoda je systémem volána tehdy, má-li aktivita zaniknout. Před její destrukcí je tedy třeba v ní uvolnit veškeré prostředky, které aktivita používala a které nebyly dříve uvolněny v metodě `onStop()`.

3.3 Fragment a jeho životní cyklus

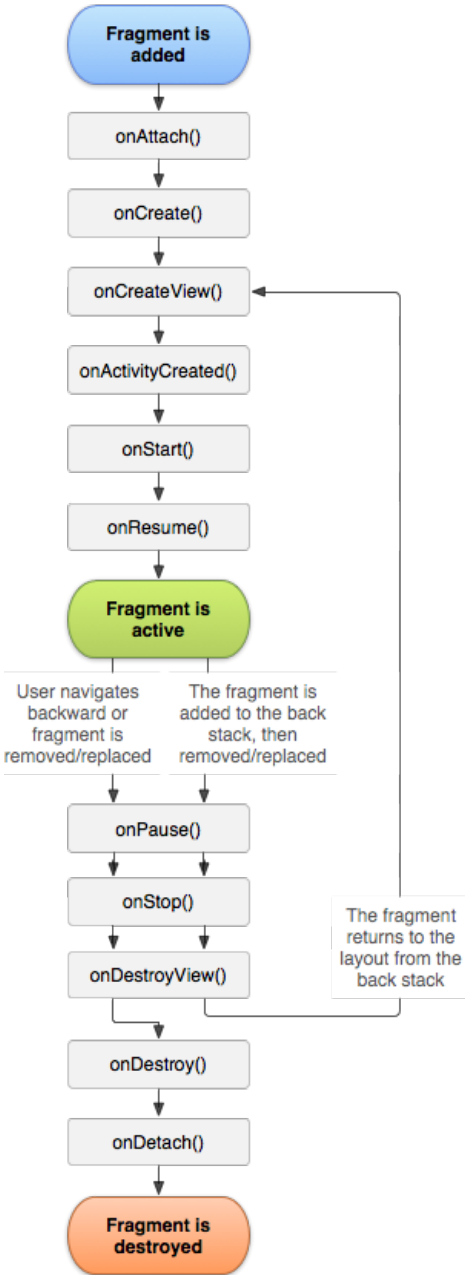
Fragment [5] je stejně jako aktivita vizuální komponenta Android aplikací. Na rozdíl od aktivit však tvoří pouze část uživatelského rozhraní, jež nemůže být zobrazena samostatně. Vkládá se tedy do jedné, případně i do několika aktivit, přičemž v každé může být umístěn libovolný počet různých fragmentů, ovšem nemusí obsahovat žádný.

S každým fragmentem je možné za běhu aplikace nezávisle manipulovat. Do hostitelské aktivity lze tedy přidávat nové, nebo z ní mazat stávající fragmenty, čímž se uživatelské rozhraní stává dynamickým a flexibilním. Manipulovat s fragmenty je však možné pouze v případě, že je hostitelská aktivita spuštěna, tj. musí se nacházet v popředí a musí interagovat s uživatelem aplikace.

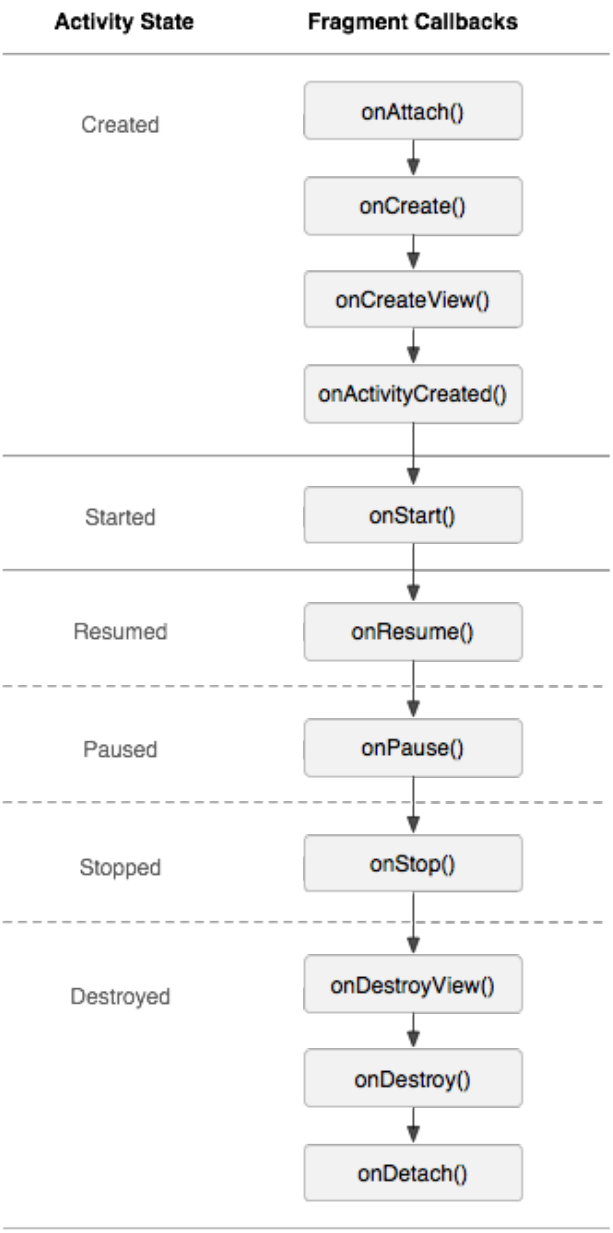
Stejně jako aktivita i fragment má svůj vlastní životní cyklus. Ten je však přímo ovlivněn životním cyklem jeho hostitele. Pokud totiž systém zavolá nějakou metodu životního cyklu aktivity, bude taktéž zavolána jí odpovídající metoda (případně i několik metod) životního cyklu fragmentu. Jestliže tedy bude pozastavena nějaká aktivita, budou taktéž pozastaveny všechny její fragmenty. Životní cyklus fragmentu se může nezávisle na aktivitě měnit pouze tehdy, je-li jeho hostitel spuštěn, pak lze tedy do něj fragmenty přidávat, resp. je z něj odebírat.

Jak je možné vidět na obrázku 3.2a, životní cyklus fragmentu se mírně liší od životního cyklu aktivity, neboť obsahuje nějaké metody navíc. Ty však reprezentují tytéž stavy (viz obrázek 3.2b), v nichž se může aktivita nacházet. V následujících odrážkách jsou rozepsané jen ty metody, které nejsou obsaženy v životním cyklu aktivity, neboť jsou si velmi podobné.

- `onAttach()` – Metoda je volána, jakmile je fragment asociován s aktivitou.
- `onCreateView()` – Zajistí vykreslení uživatelského rozhraní fragmentu.
- `onActivityCreated()` – Je spuštěna poté, co se dokončí metoda `onCreate()` životního cyklu aktivity.
- `onDestroyView()` – Metoda je volána za účelem odstranění všech prvků uživatelského rozhraní fragmentu.
- `onDetach()` – Zajišťuje odpojení fragmentu od aktivity.



(a) Obrázek znázorňuje životní cyklus fragmentu a je převzat z [5].



(b) Obrázek demonstruje, jak je tok životního cyklu fragmentu ovlivněn jednotlivými stavy hostitelské aktivity, a je převzatý z [5].

Obrázek 3.2: Na levém obrázku je zobrazen životní cyklus fragmentu. Ten je ovlivněn změnami stavů aktivity, jíž je součástí. Změní-li totiž aktivita svůj stav, systém nejenže zavolá jemu odpovídající metodu životního cyklu aktivity, ale taktéž budou vykonány i patřičné metody životního cyklu fragmentu. Které metody životního cyklu fragmentu jsou volány po změně stavu hostitelské aktivity, pak znázorňuje obrázek vpravo. Pokud tedy aktivita přejde kupř. do stavu *Created*, tj. bude-li vytvořená, systém zavolá první čtyři metody životního cyklu fragmentu. Jelikož je však možné do aktivity přidávat a i z ní odebírat jednotlivé fragmenty, je zapotřebí, aby se jejich životní cyklus mohl měnit nezávisle na ní. K jejich nezávislým změnám může dojít pouze tehdy, je-li jejich hostitelská aktivita spuštěna, tj. pokud je aktivita ve stavu *Resumed* [5].

3.4 Aktivace základních komponent aplikace

Jak již bylo zmíněno v kapitole 3.1.1, každá aplikace může spustit aktivitu, ale také službu či přijímač jiné aplikace. Tato komponenta se pak po její aktivaci spustí v procesu aplikace, jíž je součástí. Její aktivaci je třeba provést asynchronní zprávou, které se říká záměr (angl. intent). Ten se vytváří pomocí objektu `Intent` [2], jenž obsahuje informace, z nichž systém zjistí, která komponenta má být spuštěna a jaká data jí mají být předána.

Asynchronní zpráva, resp. záměr může být buď explicitní, nebo implicitní. Explicitní záměr obsahuje jméno balíčku cílové aplikace, nebo konkrétní název třídy komponenty, která má být aktivována. Obvykle se tedy používá v aplikacích ke spuštění komponent, jež vlastní [6]. Implicitní záměr specifikuje pouze typ komponenty [2], tj. deklaruje obecnou akci, kterou na základě uživatelského vstupu vykoná komponenta jiné aplikace.

Pokud aplikace používá implicitní záměry, systém musí dle informací obsažených v objektu `Intent` najít v mobilním zařízení vhodnou komponentu, jež vykoná požadovanou činnost. Tyto informace porovnává s tzv. filtry záměrů, tj. s výrazy, jenž jsou deklarované v souboru *AndroidManifest.xml*, který obsahuje každá aplikace. Jestliže se filtry, jež specifikují, jaké typy záměrů daná komponenta očekává, shodují s informacemi obsaženými v objektu `Intent`, systém tuto komponentu spustí a objekt `Intent` jí předá. Pakliže systém nalezne více komponent, jejichž filtry jsou se záměrem kompatibilní, zobrazí uživateli dialogové okno, ve kterém si může vybrat, která aplikace má být spuštěna. Pokud však nějaká komponenta aplikace nemá v souboru *AndroidManifest.xml* deklarované žádné filtry, může být spuštěna pouze pomocí explicitního záměru.

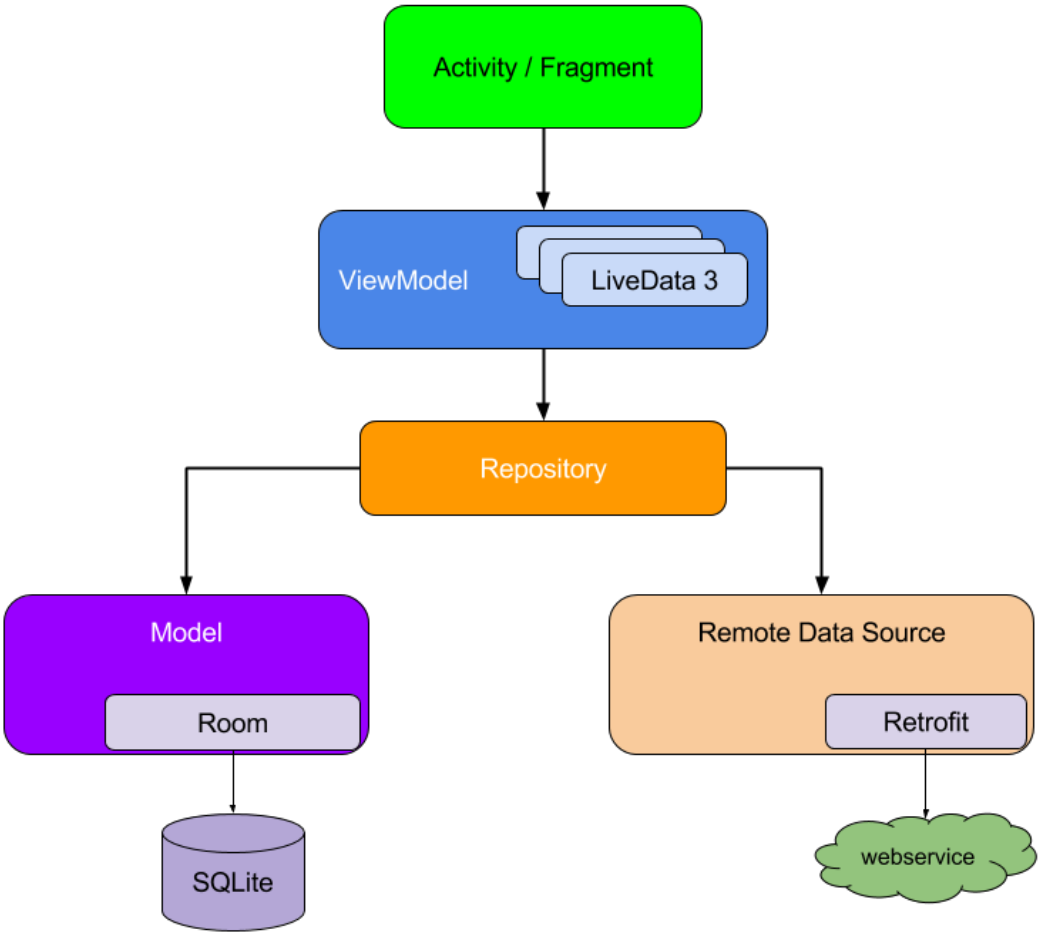
Pakliže je zapotřebí komponentě, jež má být spuštěna, předat nějaká data, je nutné použít jednu ze dvou metod objektu `Intent`, jimiž jsou `putExtra()` a `putExtras()`. Ty se liší v jejich použití, nicméně obě zajišťují postoupení požadovaných dat volané komponentě [6]. Postupovaná data mohou být jak primitivního, tak referenčního datového typu, ten však musí být serializovatelný, tj. musí implementovat rozhraní `Serializable` či `Parcelable`. Podle [28] rozhraní `Serializable` zajišťuje při transformaci objektů a jejich datové reprezentace dynamické rozpoznávání typů, zatímco druhé vyžaduje explicitní definici serializace a deserializace dané třídy. Přesto je však výhodnější používat právě rozhraní `Parcelable`, neboť je oproti rozhraní `Serializable` výrazně rychlejší. Jak je uvedeno v [6], serializované objekty však není vhodné používat u implicitních záměrů, poněvadž komponenty jiných aplikací, jež mohou být na základě přijetí asynchronní zprávy spuštěny, nemají přístup ke třídám implementujícím deserializaci předaných objektů.

3.5 Architektura aplikace

Uživatelé mobilních zařízení často v krátkém časovém horizontu interagují s více aplikacemi. Spuštěná aplikace může být totiž kdykoliv přerušena jinou (např. telefonním hovorem či notifikací), po jejímž ukončení uživatel očekává, že bude moci pokračovat v tom, co dělal před tím. Mobilní zařízení však mají omezené systémové zdroje, a tak mohou být jednotlivé komponenty aplikace spouštěny individuálně, a tedy nezávisle na ostatních, stejně tak mohou být kdykoliv operačním systémem zcela zničeny. Data, se kterými aplikace pracuje, by tedy neměla být v komponentách ukládána [1], neboť by mohla být v důsledku jejich zničení či znovuvytvoření ztracena [7].

Výše zmíněný problém zohledňuje doporučená architektura, jež umožňuje vytvářet robustní, dobře testovatelné a udržitelné mobilní aplikace pro operační systém Android [1]. Ji ilustruje obrázek 3.3, na němž jde vidět, že vychází z návrhového vzoru *Model-View-*

ViewModel. Její jednotlivé části jsou popsány ve výčtu, jenž se nachází pod následujícím obrázkem.



Obrázek 3.3: Obrázek znázorňuje doporučenou architekturu pro vývoj mobilních aplikací pro operační systém Android. Ta vychází z návrhového vzoru *Model-View-ViewModel*, kde *Model* je zodpovědný za uchování dat, se kterými aplikace pracuje, a vrstva *View*, jež je tvořena aktivitami a fragmenty, tato data zobrazuje [1]. Vrstva *ViewModel* pak dříve uvedené spojuje, tedy pomocí modulu *Repository* získává data, která mají být zobrazena, a do vrstvy *Model* propaguje žádosti o jejich změnu [29]. Obrázek byl převzat z [1].

- *Model* reprezentuje veškerou logiku aplikace [29] a je zodpovědný za uchování dat, s nimiž aplikace pracuje. Tento datový model, jenž by měl být nejlépe perzistentní (viz podkapitola 3.5.1), je nezávislý na vrstvě *View*, která je tvořena aktivitami a fragmenty, takže není ovlivněn jejich životním cyklem [1].
- *Remote Data Source* je vzdálený zdroj dat (např. databáze běžící na vzdáleném serveru).
- *Repository* [1] je modul, jenž provádí operace nad daty. Ví tedy, odkud má data získat, zdali např. z perzistentního úložiště, nebo ze vzdáleného zdroje dat.
- *ViewModel* [1] obsahuje logiku pro komunikaci se třídami vrstvy *Model*. Jeho jednotlivé části mohou volat třídy pro načtení dat, jež poskytují konkrétním komponentám

z vrstvy *View*, která je zobrazuje uživateli. Tyto komponenty však nezná, takže není ovlivněn změnami jejich konfigurace (např. znovuvytvoření aktivity po otočení obrazovky zařízení). Od nich získané žádosti o úpravu uživatelem pozměněných dat pak předává konkrétním třídám vrstvy *Model*.

- *View* je vrstva, jež je tvořená aktivitami, fragmenty a jim korespondujícími soubory ve formátu XML [1], představující uživatelské rozhraní aplikace [29]. Zobrazuje tedy data, jež získává od jednotlivých tříd vrstvy *ViewModel* [1].

  I když komponenty této vrstvy podléhají změnám konfigurace, znovuvytvořená komponenta obdrží tentýž objekt potomka abstraktní třídy *ViewModel*, který byl vytvořen při jejím prvním spuštění. Tento objekt je tedy automaticky zachován, dokud nedojde ke zničení komponenty, která jej používala. Pakliže tedy nastane krátkodobé přerušení aplikace, objekt potomka abstraktní třídy *ViewModel* zajistí, že nedojde ke ztrátě dat. Ta jsou po obnovení dříve přerušené aplikace okamžitě k dispozici, a tak mohou být uživateli ihned znovu zobrazena [7].

3.5.1 Knihovna Room a databáze SQLite

Jak již bylo na začátku kapitoly 3.5 řečeno, potřebuje-li operační systém uvolnit systémové prostředky, může zničit jednotlivé komponenty aplikace, ale taktéž ji může ukončit celou [1]. V tom případě dojde i ke zničení objektů tříd z vrstvy *ViewModel* doporučené architektury, které uchovávaly dočasná data aplikace. Z tohoto důvodu je nutné data někde trvale uložit. K tomu se používá knihovna *Room*, jež dle [8] tvoří abstraktní vrstvu nad lokální databází *SQLite*, k níž umožňuje jednoduše přistupovat. Ta zajišťuje, že aplikace má potřebná data k dispozici i v případě, že není dostupné síťové připojení.

Knihovna *Room* zajišťuje s minimalizací kódu mapování objektů do lokálního perzistentního úložiště, usnadňuje tvorbu databázových dotazů, jež jsou v době kompilace zdrojových kódů validovány překladačem, čímž zabraňuje tomu, že v průběhu běhu aplikace dojde k jejímu selhání, a taktéž umožňuje s využitím třídy *LiveData* (viz kapitola 3.5.3) sledovat, zda došlo ke změně uložených dat [1]. Tím tak lze zajistit, aby byla uživateli zobrazena aktuální data [9]. K tomu poskytuje tři komponenty, jimiž jsou *databáze*, *entita* a *DAO*. Ty jsou popsány v následujícím výčtu:

- *Databáze* [8] slouží jako hlavní přístupový bod k relačním datům perzistentní databáze. Je implementována abstraktní třídou rozšiřující třídu `RoomDatabase`, přičemž před její deklarací musí být uvedena anotace `@Database`. Třída uchovává seznam entit, jež jsou s databází spojeny, a obsahuje abstraktní metody, jejichž návratovým typem je rozhraní či abstraktní třída, které jsou před svojí deklarací anotovány výrazem `@Dao`.

- *Entita* představuje jednu databázovou tabulku [8], jež je reprezentována třídou datového modelu, tj. v architektuře aplikace se nachází ve vrstvě *Model* [1]. Její jméno ve výchozím nastavení určuje název databázové tabulky, jejíž jednotlivé sloupce odpovídají atributům této třídy. Ty mohou být veřejné, v tom případě tak není zapotřebí definovat její konstruktor i gettery a settery pro zpřístupnění či nastavení hodnot jednotlivých atributů.

  Tato třída musí být před svojí deklarací anotována výrazem `@Entity` a taktéž musí obsahovat nejméně jeden atribut reprezentující primární klíč tabulky. Pakliže má entita právě jeden takovýto identifikátor, je jemu odpovídající atribut anotován výrazem

`@PrimaryKey`. Do něj lze přidat vlastnost `autoGenerate`, jež zajistí, že knihovna *Room* automaticky vygeneruje unikátní hodnotu primárního klíče [10]. Ten figuruje ve vztazích mezi objekty jako cizí klíč, pomocí něhož se jeden objekt odkazuje na druhý. Knihovna *Room* tedy neumožňuje relaci reprezentovat pomocí přímého ukazatele na druhý objekt [30].

- *DAO* (neboli Data Access Objects) je rozhraní, nebo abstraktní třída s anotací `@Dao` [11], jež obsahuje deklarace metod zajišťujících přístup k databázi. Tyto metody z ní tedy umožňují získat potřebná data a uložit do ní nad nimi provedené změny [8]. Knihovna *Room* je kategorizuje do čtyř typů, které rozlišuje pomocí čtyř anotací. Tři z nich, jež jsou anotovány jedním z výrazů `@Insert`, `@Update` či `@Delete`, přijímají jeden či několik objektů tříd majících anotaci `@Entity` a nad příslušnou databázovou tabulkou v jedné transakci vykonávají jim odpovídající dotaz v jazyce SQL, který knihovna vygeneruje. Čtvrtý typ dotazovacích metod je anotován výrazem `@Query`, do jehož těla lze zadat libovolný dotaz v jazyce SQL. Jim lze taktéž předat parametry, jimiž mohou být např. identifikátory objektů, které mají být metodami vráceny [11].

3.5.2 Knihovny Retrofit a Gson

Retrofit je HTTP klient pro operační systém Android [20], jenž usnadňuje přístup k webovým službám REST API [23], a to tak, že generuje implementaci rozhraní API obsahující deklarace metod, které představují jednotlivé požadavky protokolu HTTP. Aby bylo zřejmé, jak mají být zpracovány, musí být všechny deklarace anotované jedním z výrazů `@GET`, `@POST`, `@PUT`, `@DELETE` či `@HEAD`, jež specifikují, jaká metoda protokolu HTTP má být v jednotlivých požadavcích použita. Každé volání metod deklarovaných v rozhraní pak vytvoří synchronní či asynchronní požadavek protokolu HTTP na vzdálený server, přičemž přesné umístění zdroje na vzdáleném serveru je dáno relativním URL uvedeným v parametru anotace [20].

Aby aplikace mohly jednoduše vzdáleným serverům odesílat své jednotlivé objekty a naopak, aby od nich mohly získávat potřebná data, knihovnu *Retrofit* [20] lze nakonfigurovat tak, aby používala pro serializaci a deserializaci objektů konkrétní konvertor. Za tímto účelem bude použita knihovna *Gson* [19], která zajišťuje převod objektů do jejich JSON reprezentace a i zpětnou konverzi dat z datového formátu JSON na jim ekvivalentní objekty.

3.5.3 Třída LiveData

Třída `LiveData` uchovává pozorovatelná data [9], čímž umožňuje registrovaným komponentám aplikace pomocí metody `observe()` monitorovat změny objektů, které zobrazují, a na ně patřičně reagovat. Třída respektuje jejich životní cyklus [1], a tak jsou o změnách dat informováni metodou `onChanged()`, pouze když jsou aktivní (tj. když jsou buď zahájené, nebo běžící). Komponenty aplikace tedy pouze sledují relevantní data, nemusí na základě změny svého stavu přerušovat, ani obnovovat pozorování objektů, neboť třída `LiveData` automaticky změny jejich stavu životního cyklu spravuje. Je-li tedy pozastavená komponenta obnovena, nebo zastavená znovuvytvořena, okamžitě obdrží nejnovější dostupná data. Pakliže je však zcela zničena [9], automaticky je třídou odregistrována, tj. odebrána ze seznamu pozorovatelů [1], čímž nedojde k úniku paměti [9].

Kapitola 4

Návrh mobilní aplikace pro sdílené nakupování

Ne každý člověk je součástí nějakého autonomního společenství vzájemně interagujících osob majících společné zájmy či podobné potřeby, s nimiž by mohl pravidelně nakupovat, ovšem každý jedinec může mít kolem sebe okruh lidí, kteří mohou upřednostňovat tentýž druh zboží (např. biopotraviny či organickou kosmetiku) či tytéž výrobce jako on sám, kteří mohou mít podobné zájmy (kupř. pečení domácího chleba), nebo kteří mu mají co nabídnout (např. přebytky ze zahrádky), avšak si třeba nejsou jisti tím, zda by o to měl zájem, nebo jen nemají kuráž někomu něco tváří v tvář nabízet. Každý má však kolem sebe jiný okruh lidí než jeho kamarád, soused či kolega z práce, jejž zná, s nímž se běžně setkává a jemuž důvěřuje, a proto si každý uživatel aplikace bude vytvářet svou vlastní skupinu lidí, tedy jakousi komunitu složenou např. z rodinných příslušníků, přátel, známých atp., kterým chce nabízet sdílené nákupy u různých prodejců či výrobců, nebo nějaké jeho produkty ke koupi – kupř. ony přebytky ze zahrádky, vejce, med aj. Bude-li tedy nějaký uživatel vědět, že bude zanedlouho potřebovat objednat kupř. krmivo pro jeho domácí zvíře, bude se tak moci jednoduše zeptat svých přátel, zda chtějí u daného prodejce taktéž něco zakoupit. Tím případně budou moci spolu sdílet náklady za poštovné i balné a rovněž organizátor zájemcům ušetří trochu času stráveného nad objednávkou. Ti mu pak mohou příště nabídnout společný nákup u jiného obchodníka, čímž se tak budou moci střídat v roli organizátora.

Aby se však uživateli mezi hromadou nabídek od jeho kamarádů neztratily ty, o něž by mohl mít skutečný zájem, budou svými tvůrci vkládány do určité míry obecných kanálů. Každý pak bude představovat jednu kategorii, nebo také soubor podobných, obvykle však současně neprobíhajících nabídek, který zajistí jejich filtraci. Kanály si pak bude dle vlastního uvážení vytvářet každý uživatel sám, přičemž jejich seznam bude viditelný pro všechny členy komunity jeho vlastníka. Ti si z něho budou vybírat jen ty kanály, jež budou pro ně relevantní.

Jelikož je tedy aplikace postavena na vzájemné interakci jedince s jeho skupinou, bude zapotřebí připojení k internetu, které je pro některé její funkce nezbytné. Při jejím návrhu se však vycházelo z toho, že ne pro všechny uživatele mobilních zařízení je celodenní internetové připojení samozřejmostí. Aplikace tak bude podporovat i off-line režim, čímž si uživatelé budou moci prohlížet nabídky svých kamarádů např. v trolejbusu při cestě domů z práce, stejně tak budou moci bez nutnosti připojení k síti vytvářet nové a editovat již existující kanály i nabídky na společné nákupy či jejich produktů ke koupi. Dále se při

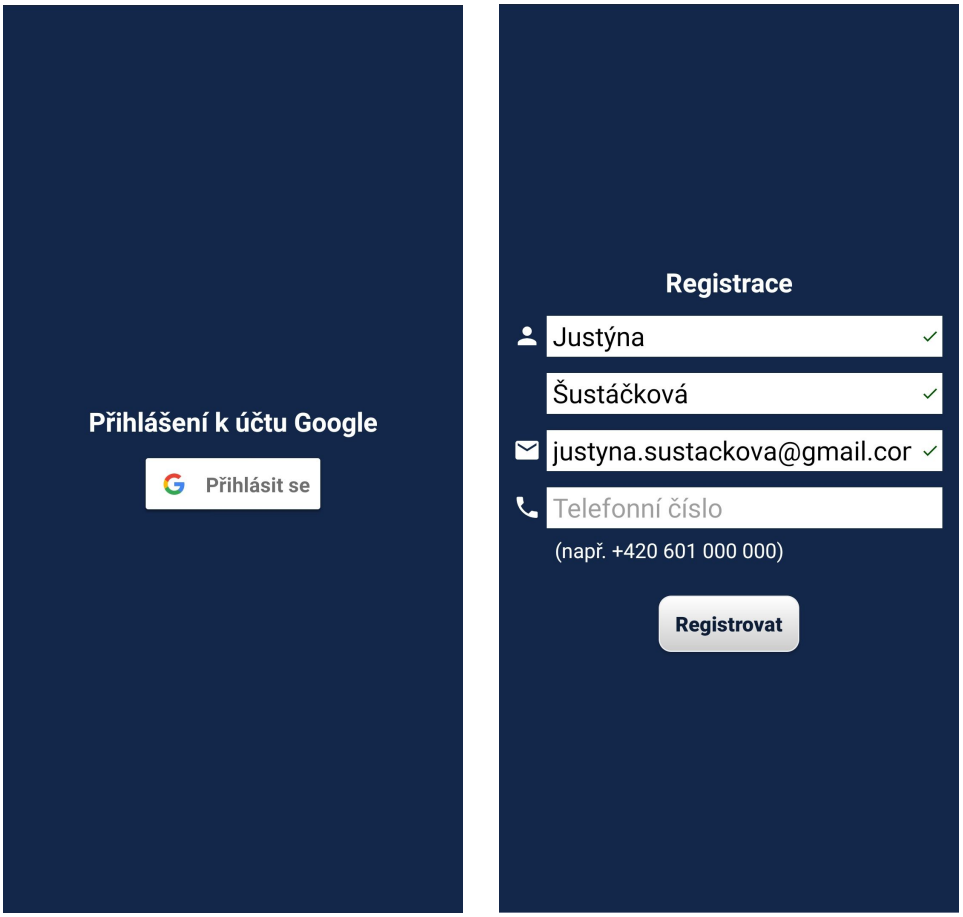
návrhu aplikace braly na zřetel odpovědi respondentů na jednotlivé otázky dotazníku, jenž je uveden v příloze A, a rovněž byla koncipována s ohledem na zachování soukromí jejích uživatelů i tak, aby jim zajistila, že budou provozovat sdílené nákupy jen s lidmi, které opravdu znají.

V této kapitole bude tedy čtenář podrobněji seznámen s konečným návrhem jednotlivých výše popsaných částí aplikace, jenž je doplněn snímky dílčích obrazovek výsledné aplikace, neboť se od prvotních mockupů v lecčems liší. Nejdříve mu tedy bude sděleno, jakým způsobem se budou potencionální uživatelé do aplikace registrovat, dále zjistí, jaké informace budou moci uvést do svého profilu a kdo jej uvidí, taktéž bude seznámen s tím, jak si uživatelé budou vytvářet své vlastní komunity i jakým způsobem uživatelé budou svým kamarádům a známým nabízet společné nákupy i jejich produkty ke koupi, a nakonec, tedy v poslední části kapitoly bude uveden model domény aplikace.

4.1 Registrace uživatele do aplikace

Jelikož by aplikace měla chránit soukromí uživatelů, neměla by tedy zveřejňovat jejich profily těm, kteří s nimi sdílené nákupy neprovozují. Aby však uživatelé mohli přes aplikaci snadno kontaktovat jejich příbuzné i kamarády za účelem nabídnutí jim vzájemné spolupráce, je vhodné zvolit takový identifikátor osob, pomocí nějž budou moci jimi požadované oslovit i bez toho, aniž by jej od nich museli pro tento účel explicitně získat. K tomu se nabízí dva možné identifikátory, jimiž jsou telefonní číslo a e-mailová adresa. Přestože telefonní číslo může být mezi lidmi častěji vyměňované, a tak se tedy může zdát být i jako vhodnějším kandidátem, ne vždy jsou lidé ochotni jej poskytnout pro ně ne zcela neznámým osobám (např. kolegům z práce či sousedům), ačkoliv s nimi mohou sdílené nákupy běžně provozovat. Tuto skutečnost potvrzují i odpovědi z dotazníku (viz příloha A), neboť právě e-mailová adresa byla respondenty častěji označována za údaj, jenž dávají, případně jenž by byli ochotni dát členům jejich komunity k dispozici.

Aby však nemohlo dojít ke zneužití identifikátorů jednotlivců, a tedy aby bylo možné uživatelům aplikace zaručit, že budou nabízet společné nákupy, eventuálně nějaké jejich produkty osobám, jež skutečně znají, nikoliv lidem, kteří se za ně jen vydávají, bude nutné při registraci každého uživatele do aplikace ověřit jeho totožnost. Tu lze v případě e-mailové adresy snadno verifikovat bez toho, aniž by bylo zapotřebí vygenerovat ověřovací kód a ten následně uživateli zaslat na uvedenou adresu, neboť společnost Google umožňuje do aplikací integrovat přihlašování k účtu Google, jenž má založený každý uživatel mobilního zařízení s operačním systémem Android, pakliže si do něj instaluje různé aplikace. Lidé se tedy budou muset před samotnou registrací do aplikace přihlásit prostřednictvím obrazovky z obrázku 4.1a ke svému účtu Google, čímž bude ověřena jejich totožnost na základě toho, co znají – tedy svoji e-mailovou adresu, která tak bude použita pro jejich identifikaci, a jí odpovídající heslo. Z každého účtu pak bude získáno jméno, příjmení i e-mailová adresa přihlášené osoby. Tyto údaje pak budou zobrazené v odpovídajících needitovatelných textových polích registrační obrazovky, již je možné zhlédnout na obrázku 4.1b. Může se však stát, že nějaká osoba ve svém účtu nemá své jméno a příjmení uvedené. V tom případě bude aplikace vyžadovat, aby tyto údaje vyplnila.



(a) Obrazovka pro přihlášení uživatele k jeho účtu Google.

(b) Obrazovka pro registraci uživatele do aplikace.

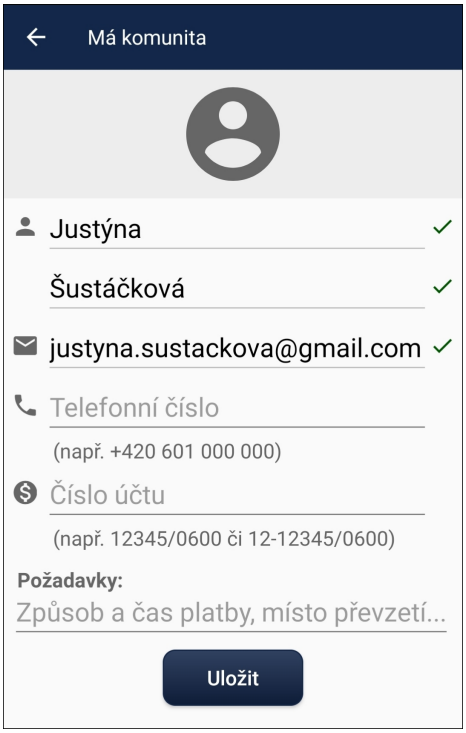
Obrázek 4.1: Na levém obrázku je zobrazena obrazovka pro přihlášení k účtu Google, jež zajistí autentizaci každé osoby před její registrací do aplikace. Pravý obrázek pak ilustruje registrační obrazovku, která bude po přihlášení dané osoby k účtu Google vyplněna jejím jménem, příjmením a e-mailovou adresou. Nebude-li mít však nějaká osoba ve svém účtu jméno a příjmení uvedené, pak ji aplikace upozorní červenými vykřičníky zobrazenými uvnitř prázdných textových polích, že musí tyto údaje vyplnit.

4.2 Nastavení profilu uživatele

Bude-li mít uživatel zájem, nebo bude-li potřebovat sdělit svým členům komunity doplňující informace o své osobě, bude moci do svého profilu (viz obrázek 4.2) uvést své telefonní číslo či číslo účtu, na něž mu budou moci příjemci jeho nabídek zaslat jím požadovanou finanční částku. Obě čísla však bude muset zadat ve formátu, jenž bude uživateli zobrazen pod odpovídajícími textovými poli.

Poněvadž z odpovědí v dotazníku, jenž je k nahlédnutí v příloze A, vzešlo najevo, že 83,3 % respondentů má při provozování skupinových nákupů stanovené obecné požadavky, resp. jakási pevně daná pravidla týkající se např. toho, do kdy a jakým způsobem mají participanti společných nákupů organizátorovi zaplatit jimi objednané zboží, případně, má-

li zřízené nějaké výdejní místo, kde si ho mohou převzít, uživatel tak bude moci do svého profilu uvést i tyto údaje.



Obrázek 4.2: Obrázek demonstruje obrazovku pro nastavení profilu daného uživatele aplikace, jenž bude viditelný pro všechny členy jeho komunity. Uživatel jim tak bude moci do odpovídajících textových polí napsat v pod nimi uvedeném formátu své telefonní číslo či číslo účtu, na které mu budou moci posílat platbu za jejich objednávky. Bude-li mít i nějaké obecné požadavky týkající se např. toho, jaký způsob platby za jimi objednané zboží preferuje, nebo kde si je mohou vyzvednout, bude moci tyto informace taktéž do svého profilu uvést.

4.3 Utváření komunity uživatele

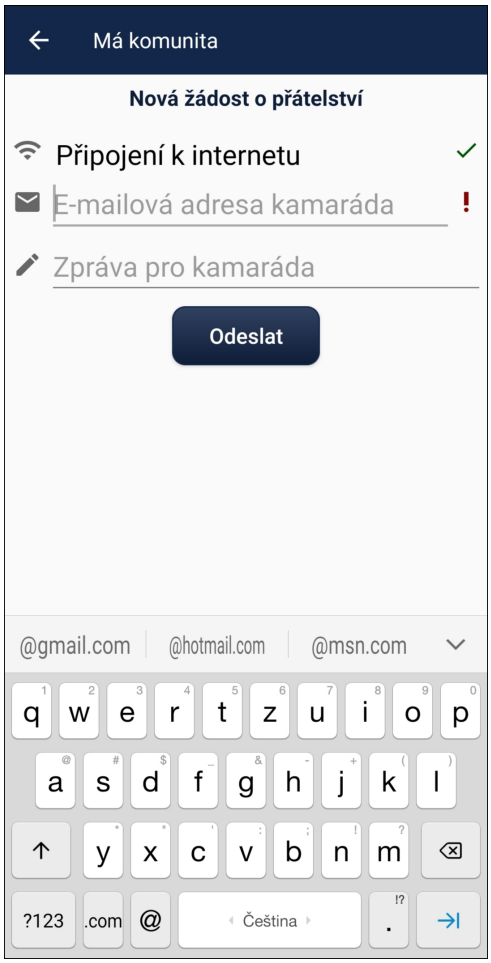
Pakliže si uživatel aplikace bude chtít přidat do své komunity nějakého svého kamaráda či známého, bude mu muset nejdříve zaslat žádost o přátelství prostřednictvím obrazovky, již je možné vidět na obrázku 4.3. Aby ji však mohl odeslat, bude muset znát jeho e-mailovou adresu, pomocí níž se do aplikace zaregistroval, jinak mu žádost nebude doručena. Do ní pak bude moci svému kamarádovi či známému napsat zprávu obsahující např. informace, proč si jej chce do své komunity přidat a jaké výhody by mu mohla vzájemná kooperace přinést. Aby však uživatel mohl dostat okamžitou zpětnou vazbu, zda mu bude možné žádost o přátelství odeslat, bude muset být jeho zařízení připojeno k internetu. Jestliže tedy omylem zadá nesprávnou e-mailovou adresu svého kamaráda či známého, bude okamžitě informován o tom, že se ji nepodařilo odeslat.

Pokud tedy bude uživatelův kamarád aplikaci používat, zobrazí se mu notifikace s informací, že jej tento uživatel žádá o přátelství, a taktéž se mu jeho jméno přidá do setříděného seznamu všech žadatelů, jenž je vyobrazen na obrázku 4.4a. Po kliknutí na položku seznamu, tj. na jméno jednoho z nich, se pak spustí nová aktivita, již ilustruje obrázek 4.4b. V ní uživatel uvidí základní údaje o příslušné osobě včetně zprávy, kterou mu v žádosti o přátelství napsala. Aby se však uživatel mohl informovat i o tom, jaké společné nákupy organizuje, nebo jaké produkty nabízí členům své komunity ke koupi, přepnutím přepínacího tlačítka se mu zobrazí seznam všech jejích kanálů. Po výběru jednoho z nich se pak objeví jeho popis (viz obrázek 4.4c).

Jestliže se uživatel rozhodne, že žádost o přátelství od dané osoby přijme, případně odmítne, vybere v nabídce voleb (angl. options menu) patřičnou položku, jež provede jím

požadovanou akci. Akceptuje-li tedy žádost od svého kamaráda, stane se členem jeho komunity a naopak, tj. i v seznamu všech členů uživatelovi komunity přibude nový, tedy jeho kamarád. Kliknutím na jeho jméno se mu pak zobrazí jeho profil (viz obrázek 4.5a) a přepnutím přepínacího tlačítka seznam všech jeho kanálů (viz obrázek 4.5b), z něhož si bude moci vybírat jen ty kanály, které jej zajímají. Jestliže se tedy uživatel bude chtít někdy v budoucnu připojit ke sdílenému nákupu, jenž opakovaně jeho kamarád s určitou mírou pravidelnosti organizuje, bude muset nejdříve začít patřičný kanál sledovat (viz obrázek 4.5c). V tom případě se mu začnou v jeho kanálu *Nabídky členů komunity* zobrazovat veškeré kamarádem zveřejňované nabídky z jím sledovaného kanálu a rovněž na nově vytvořené bude upozorňován notifikacemi.

Bude-li uživatel chtít ze své komunity odstranit nějakého člena, bude muset být jeho zařízení připojeno k internetu, aby bylo možné ověřit, že ani jeden z uživatelů nemá nějaký závazek k tomu druhému. Tj. pokud alespoň jeden z uživatelů akceptuje nějakou nabídku toho druhého a vyplní nákupní seznam, pak v ní bude muset být uvedeno, že ji i zaplatil a že si ji převzal.



Obrázek 4.3: Na obrázku je zobrazena obrazovka, pomocí níž si uživatel aplikace bude moci do své komunity přidat nového člena. Ten však do jeho komunity bude přidán až poté, co přijme uživatelem odeslanou žádost o přátelství. Ta bude muset obsahovat e-mailovou adresu, kterou uživatelův kamarád použije pro registraci do aplikace, a rovněž v ní bude moci být napsáno, jaké výhody může jeho známému vzájemná spolupráce přinést atp.

(a) Obrazovka se setříděným seznamem osob, kteří žádají daného uživatele aplikace o přátelství.

(b) Obrazovka se základními údaji o osobě, jež žádá daného uživatele o přátelství.

(c) Obrazovka se seznamem kanálů osoby, která zaslala danému uživateli žádost o přátelství.

Obrázek 4.4: První obrázek zdola ilustruje setříděný seznam všech osob, jež žádají daného uživatele aplikace o přátelství. Kliknutím na jednu z nich se pak uživateli zobrazí základní údaje o příslušném žadateli a zpráva, již mu do žádosti o přátelství napsal. Přepnutím přepínacího tlačítka si pak uživatel bude moci prohlédnout seznam všech jeho kanálů, který je vidět na vrchním obrázku. Vybere-li pak jeden z nich, objeví se mu jeho popis.

(a) Obrazovka s profilem člena komunity daného uživatele aplikace.

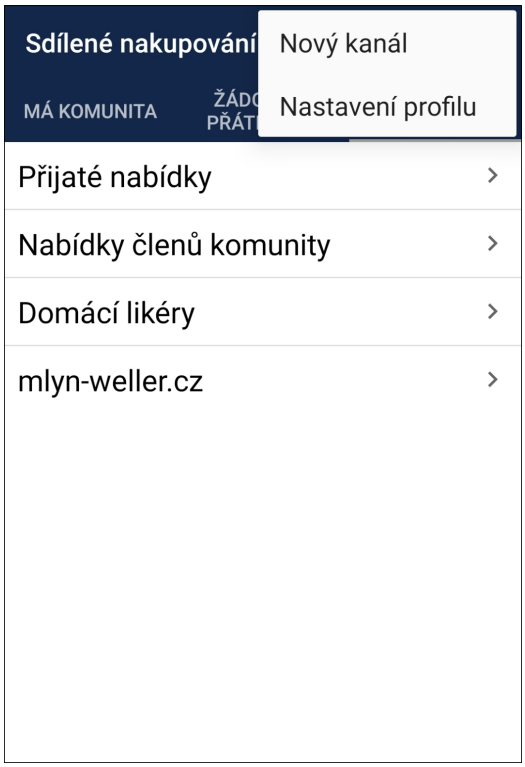
(b) Obrazovka se seznamem kanálů člena komunity daného uživatele.

(c) Obrazovka s popisem konkrétního kanálu člena komunity uživatele aplikace a s jeho požadavky týkajícími se prodeje jeho produktů.

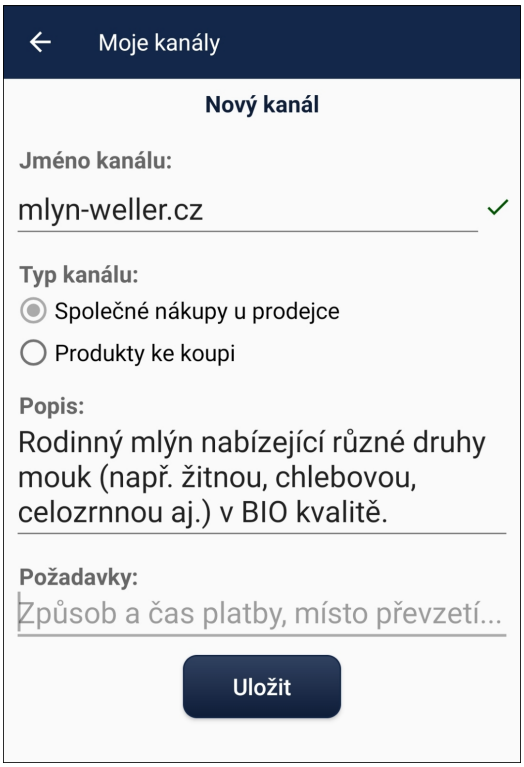
Obrázek 4.5: Na prvním obrázku zdola je zobrazen profil konkrétního člena komunity daného uživatele aplikace, z něhož bude moci vyčíst jeho telefonní číslo, číslo účtu a případně i jeho obecné požadavky na sdílené nákupy. Přepnutím přepínacího tlačítka si pak uživatel bude moci prohlédnout seznam jeho kanálů, v němž rovnou uvidí, které sleduje a které nikoliv. Výběrem jednoho z nich si pak bude moci přečíst jeho popis a eventuálně jej začít sledovat.

4.4 Správa kanálů a jejich nabídek

Jak již bylo v úvodu této kapitoly sděleno, každý uživatel si bude moci vytvářet své vlastní kanály. Jim budou dávat jména – jím může být např. jméno konkrétního prodejce, od něhož bude chtít uživatel aplikace hromadně nakupovat, nebo druh produktu, který bude chtít v daném kanálu svým členům komunity nabízet ke koupi – a rovněž jim budou přiřazovat jeden ze dvou možných typů. Do jednoho typu kanálu pak budou vkládat nabídky na společné nákupy, do druhého nabídky jejich produktů. Jak je možné vidět na obrázku 4.6b, uživatelé budou moci do svých kanálů uvést i popis daného prodejce či popis jimi nabízených produktů a případně i doplňující informace specifické pro daný typ nákupů. Po uložení nově vytvořeného kanálu se pak jeho tvůrci zobrazí v seznamu všech (viz obrázek 4.6a) a jeho členové komunity budou na nově vzniklý kanál upozorněni notifikací. Tím tak uživatelé aplikace nebudou muset pravidelně kontrolovat, zda nějaký jejich kamarád či známý nepřidal do svého seznamu kanálů nový, který by je mohl zaujmout a který by mohli chtít sledovat.



(a) Obrazovka se seznamem všech kanálů uživatele aplikace.



(b) Obrazovka určená pro tvorbu nového kanálu.

Obrázek 4.6: Na levém obrázku je zobrazen seznam všech kanálů uživatele, kde první dva budou obsahovat nabídky členů jeho komunity, které byly jimi vložené do uživatelem sledovaných kanálů. Zbylé kanály pak budou dle jejich jména seřazeny podle abecedy a budou obsahovat uživatelem zveřejněné nabídky. Pravý obrázek pak ilustruje obrazovku pro vytvoření nového kanálu, jenž musí mít jméno a typ, který určuje, jaký typ nabídek bude shromažďovat. Do vytvářeného kanálu též uživatel bude moci uvést popis prodejce, od něhož chce hromadně nakupovat, či jím nabízené zboží a rovněž bude moci svým členům komunity sdělit doplňující informace specifické pro daný typ nákupů.

Seznam všech kanálů uživatele bude seřazený dle jejich jmen podle abecedy s výjimkou prvních dvou, jež budou obsahovat nabídky jeho kamarádů a známých, které zveřejnili v jím sledovaných kanálech. Klikne-li pak uživatel na jeden z nich, zobrazí se mu seznam všech jeho nabídek, jejichž struktura se bude mírně odlišovat podle jeho účelu, případně i typu. V následujících podkapitolách jsou jednotlivé typy nabídek popsány.

4.4.1 Správa nabídek na sdílené nákupy

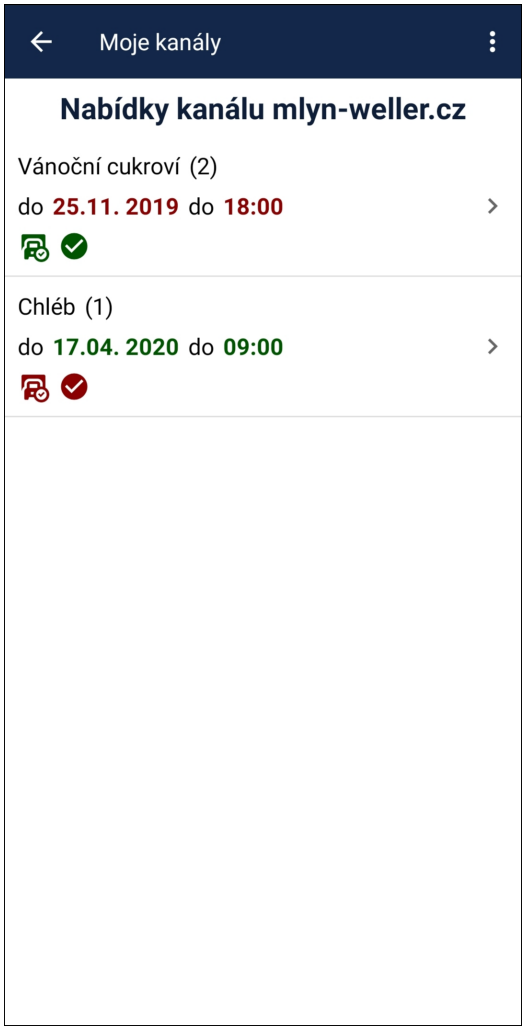
Aby uživatelé mohli snadno v kanálech rozlišovat své jednotlivé nabídky, budou je pojmenovávat, dále do nich budou moci uvést jejich popis a rovněž u nich budou určovat datum a čas, do kdy je mohou zájemci přijmout a vyplnit, co chtějí u daného prodejce zakoupit. Budou-li pak uživatelé potřebovat, budou v nich moci jejich příjemcům upřesnit informace uvedené v jim odpovídajících kanálech, tedy jakým způsobem a do kdy jim mají jimi objednané zboží zaplatit i kde a kdy si je mohou převzít. Organizátoři nákupu se tedy v aplikaci nebudou domlouvat na způsobu platby a převzetí s každým nakupujícím zvlášť, neboť většina dotazovaných v dotazníku (viz příloha A) uvedla, že tyto informace všem participantům sděluje hromadně.

Vytvoří-li tedy uživatel s pomocí obrazovky, kterou je možné si prohlédnout na obrázku 4.7b, novou nabídku, zobrazí se mu v patřičném kanálu mezi ostatními (viz obrázek 4.7a) a členové jeho komunity, již daný kanál sledují, budou o jejím vzniku informováni notifikací. Každá položka ze seznamu nabídek pak bude mít za jejím jménem v závorce uvedený počet kamarádů daného uživatele, kteří ji přijali, a rovněž mu bude zeleně zobrazovat datum a čas, do kdy ji budou moci zájemci přijmout. Bude-li však již příjem objednávek ukončen, datum i čas budou přebarveny na červenou barvu. Ikonky každé nabídky ze seznamu pak uživateli budou sdělovat, jestli již byl sdílený nákup doručen i jestli již byl ukončen. Mezi nimi však nebude taková, jež by uživatelům indikovala, zda již byl nákup objednán, či nikoliv, neboť většina respondentů v dotazníku (viz příloha A) uvedla, že uvedení této informace není nezbytné, a dokonce by nutnost jejího sdělení příjemcům nabídek mohla být organizátorem kolektivního nákupu vnímána za obtěžující.

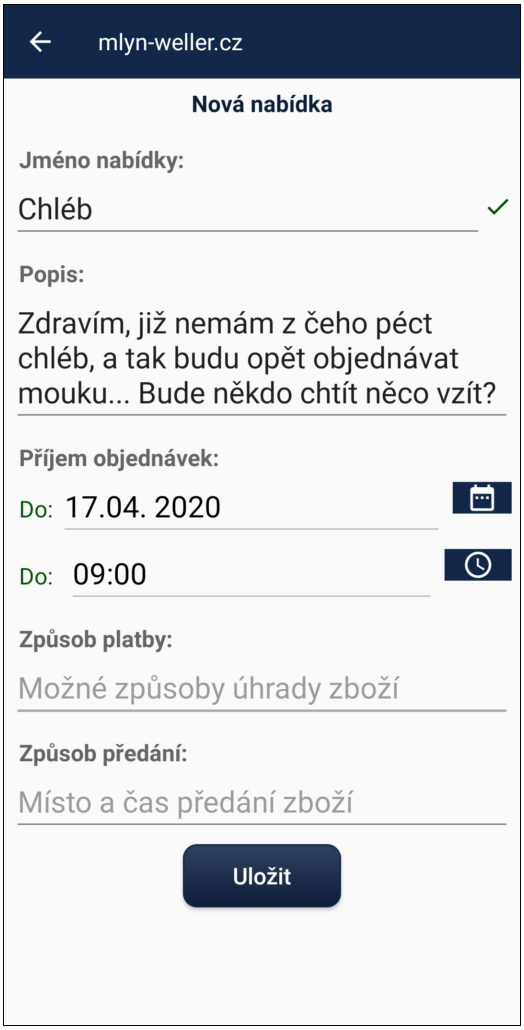
Zobrazí-li si pak uživatel soupis informací o nějaké své nabídce, v němž bude moci nastavit, že mu již byla hromadná objednávka doručena (viz obrázek 4.8a), přepnutím přepínacího tlačítka si bude moci prohlédnout seznam všech jejích příjemců, jenž je ilustrován na obrázku 4.8b. Každá jeho položka pak bude uvádět jméno konkrétního příjemce a rovněž bude obsahovat ikony, jež budou organizátorovi společného nákupu indikovat, zda již jeho kamarád či známý vyplnil nákupní seznam, jestli mu již objednané zboží zaplatil, jestli si je převzal a zda převzetí potvrdil. Kliknutím na jím požadovanou položku se pak organizátor bude moci podívat na nákupní seznam zvolené osoby (viz obrázek 4.8c), taktéž bude moci vyplnit cenu jejího nákupu, případně i dopravy, a uvést, zda mu již daná osoba nákup zaplatila a zda si jej převzala. O provedených změnách objednávky organizátorem pak bude její tvůrce informován notifikací.

Pakliže uživatel bude chtít svoji nabídku editovat, bude též moci změnit i datum a čas, do kdy ji bude možné zájemci o sdílený nákup přijmout a vyplnit. To však bude moci učit pouze v případě, bude-li nabídka stále otevřená, tj. pokud ji bude možné přijmout, a jestliže změnou data a času nezkrátí členům jeho komunity dobu pro její akceptování. Dobu ukončení příjmu objednávek bude tedy možné pouze odložit na později, aby zájemci nebyli o zkrácení doby pro její přijetí i vyplnění informování až po jejím uzavření. Nabídku rovněž organizátor společného nákupu nebude moci znovu otevřít, aby se nemohl případně vyhýbat předání zboží zakoupeného členy jeho komunity.

Nabídku na skupinový nákup bude uživatel moci odstranit pouze tehdy, bude-li označena za ukončenou, tj. bude-li mít již ukončený příjem objednávek a pokud všichni její příjemci, již vyplnili nákupní seznam, svoji objednávku organizátorovi kolektivního nákupu zaplatili a také pokud si ji převzali. Budou-li pak všechny nabídky daného kanálu ukončené, uživatel bude moci odstranit i ten.



(a) Obrazovka obsahující seznam všech uživatelem vytvořených nabídek na společný nákup u prodejce *mlyn-weller.cz*.



(b) Obrazovka určená pro přidání nové nabídky na společný nákup do kanálu *mlyn-weller.cz*.

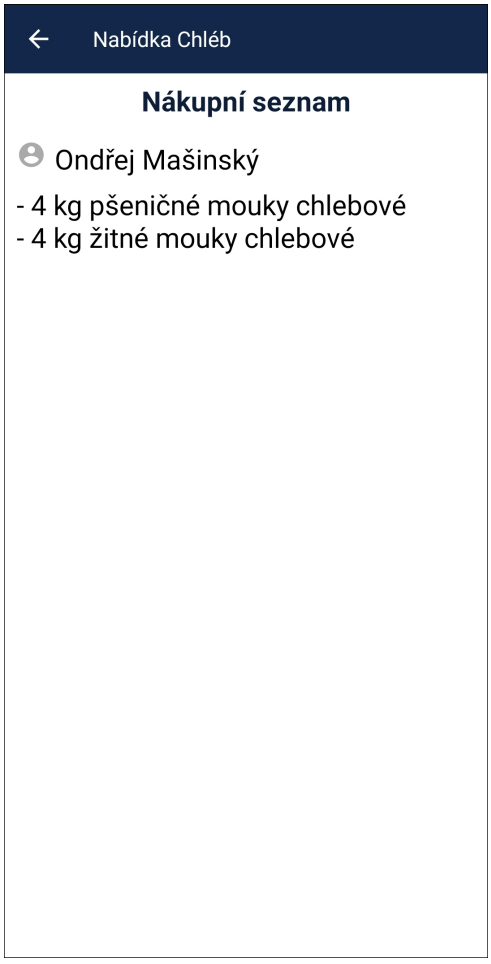
Obrázek 4.7: Na pravém obrázku je obrazovka pro vytvoření nové nabídky na společný nákup, která bude muset mít vyplněné jméno a datum s časem, do kdy ji budou moci zájemci o sdílený nákup akceptovat a vyplnit, co chtějí u daného prodejce zakoupit. Do nabídky uživatel bude rovněž moci uvést její popis či nějaký text, dále jaký způsob platby za objednané zboží jeho kamarády upřednostňuje a kde i kdy si je mohou převzít. Po jejím uložení se pak organizátorovi skupinového nákupu zobrazí v patřičném kanálu se seznamem všech jím vytvořených nabídek, jenž je vyobrazen na obrázku vlevo. Každá jeho položka bude mít za jejím jménem v závorce uvedený počet osob, které danou nabídku uživatele přijaly, dále bude obsahovat datum a čas, do kdy ji bude možné zájemci přijmout, a nakonec ikony. Ty budou indikovat, zda již byl nákup uživateli doručen a zda již byl ukončen.

(a) Obrazovka s informacemi o sdíleném nákupu u prodejce *mlyn-weller.cz*.

(b) Obrazovka se seznamem všech příjemců nabídky na sdílený nákup u téhož prodejce.

(c) Obrazovka s nákupním seznamem jednoho z příjemců nabídky na sdílený nákup.

Obrázek 4.8: Na prvním obrázku zdola je zobrazena obrazovka obsahující informace o uživatelem vytvořené nabídce na sdílený nákup, v níž bude moci nastavit, že mu již byla objednávka od daného prodejce doručena. Prostřední obrázek pak ilustruje soupis všech jejích příjemců, v němž uživatel také uvidí, jestli již jeho jednotliví kamarádi vypsali své nákupní seznamy, zda mu již jimi objednané zboží zaplatili, jestli si je převzali a zda jeho převzetí potvrdili. Na posledním, tedy horním obrázku je pak k vidění obrazovka s nákupní seznamem jednoho z příjemců nabídky, do které uživatel bude moci uvést cenu jeho nákupu, eventuálně i poměrnou část ceny dopravy, a nastavit, zda mu již jeho kamarád nákup zaplatil a jestli si jej převzal.



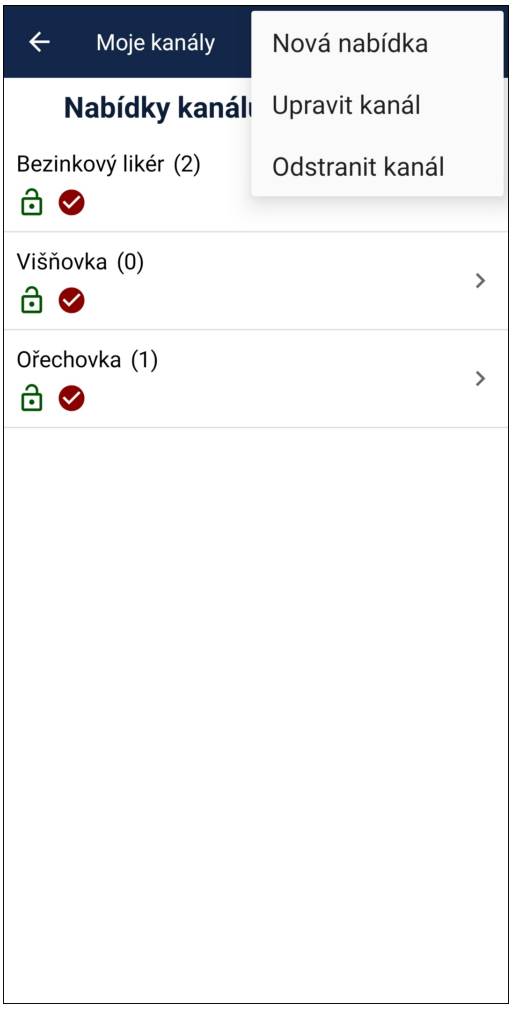
Obrázek 4.9: Obrázek demonstruje obrazovku zobrazující soupis jednotlivých nákupních seznamů všech příjemců nabídky na společný nákup u prodejce *mlyn-weller.cz*. Ta organizátorovi nákupu usnadní vytvoření hromadné objednávky u daného prodejce, neboť si nebude muset procházet jednotlivé nákupní seznamy jeho kamarádů zvlášť.

4.4.2 Správa nabídek produktů ke koupi

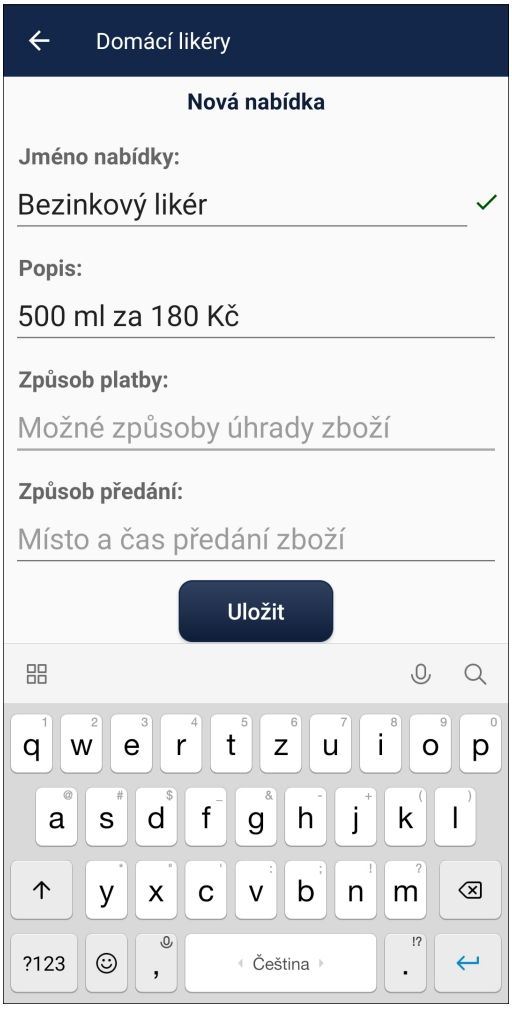
V prvotním návrhu aplikace jsem nikterak nerozlišovala jednotlivé nabídky na společné nákupy od nabídek produktů ke koupi. V pozdější fázi vývoje jsem však byla jedním z respondentů upozorněna, že nabízení např. domácích výrobků nebývá v reálném životě časově omezené, neboť je uživatelé mohou nabízet i dlouhodobě. Nabídky produktů na rozdíl od nabídek na společné nákupy tedy nebudou obsahovat datum a čas jejich uzavření, čímž tak uživatelé budou moci postupně jednotlivé objednávky od členů jejich komunity vyřizovat. Příjem objednávek pak prodejce bude moci explicitně ukončit (viz obrázky 4.10a a 4.11a), jakmile kupř. veškeré jím nabízené produkty jednoduše vyprodá. Pakliže však nějaký uživatel nabídku přijme před jejím ukončením, avšak prodávající již bude mít uživatelem objednané produkty vyprodané, objednávku tohoto člověka zruší, v opačném případě ji potvrdí (viz obrázek 4.11c).

Ukončí-li pak prodejce u nějaké své nabídky příjem objednávek, bude ji moci odstranit, pakliže všechny její objednávky, jež budou mít vyplněný nákupní seznam, jím budou

potvrzené či zrušené, přičemž potvrzené objednávky budou muset mít nastaveno, že byly kupujícím zaplacené a i převzaté.



(a) Obrazovka se seznamem všech uživatelem vytvořených nabídek produktů.



(b) Obrazovka určená pro přidání nové nabídky produktů ke koupi do kanálu *Domácí likéry*.

Obrázek 4.10: Na pravém obrázku je obrazovka pro vytvoření nové nabídky produktů, která na rozdíl od obrazovky pro přidání nové nabídky na společný nákup nebude obsahovat textová pole s datem a časem uzavření příjmu objednávek, neboť jednotlivé produkty mohou být uživateli nabízené dlouhodobě. Po uložení nové nabídky se pak prodejci zobrazí v kanálu *Domácí likéry* mezi ostatními nabídkami, jež jsou k vidění na obrázku vlevo. Stejně jako u nabídek na společné nákupy i zde bude mít každá položka seznamu za jejím jménem v závorce uvedený počet osob, kteří ji přijaly, a taktéž bude obsahovat ikonky, jež budou prodejci indikovat, zda již příjem objednávek uzavřel a zda již byla nabídka zcela ukončena.

(a) Obrazovka s informacemi o nabídce produktů ke koupi.

(b) Obrazovka se seznamem všech příjemců nabídky produktů ke koupi.

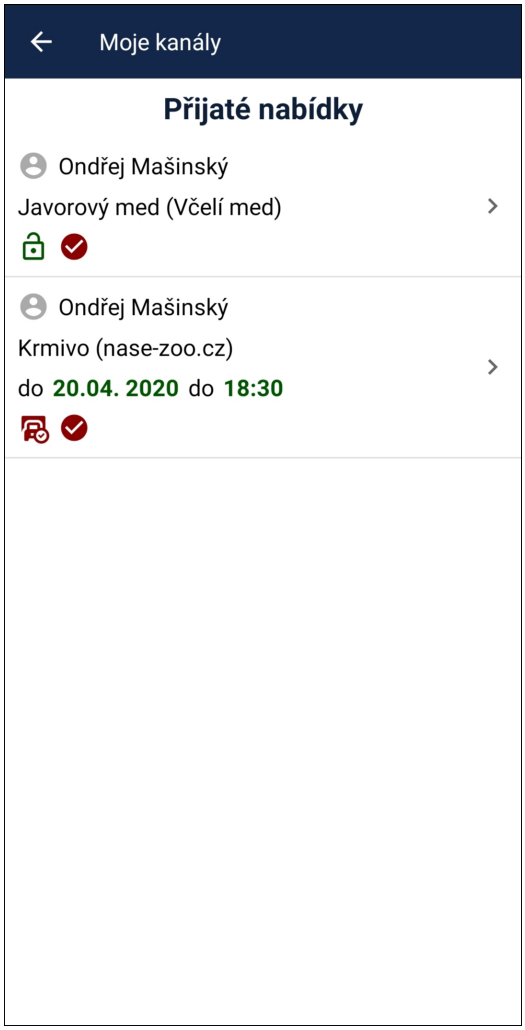
(c) Obrazovka s nákupním seznamem jednoho z příjemců nabídky produktů ke koupi.

Obrázek 4.11: První obrázek zdola ilustruje obrazovku obsahující informace o uživatelem vytvořené nabídce produktů ke koupi, ve které bude moci zastavit příjem nových objednávek od potenciálních zájemců. Prostřední obrázek ilustruje soupis všech jejích příjemců. Ten je podobný seznamu všech příjemců nabídky na sdílený nákup, jenž byl podrobněji vysvětlen u obrázku 4.8. Na posledním, tedy vrchním obrázku je pak možné zhlédnout obrazovku s nákupním seznamem jednoho z příjemců nabídky, ve které oproti obrázku 4.8c přibyla dvě přepínací tlačítka pro potvrzení, případně zrušení objednávky od daného příjemce.

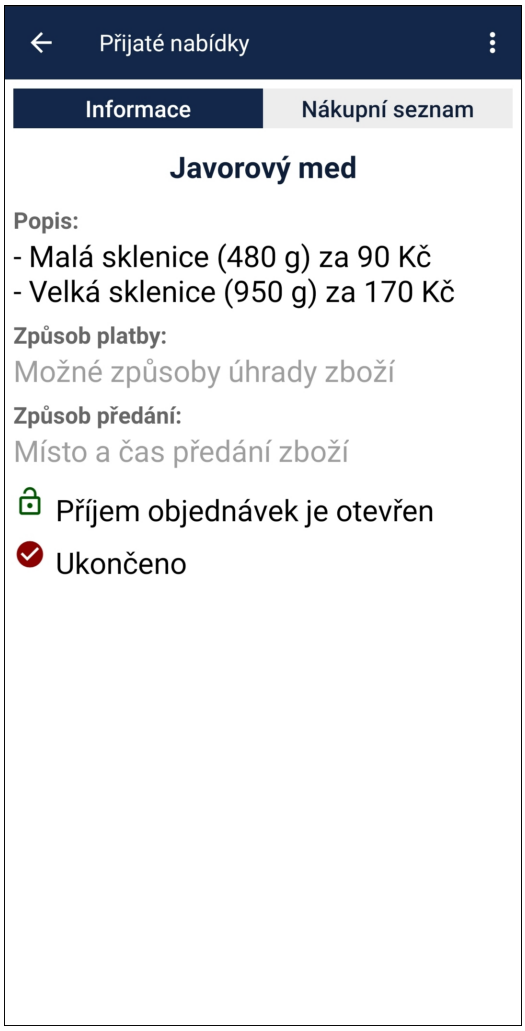
4.4.3 Nabídky členů komunity a uživatelem akceptované nabídky

Pakliže si uživatel vybere ze svého seznamu kanálů, jenž byl k vidění na obrázku 4.6a, kanál *Nabídky členů komunity* či *Přijaté nabídky*, zobrazí se mu veškeré nabídky na společné nákupy i produktů ke koupi od jeho kamarádů a známých (viz obrázek 4.12). V každé položce, již si uživatel bude moci kliknutím zobrazit (viz obrázky 4.13a a 4.14a), vykresleného soupisu pak uvidí mj. jméno jejího vlastníka a za jejím názvem bude v závorce uvedeno jméno kanálu, do něhož byla jejím tvůrcem vložena. Bude-li pak uživatel chtít nějakou nabídku z kanálu *Nabídky členů komunity* přijmout, bude muset být jeho zařízení připojeno k internetu, aby bylo možné zjistit, zda je stále otevřen příjem objednávek nabízených produktů. Po jejím akceptování pak dojde k jejímu přesunutí do kanálu *Přijaté nabídky*, nicméně většina respondentů se v dotazníku (viz příloha A) shodla, že by obrazovka měla uživateli stále zobrazovat zbylé nabídky z kanálu *Nabídky členů komunity*.

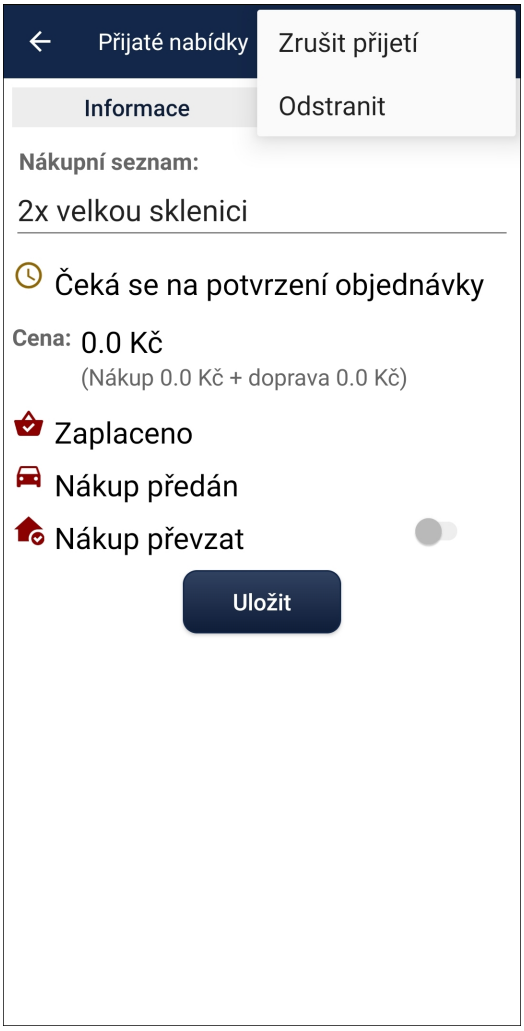
Zobrazí-li si uživatel jednu z jím akceptovaných nabídek, bude do ní moci vyplnit svůj nákupní seznam a rovněž v ní bude moci potvrdit převzetí jím objednaného nákupu u jeho kamaráda či známého (viz obrázky 4.13b a 4.14b). Aby však bylo možné zajistit, že se organizátor sdíleného nákupu nejpozději v době ukončení nabídky dozví, co si chtějí jeho kamarádi a známí zakoupit, bude muset být zařízení příjemce nabídky připojeno k internetu, pakliže bude aktualizovat svoji objednávku.



Obrázek 4.12: Na obrázku je zobrazena obrazovka se seznam všech uživatelem přijatých nabídek, v němž budou uvedena jména jejich vlastníků, dále jejich názvy a v závorce budou napsaná jména kanálů, do kterých byly vloženy svými tvůrci. Uživatel v seznamu rovněž uvidí, do kdy má u jím akceptovaných nabídek na sdílený nákup vyplnit jednotlivé nákupní seznamy, a též mu bude prostřednictvím ikon sděleno, zda je stále otevřen příjem objednávek nabízených produktů, zda již byly organizátorům skupinových nákupů doručeny hromadné objednávky a jestli již jím akceptované nabídky byly ukončeny – tj. jestli zaplatil své jednotlivé objednávky a jestli si je převzal.

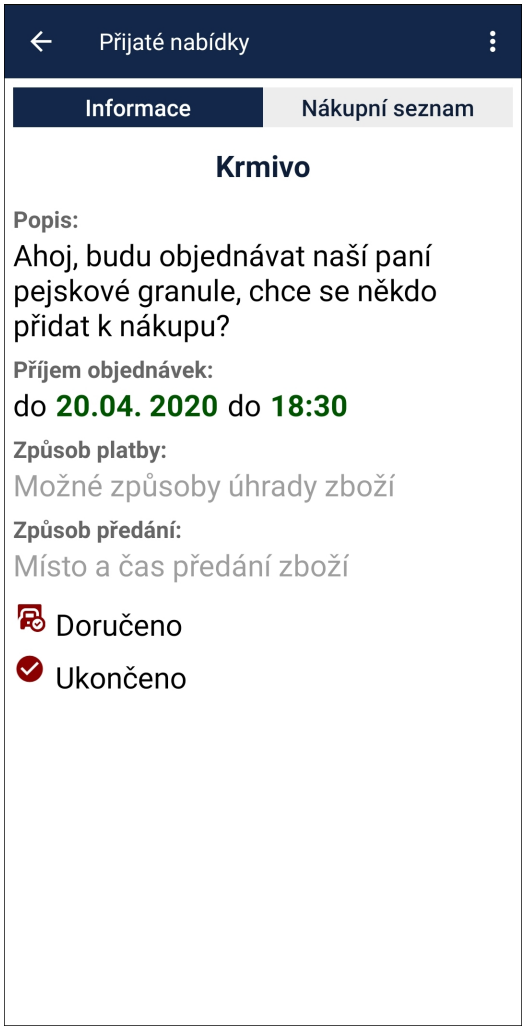


(a) Obrazovka s informacemi o přijaté nabídce produktů ke koupi.

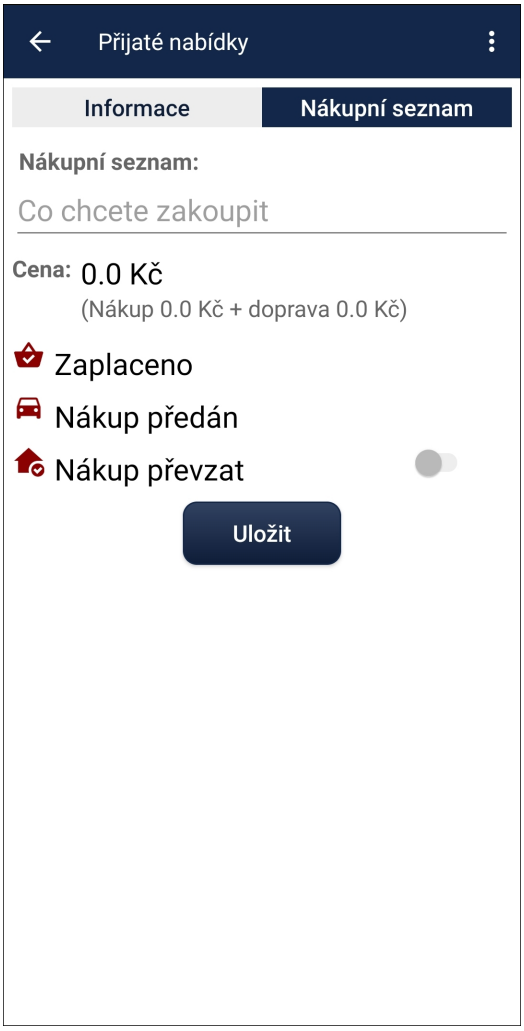


(b) Obrazovka s nákupním seznamem daného uživatele aplikace.

Obrázek 4.13: Na levém obrázku je vyobrazena obrazovka obsahující informace o uživatelem akceptované nabídce, v níž mu jeho kamarád či známí nabízí produkty ke koupi. Na obrázku vpravo je pak znázorněna obrazovka s jeho objednávkou, u které uživatel, v případě přijaté nabídky produktů, uvidí, jestli byla prodejcem potvrzena, a rovněž se dozví, kolik mu má zaplatit, zda jeho platbu potvrdil a jestli si od něj nákup převzal.



(a) Obrazovka s informacemi o přijaté nabídce na společný nákup.



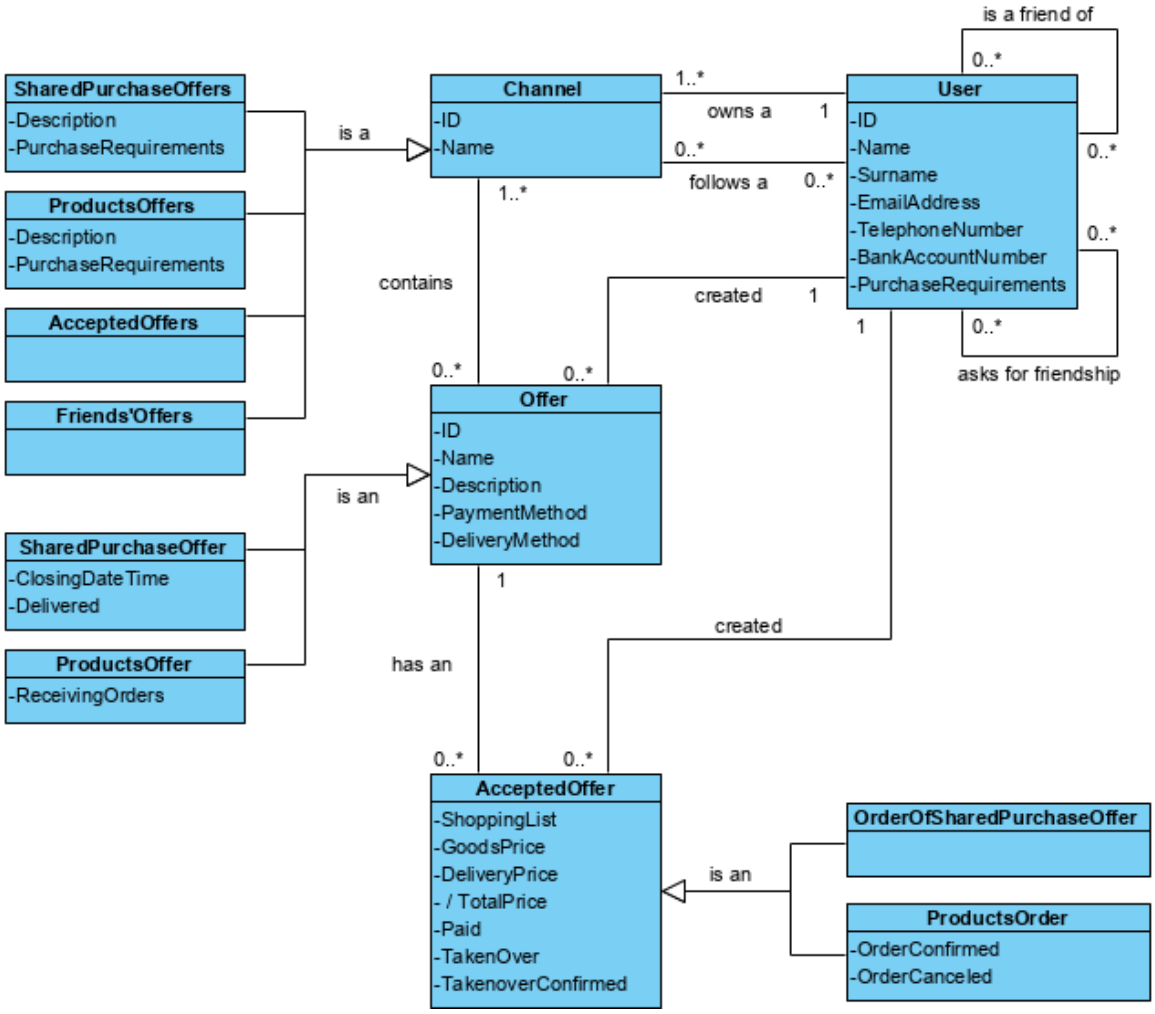
(b) Obrazovka s nákupním seznamem daného uživatele aplikace.

Obrázek 4.14: Levý obrázek demonstruje obrazovku s informacemi o uživatelem přijaté nabídce na společný nákup u prodejce *nase-zoo.cz*. Na obrázku vpravo je pak obrazovka, pomocí níž bude moci příjemce nabídky organizátorovi společného nákupu sdělit, co chce u dané společnosti zakoupit, a ze které se opět dozví, kolik mu má za jeho objednávku zaplatit, zda mu již platbu předal či zaslal a jestli si od něj nákup převzal.

4.5 Model domény aplikace

Na obrázku 4.15 je zobrazen konceptuální model domény aplikace, jenž vychází z výše popsaného návrhu, fyzické schéma lokální databáze pak bude k vidění v kapitole 5.1, kde i bude podrobněji popsáno. Na obrázku je možné vidět, že se v aplikaci budou nacházet čtyři typy kanálů, přičemž každý bude vlastnit právě jeden uživatel. Ten bude mít ve své komunitě několik členů, již si bude moci dynamicky rozšiřovat o nové kamarády pomocí zasílání žádostí a rovněž z ní bude moci mazat stávající členy. I on však bude členem jednotlivých komunit jeho přátel a taktéž jej budou moci jeho známí žádat o přátelství. Členové komunity uživatele pak budou moci sledovat jakékoliv jím vytvořené kanály, stejně

tak i on bude moci sledovat jejich kanály. Ty budou obsahovat nabídky na společné nákupy či produktů ke koupi, přičemž každá bude vytvořena právě jedním uživatelem. Jednotlivé nabídky pak budou moci být akceptované několika členy jeho komunity a rovněž jejich příjemci budou moci uvést, co si chtějí zakoupit.



Obrázek 4.15: Obrázek ilustruje model domény aplikace, jež bude obsahovat čtyři typy kanálů, v nichž budou obsaženy nabídky na společné nákupy či produktů ke koupi. Ty budou moci akceptovat členové komunity jejich vlastníka a taktéž budou moci uvést, co si chtějí zakoupit.

Kapitola 5

Implementace mobilní aplikace

Jelikož jsou do jednotlivých nových verzí operačního systému Android postupně přidávány nové funkce, společnost Google zavedla tzv. úrovně API. Každá verze této platformy má pak přiřazenou právě jednu úroveň API, čímž specifikuje, jakou sadu funkcí podporuje. Aby však bylo možné zaručit, že je daná aplikace kompatibilní s nějakým zařízením, na němž je nainstalovaná určitá verze operačního systému Android, každá by měla mít nastavenou hodnotu minimální úrovně API. Ta je reprezentovaná celým číslem a definuje, jaké funkce API jsou pro její správný běh nezbytné [31]. Pro zajištění co největší dostupnosti aplikace byla tedy nastavena minimální úroveň API na hodnotu 17, čímž by měla být kompatibilní s 99,2 % zařízení.

Aplikace byla vytvořena podle návrhu, jenž byl popsán v kapitole 4 a jenž odráží, co uživatelům nabízí. Byla implementována v multiplatformním objektově orientovaném programovacím jazyce Java podle doporučené architektury, jež vychází z návrhového vzoru *Model-View-ViewModel* a jejíž jednotlivé části budou popsány v následujících podkapitolách. Její základní třídou je `SharedShoppingApp`, která uchovává globální stav aplikace. Obsahuje tedy metody pro získání instancí tříd `AppLocalDatabase` a `AppRepository`, tedy třídy zajišťující přístup k lokálnímu perzistentnímu úložišti dat aplikace a třídy udržující objekty tříd vykonávajících databázové operace nad jednotlivými instancemi tříd datového modelu, tedy nad entitami. Tato třída, tedy `SharedShoppingApp`, je potomkem třídy `Application`[1], což je hlavní třída každé aplikace, jež obsahuje všechny její komponenty a jež je první třídou, která je instanciována, jakmile je vytvořen proces aplikace.

5.1 Model

Podle návrhu mobilní aplikace bylo nejdříve vytvořeno fyzické schéma lokální databáze, z něhož vychází i uspořádání dat ve vzdálené databázi. To, jak je možné vidět na obrázku 5.1, se skládá ze tří entit a pěti vazebních entit. Jejich význam je popsaný v následujícím výčtu:

- *User* reprezentuje konkrétního uživatele aplikace, který je identifikován celočíselnou hodnotou uloženou v atributu `id`, nikoliv tedy e-mailovou adresou, neboť tu by mohl v průběhu času změnit. Jelikož jsem však původně uvažovala o tom, že by v lokální databázi mohlo dojít ke kolizi dvou hodnot primárního klíče entity (viz dále), každý objekt je v lokální databázi identifikován jinou celočíselnou hodnotou než ve vzdálené.

[1] https://developer.android.com/reference/android/app/Application

Přestože se však později ukázalo, že je při přihlašování jednotlivých uživatelů k účtu Google vhodné, aby jejich zařízení bylo připojeno k internetu, entita má ještě jeden celočíselný atribut – `remoteDbId`. V něm je tedy uložena hodnota primárního klíče, pod nímž je daný objekt uložen ve vzdálené databázi. Této skutečnosti však bylo později využito za účelem jednoznačného určení, jaký uživatel ze všech v lokální databázi uložených je na daném zařízení do aplikace zaregistrován, a to tak, že každému uživateli je při registraci do aplikace přiřazen identifikátor hodnoty jedna.

- *FriendRequest* představuje žádost o přátelství zaslanou uživatelem, jehož hodnota primárního klíče lokální databáze je uložena v atributu `fromUserId`, nějakému jeho kamarádovi.
- *Friends* znázorňuje, kteří dva uživatelé spolu kooperují, tj. jaká osoba je členem komunity jiné a naopak.
- *Channel* představuje kanál uživatele aplikace či člena jeho komunity, jenž je opět identifikovaný celočíselnou hodnotou uloženou v atributu `id`. Každý kanál však může být vytvořen i v době, kdy jeho tvůrce není připojen k internetu. Aby tedy v lokální databázi nemohlo dojít ke kolizi dvou hodnot primárního klíče – k ní by mohlo dojít např. tehdy, kdyby měl uživatelem nově vytvořený kanál přiřazen od vzdálené databáze identifikátor, pod nímž by již byl v lokální databázi uložený jiný nově vytvořený a doposud do vzdálené databáze nepropagovaný kanál –, opět se do ní ukládají jako u entity *User* dvě celočíselné hodnoty, přičemž jednou z nich je daný objekt identifikován v lokální, druhou pak ve vzdálené databázi.
- *Offer* reprezentuje nabídku na společný nákup či produktů ke koupi, která byla vytvořena jedním z uživatelů. I ona má ze stejného důvodu jako entita *Channel* dva celočíselné atributy, jež zajišťují, že jsou do lokální databáze ukládány i hodnoty primárního klíče, jimiž jsou jednotlivé nabídky identifikované ve vzdálené databázi.
- *ChannelOffer* zastupuje vztah kanál – nabídka, tj. v jakém kanálu se daná nabídka vyskytuje.
- *FollowedChannel* znázorňuje, kým je daný kanál sledován.
- *AcceptedOffer* představuje přijatou nabídku, tj. jedná se objednávku uživatele aplikace či člena jeho komunity.

Dle schématu databáze byly následně s využitím knihovny *Room*, jež byla popsaná v kapitole 3.5.1, implementované jednotlivé entity, tedy třídy anotované výrazem `@Entity`, které reprezentují tabulky lokální perzistentní databáze *SQLite*. K nim pak byla vytvořena rozhraní anotovaná výrazem `@Dao`. Ta obsahují deklarace metod (viz výpis 5.1), jejichž implementaci zajišťuje knihovna *Room* a jež umožňují přístup k požadovaným datům jednotlivých tabulek databáze. Jako hlavním přístupovým bodem k lokální databázi *SQLite* je pak abstraktní třída `AppLocalDatabase`, neboť obsahuje abstraktní metody, jejichž návratovým typem jsou právě ona rozhraní zajišťující přístup k jejím relačním datům. Abstraktní třída byla implementovaná podle návrhového vzoru *Singleton*, přičemž instanci jejího potomka vytváří knihovna *Room*, a kromě seznamu entit uchovává i soupis konvertorů, které zajišťují převod některých objektů (např. třídy `Date`) na jim odpovídající hodnotu primitivního datového typu, jež bude moci být do lokální databáze uložena, a zpětnou konverzi hodnoty získané z databáze na jí odpovídající objekt.

```java
/**
 * Rozhraní zajišťující přístup k~datům tabulky user lokální perzistentní
 * databáze SQLite.
 */
@Dao
public interface UserDao {
    @Insert
    Long insertUser(UserEntity userEntity);

    @Update
    void updateUser(UserEntity userEntity);

    @Delete
    void deleteUser(UserEntity userEntity);

    @Query("SELECT * FROM user " +
           "WHERE id = :id")
    UserEntity getUserById(final Long id);
}
```

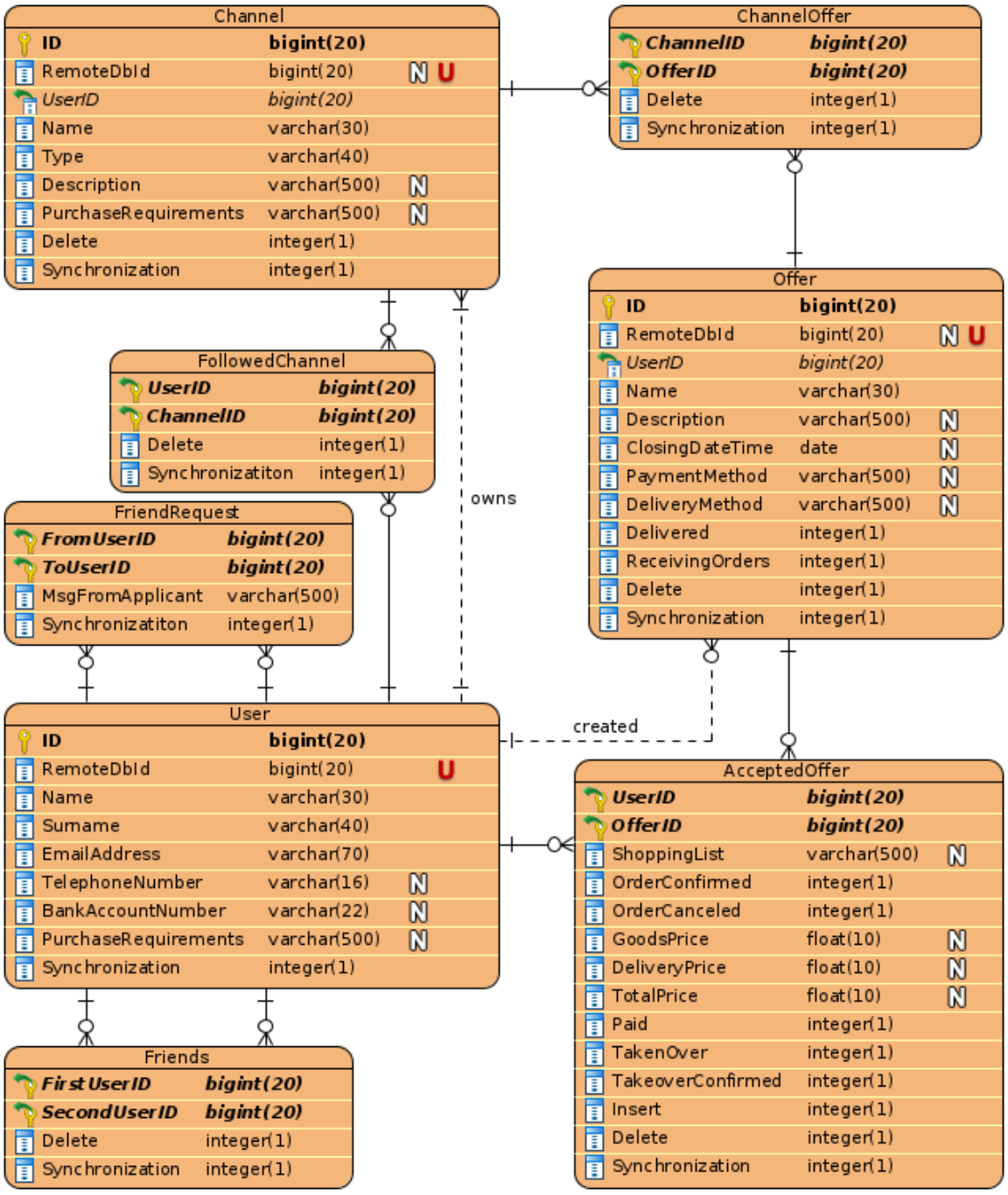
Výpis 5.1: Ukázka rozhraní, které zajišťuje prostřednictvím deklarovaných metod přístup k jednotlivým datům databázové tabulky, jež obsahuje informace o uživatelích aplikace. Ty jsou anotovány výrazy, jež určují, jaká databázová operace má být nad danou tabulkou vykonána.

Některé entity jsou potomky jim odpovídajících abstraktních tříd datového modelu, které obsahují pomocné metody implementující logiku aplikace. Zajišťují tedy např. kontrolu platnosti vstupních dat, ověření, zda již byl příjem objednávek uzavřen či zda byla nabídka ukončena, atp. Aby však bylo možné jednoduše seřadit množinu kanálů daného uživatele podle abecedy, jim odpovídající třída implementuje metodu `compareTo()` rozhraní `Comparable`[2], jež vrací celé číslo indikující, který ze dvou objektů předchází tomu druhému, nebo zda jsou porovnávané instance stejné. Pro zajištění předávání objektů datového modelu mezi jednotlivými komponentami aplikace některé abstraktní třídy rovněž implementují rozhraní `Parcelable`[3]. V implementacích jejich metod jsou pak uvedeny explicitní definice, jak mají být jednotlivé objekty serializovány a jak deserializovány.

Jelikož je zapotřebí kontrolovat, zda je zařízení připojeno k internetu, či nikoliv, byla vytvořena třída `NetworkStateChangeReceiver`, jež je implementována podle návrhového vzoru *Observer*. Ta je potomkem abstraktní třídy `BroadcastReceiver`[4], tedy přijímače, čímž může od systému přijímat upozornění, že došlo k nějaké události. Jakmile tedy třída `NetworkStateChangeReceiver` obdrží asynchronní zprávu, tj. záměr, jímž je na vzniklou událost upozorněna, zareaguje na ni v jí předefinované metodě `onReceive()`. Nejprve tedy zjistí, zda je zařízení připojeno k síti, či nikoliv, a pak všechny své posluchače upozorní voláním jedné z metod rozhraní `NetworkStateChangeObserver`, které implementují, že došlo ke změnu stavu. Pokud však nějaké třídy nemusejí být průběžně o jednotlivých změnách

[2] https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html
[3] https://developer.android.com/reference/android/os/Parcelable
[4] https://developer.android.com/reference/android/content/BroadcastReceiver

informované, třída poskytuje i statickou metodu pro získání informace o aktuálním stavu internetového připojení.



Obrázek 5.1: Obrázek znázorňuje fyzické schéma lokální databáze aplikace.

5.2 Přístup ke vzdálenému zdroji dat

Přístup ke vzdálenému serveru zajišťuje třída `AppRemoteDatabase`, jež vytváří s použitím knihovny *Retrofit*, která byla popsána v kapitole 3.5.2, HTTP klienta. Ten pro výměnu dat se vzdáleným serverem používá knihovnu *Gson*, jež zajišťuje konverzi odchozích dat na řetězec ve formátu JSON a zpětný převod příchozích dat na jim odpovídající objekty. Pro zabezpečení datového přenosu mezi oběma stranami pak HTTP klient používá protokol TLS, čímž tak bylo zapotřebí nakonfigurovat certifikát serveru.

Jednotlivé požadavky protokolu HTTP jsou reprezentovány deklaracemi metod v rozhraní `Webservice`, jejichž implementaci generuje knihovna *Retrofit*. U nich, jak je možné vidět na výpisu 5.2, jsou uvedené názvy metod protokolu HTTP a relativní URL, která specifikují přesné umístění zdroje na vzdáleném serveru. Těmi jsou skripty ve formátu PHP, jež kontrolují příchozí data, zajišťují vykonávání databázových dotazů, vytváří odpovědi a ty zpravidla ve formátu JSON zasílají zpět klientovi. Připojení k databázi pak vytváří skript *connection.php*.

```
/**
 * Rozhraní deklarující metody pro přístup ke vzdálenému serveru.
 */
public interface Webservice {
   @FormUrlEncoded
   @POST("getFriendsChannels.php")
   Call<List<ChannelEntity>> getFriendsChannels(
           @Field("userId") Long userRemoteDbId);
}
```

Výpis 5.2: Ukázka rozhraní obsahujícího deklaraci metody, jež představuje požadavek protokolu HTTP. Ta je anotovaná výrazem, jenž určuje, jaká metoda protokolu HTTP má být použita. Před deklarací je rovněž uvedeno relativní URL, které specifikuje přesné umístění skriptu ve formátu PHP na vzdáleném serveru. Ten je určený pro získání všech kanálů jednotlivých členů komunity daného uživatele i uživatelů, kteří jej žádají o přátelství.

5.3 Repository

Repository je modul provádějící operace nad daty. Jeho hlavní třídou je třída `AppRepository`, která zajišťuje synchronizaci lokální databáze se vzdálenou a rovněž udržuje objekty tříd vykonávajících databázové operace nad jednotlivými instancemi tříd datového modelu, tedy nad entitami. Ty jsou stejně jako třída `AppRepository` implementovány podle návrhového vzoru *Singleton* a poskytují metody pro synchronizaci jednotlivých tabulek se vzdálenou databází i pro vytváření synchronních a asynchronních požadavků protokolu HTTP, jež zajišťují vykonání databázových operací na vzdáleném serveru. Aby však uživatel nemusel kontrolovat, jestli nějaký člen jeho komunity např. nevytvořil novou či neaktualizoval svou již existují nabídku, nebo jestli needitoval svoji či jeho objednávku atp., některé z těchto metod vytváří notifikace, které zobrazuje statická metoda třídy `Notification`, a rovněž některé vytváří bublinové dialogy s krátkými informativními textovými zprávy.

Jelikož aplikace podporuje off-line režim, je zapotřebí po připojení zařízení k internetu propagovat do vzdálené databáze nově vytvořené záznamy či provedené změny nad již

existujícími. Aby však bylo možné zjistit, jak byla lokální databáze oproti vzdálené změněna, každá entita má pomocné atributy. Ty indikují, zda u daného objektu došlo k nějaké změně, či nikoliv a případně jaká databázová operace byla nad ním provedena. Počet pomocných atributů se však u jednotlivých entit liší. Např. u entity *Channel* jsou zapotřebí dva – kupř. `delete` a `synchronization`, kde atribut `delete` indikuje, zda má být daný objekt odstraněn, či nikoliv, kdežto atribut `synchronization` umožňuje v lokální databázi snadno vyhledat všechny kanály, u nichž došlo k nějaké změně. Jestli pak má být daný objekt do vzdálené databáze uložen či jestli v ní má být aktualizován jemu odpovídající záznam, lze zjistit z toho, zda má již přiřazenou hodnotu primárního klíče vzdálené databáze, či nikoliv. Přestože však nad každým objektem může být v době, kdy zařízení není připojeno k internetu, provedeno hned několik různých databázových operací, není zapotřebí je ve vzdálené databázi vykonat všechny. Jaké změny provedené nad jednotlivými kanály uloženými v lokální databázi, které mají nastavený atribut `synchronization`, musí být promítnuty do vzdálené databáze, shrnuje následující tabulka 5.1:

Insert	**Update**	**Delete**	**Výsledná databázová operace**
1	1	1	Odstranění záznamu jen z lokální databáze.
1	1	0	Uložení záznamu do vzdálené databáze.
1	0	1	Odstranění záznamu jen z lokální databáze.
1	0	0	Uložení záznamu do vzdálené databáze.
0	1	1	Odstranění záznamu ze vzdálené i z lokální databáze.
0	1	0	Aktualizace záznamu ve vzdálené databázi.
0	0	1	Odstranění záznamu ze vzdálené i z lokální databáze.
0	0	0	Žádná operace.

Tabulka 5.1: Tabulka ukazuje, jaká z databázových operací *insert*, *update* či *delete*, které byly v různých kombinacích provedené nad nějakým kanálem uloženým v lokální databázi v době, kdy zařízení nebylo připojeno k síti, má být propagována do vzdálené databáze. Lze si všimnout, že většina výsledných operací odvozených z jednotlivých kombinací databázových operací *insert*, *update* a *delete* není ovlivněna hodnotou sloupce *Update*, proto tedy není zapotřebí do objektů ukládat informaci, zda došlo k aktualizaci nějakého kanálu, či nikoliv. Přestože by však byla u nějakého objektu nastavena hodnota atributu `synchronization`, ale nebyly by u něj provedeny žádné změny, aktualizací jemu odpovídajícího záznamu ve vzdálené databázi by nedošlo k žádné chybě, neboť tu může provést pouze jeho vlastník.

Protože každý uživatel zpravidla spravuje pouze jím vytvořené objekty, je pak snadné zajistit, aby nedošlo k inkonzistenci dat. Jedinou tabulkou, jejíž data mohou měnit dvě osoby současně, jsou jednotlivé objednávky od konkrétních uživatelů, již přijali nabídku člena své komunity na společný nákup či produktů ke koupi. Výhodou však je, že každé její pole může měnit pouze jeden uživatel, a to na základě toho, v jaké roli se právě nachází – tedy zda je objednávajícím či zda je organizátorem, resp. prodejcem. Jakmile tedy dojde k aktualizaci nějaké objednávky, do vzdálené databáze jsou propagovány jen ty hodnoty atributů, které daná osoba v patřičné roli může editovat. Pokud pak organizátor či prodejce

ze svého zařízení odstraní jím zveřejněnou a již ukončenou nabídku, bude rovněž vymazána ze vzdálené databáze včetně objednávek, jež jsou na ni vázané. K jejímu smazání však nedojde u uživatelů, kteří ji přijali, neboť ta bude z jejich lokální databáze odstraněna až tehdy, jakmile ze svého zařízení smažou jí odpovídající uzavřenou objednávku.

Nejen, že je zapotřebí do vzdálené databáze propagovat změny, jež byly provedeny v době, kdy zařízení nebylo připojeno k internetu, rovněž je nutné z ní pravidelně získávat nová či aktualizovaná data. Třída `AppRepository` je implementována jako pozorovatel třídy `NetworkStateChangeReceiver`, který čeká na upozornění, že dané zařízení bylo připojeno k internetu. Jakmile toto upozornění obdrží, spustí metodu, jež zajistí voláním jednotlivých asynchronních požadavků protokolu HTTP synchronizaci lokální databáze se vzdálenou. Ta je pak opakovaně po uplynutí určité doby spouštěna, dokud objekt třídy `AppRepository` nedostane upozornění, že zařízení bylo odpojeno od sítě.

5.4 ViewModel

Vrstva *ViewModel* obsahuje logiku pro získávání objektů tříd datového modelu. Jednotlivé její třídy dědí ze třídy `AndroidViewModel`[5], která umožňuje jejím potomkům uchovávat kontext aplikace, pomocí něhož lze pak získat objekt základní třídy aplikace, tedy `SharedShoppingApp`. Aby však instancím těchto tříd bylo možné přes konstruktor předat i jiné parametry než pouze objekt třídy `Application`, bylo zapotřebí pro každou třídu vytvořit potomka třídy `ViewModelProvider.NewInstanceFactory`[6]. Jedná se tedy o jednoduchou továrnu, jež předefinovává metodu `create()`, která vrací vytvořený objekt požadované třídy vrstvy *ViewModel*. Samotná instance této třídy je pak vytvořena v patřičné aktivitě či fragmentu třídou `ViewModelProvider`[7], jíž musí být předána reference na odpovídající továrnu (viz výpis 5.3).

```
FriendRequestsViewModel.Factory factory =
   new FriendRequestsViewModel.Factory(getApplication(), myself);
FriendRequestsViewModel friendRequestsViewModel =
   new ViewModelProvider(this, factory).get(FriendRequestsViewModel.class);
```

Výpis 5.3: Ukázka vytvoření instance třídy vrstvy *ViewModel*. Aby ji však bylo možné přes konstruktor předat i jiné parametry než pouze objekt třídy `Application`, musí ji vytvořit třída `ViewModelProvider`. Té je přes konstruktor předána reference na jednoduchou továrnu, která obsahuje metodu `create`, jež vrací požadovaný objekt třídy vrstvy *ViewModel*.

Z tříd vrstvy *ViewModel*, které uchovávají kontext aplikace, lze pak přes základní třídu `SharedShoppingApp` přistupovat k lokální databázi zařízení a rovněž k objektu třídy `AppRepository`, čímž tak mohou být třídám vrstvy *View* poskytována potřebná data a taktéž nad nimi mohou být vykonávané potřebné operace. Některá data jsou pak obalená třídou `LiveData`, a tak mohou být jednotlivými aktivitami či fragmenty voláním metody `observe()` pozorována. Jak je možné vidět na výpisu 5.4, volané metodě je zapotřebí předat referenci na objekt, jenž chce jednotlivé změny dat monitorovat, a na metodu, která se má v případě, že došlo k aktualizaci pozorovaných dat, provést.

[5] https://developer.android.com/reference/android/arch/lifecycle/AndroidViewModel
[6] https://developer.android.com/reference/android/arch/lifecycle/ViewModelProvider.NewInstanceFactory
[7] https://developer.android.com/reference/android/arch/lifecycle/ViewModelProvider

```
final Observer<List<UserEntity>> showFriendRequests =
   new Observer<List<UserEntity>>() {
      @Override
      public void onChanged(@Nullable final List<UserEntity> requests) {
         if (requests != null) {
            // Aktualizace obrazovky
         }
      }
   };

if (friendRequestsViewModel.getFriendRequests() != null) {
   friendRequestsViewModel.getFriendRequests()
      .observe(this, showFriendRequests);
}
```

Výpis 5.4: Ukázka monitorování dat obalených třídou `LiveData`, jež daná aktivita získává od třídy z vrstvy *ViewModel*. Aby však mohla na jednotlivé jejich změny reagovat, musí zavolat metodu `observe` a předat jí referenci na metodu, která zajistí překreslení požadované části obrazovky.

5.5 Aktivity a fragmenty

Hlavní aktivitou aplikace je třída `MainActivity`, jež nejdříve zjistí, jestli již byl na daném zařízení do aplikace zaregistrován nějaký uživatel, či nikoliv. Pakliže ne, zobrazí přihlašovací obrazovku k účtu Google, která je nakonfigurována tak, aby byly o uživateli ze vzdáleného zdroje dat získány pouze základní údaje včetně e-mailové adresy. Ty pak před samotnou registrací zpracuje třída `RegistrationActivity`, tj. zkontroluje, jestli jsou všechny povinné údaje o uživateli k dispozici či jestli musí uvést své jméno a příjmení. Po jeho registraci, kdy se o něm získané údaje uloží do lokálního úložiště a synchronním požadavkem protokolu HTTP do vzdálené databáze, je opět spuštěna hlavní obrazovka, jež nyní zobrazuje prázdný seznam členů jeho komunity.

Jakým způsobem pak mají být strukturované jednotlivé seznamy uživatelů, kanálů a nabídek daného kanálu zajišťují třídy `UsersListAdapter`, `ChannelsListAdapter` a `OffersListAdapter`. Ty jsou potomky třídy `ArrayAdapter`[8] a předefinovávají její metodu `getView()`, která pro každý objekt kolekce určuje, jaká aktivita má být spuštěna, pakliže uživatel vybere jemu odpovídají položku seznamu, a rovněž definuje, jak má být v daném seznamu všech objektů zobrazen.

Aby bylo možné vytvářet neblokující synchronní požadavky protokolu HTTP, byla vytvořena abstraktní třída `AsyncLoaderForActivity`, která implementuje jednotlivé metody rozhraní `LoaderManager.LoaderCallbacks`[9]. Ty umožňují vykonávat dlouhotrvající operace mimo hlavní vlákno aplikace, jež je určené pro interakci s uživatelem, a rovněž umožňují po jejich dokončení patřičně reagovat na jednotlivé odpovědi získané ze vzdáleného serveru. Abstraktní třídu `AsyncLoaderForActivity` pak dědí aktivity, které pro vytvoření a i řízení

[8] https://developer.android.com/reference/android/widget/ArrayAdapter
[9] https://developer.android.com/reference/android/app/LoaderManager.LoaderCallbacks

objektu třídy `Loader`[10], jenž zajišťuje vykonávání asynchronních operací nad daty, volají její metodu `loaderInitialization()`.

Princip dědičnosti byl použit i u některých dalších tříd – např. u aktivit pro vytvoření nové a pro editaci již existující nabídky, neboť jejich grafické uživatelské rozhraní je zcela shodné. To tedy ve svých metodách vytváří abstraktní třída `NewEditOfferActivity` a rovněž obsahuje pomocné metody, jež jsou pro oba její potomky společné. Jelikož se však u obrazovek liší chování některých jejich grafických komponent uživatelského rozhraní, třída obsahuje i deklarace abstraktních metod, které musí její potomci `NewOfferActivity` a `EditOfferActivity` implementovat.

[10] https://developer.android.com/reference/android/content/Loader

Kapitola 6

Testování mobilní aplikace

Za účelem testování jsem oslovila pouze lidi, kteří běžně komunitní nakupování v libovolné jeho podobě provozují. Obrátila jsem se tedy opět na respondenty, již mi vyplnili dotazník, jenž je uvedený v příloze A. Konkrétně jsem požádala pouze ty, kteří mi na sebe v dotazníku nechali e-mailovou adresu. Jelikož však drtivou většinu těchto lidí osobně neznám a rovněž jelikož se nachází v různých částech republiky, rozhodla jsem se pro testování vytvořit další dotazník, což mi tak znemožnilo sledovat, jak se jednotlivé osoby chovají při používání aplikace. Do dotazníku jsem kromě sady nepovinných otázek (viz příloha C) umístila odkaz na publikovanou verzi na Google Play, již si zájemci mohli nainstalovat do svého mobilního zařízení za účelem testování, a scénář testování s úkoly, podle nichž si mohli projít podstatné funkce aplikace (viz příloha B). Pro ty, kteří nevlastní zařízení s operačním systémem Android, jsem pak vytvořila video aplikace a rovněž jsem do dotazníku vložila odkaz na stručného průvodce aplikací, jenž byl doplněn jednotlivými snímky obrazovek. Z jedenácti oslovených dotazník vyplnilo celkem pět lidí, a to ve věkové kategorii 22 až 40 let, přičemž tři z nich si mobilní aplikaci stáhli do svého zařízení.

Na otázku, zda by měly být do aplikace přidány nějaké funkce, jeden z respondentů uvedl, že by zasílání žádostí o přátelství nemuselo být omezeno pouze na aplikaci. Dle něj by tedy mohla v případě, že nějaká osoba aplikaci nepoužívá, generovat elektronické zprávy na uživatelem uvedené e-mailové adresy. Ty by pak měly obsahovat informaci o tom, že nějaká osoba zve jinou k používání aplikace za účelem provozování společných nákupů, a taktéž by v ní měl být uvedený stručný popis aplikace i odkaz na publikovanou verzi na Google Play.

Tentýž respondent uvedl, že u jednotlivých položek seznamu všech uživatelem přijatých nabídek, jenž byl k vidění na obrázku 4.12, by mohly být zobrazené i ikony indikující, zda již jejich příjemce vyplnil nákupní seznam, jestli již od něj organizátor sdíleného nákupu či prodejce obdržel platbu i jestli potvrdil převzetí jím zakoupeného zboží. Ty by podle dotazovaného měli zajistit to, že si uživatel nebude muset potřebovat pokaždé zobrazit podrobnosti o nabídce, neboť by mu zobrazené ikony měli připomenout, zda již výše uvedené činnosti učinil, či nikoliv.

Jiný respondent uvedl, že by aplikace mohla nabízet funkci, která by zajistila sledování všech kanálů konkrétního člena komunity daného uživatele a rovněž která by uživateli umožnila zrušit sledování všech kanálů vytvořených nějakou osobou. Další respondent pak do dotazníku napsal, že by do aplikace přidal vyhledávací pole pro nalezení jednotlivých členů komunity uživatele a i jím vytvořených kanálů. Všechny z výše uvedených připomínek bych ráda do aplikace v budoucnu zapracovala.

Přestože tedy žádný z výše zmíněných podnětů od jednotlivých dotazovaných nebyl z časových důvodů zrealizován, respondenti se shodli na tom, že je aplikace v současném stavu použitelná, nicméně jsem ji na Google Play nezveřejnila, neboť se po přijetí žádosti o přátelství může stát, že se nový člen komunity uživatele nezobrazí v seznamu mezi ostatními. Rovněž dotazovaní uvedli, že uživatelům umožňuje jednoduše svým známým nabízet společné nákupy, eventuálně jejich produkty ke koupi a taktéž že jim poskytuje na jednom místě přehled nejen o tom, co si chtějí jednotliví zájemci zakoupit.

	Karta	**Počet označení**
Pozitivní	Líbivé	4
	Přátelské	4
	Nápomocné	3
	Pohodlné	2
	Šetří čas	2
	Důvěryhodné	2
	Intuitivní	3
	Lákavé	0
Neutrální	Klidné	3
	Jednoduché	3
	Přímočaré	2
	Užitečné	3
	Dobře použitelné	3
	Ucelené	2
	Srozumitelné	3
	Uspořádané	2
Negativní	Matoucí	0
	Stresující	0
	Plýtvání mým časem	0
	Příliš technické	0
	Nezajímavé	0
	Nepředvídatelné	0
	Složité	0
	Nekonzistentní	0

Tabulka 6.1: Tabulka zobrazuje sadu tzv. *emočních karet*, jež jsou buď pozitivně, neutrálně či negativně naladěné. Z nich měli respondenti vybrat ty, které nejvíce vystihují jejich dojmy a pocity z testované aplikace. Celkový počet označení jednotlivých karet dotazovanými je pak možné vidět v pravém sloupci tabulky.

Pro vyhodnocení, jak na jednotlivé respondenty působí uživatelské rozhraní, byla zvolena metoda tzv. *emočních karet*[1] (angl. Card Sorting). Ta se používá tak, že tester ze sady

[1] https://www.aitom.cz/get.php?id=1242

karet obsahujících přídavná jména, jež jsou laděná buď pozitivně, negativně či neutrálně, vybere ty, které nejvíce vystihují jeho dojmy a pocity z testované aplikace. Podle vybraných karet je pak možné snadněji analyzovat subjektivní pocity jednotlivých testerů. Ze sady všech karet, jež jsou k dispozici pod uvedeným odkazem v poznámce pod čarou, bylo zvoleno dvacet čtyři z nich, přičemž z každé skupiny bylo vybráno právě osm karet. Kolikrát byly jednotlivé karty respondenty označeny shrnuje tabulka 6.1. Na ní lze vidět, že dotazovaní vybírali karty jen pozitivně a neutrálně naladěné, přičemž uživatelské rozhraní většina označila za intuitivní a srozumitelné. Ti, co vyplnili otázku, zda by byli schopni začít s aplikací pracovat bez použití jakéhokoliv návodu, se pak shodli na tom, že aplikace je snadná na pochopení i používání. Taktéž nikdo nenaznal, že by se aplikace v nějakých situacích zachovala jinak, než by očekával.

Kapitola 7

Závěr

Cílem práce bylo vytvořit mobilní aplikaci pro operační systém Android, která lidem provozujícím komunitní nakupování se svými blízkými či přáteli usnadní jeho organizaci a taktéž která jim poskytne potřebné informace o dílčích nákupech jednotlivých jeho účastníků. Toho bylo dosaženo kategorizací jednotlivých nabídek do kanálů, jež jsou zobrazeny jen těm, kteří by o ně mohli mít zájem, aniž by je o nich musel organizátor explicitně informovat, a také ukládáním informací, jež mu sdělují, co si chtějí participanti sdíleného nákupu zakoupit, zda mu již nákup zaplatili apod.

V první části práce bylo zapotřebí nejdříve zjistit, jak i v jakých formách sdílené nákupy probíhají a rovněž proč někteří lidé tímto způsobem nakupují. Dále bylo nutné nastudovat, jaká je doporučená architektura mobilních aplikací a z jakých komponent se skládají, stejně tak bylo nutné získat informace o tom, jakým způsobem se vytváří a i přistupuje k lokální perzistentní databázi zařízení i jakým způsobem je možné zajistit bezpečnou komunikaci se vzdáleným serverem. Následně byl vytvořen návrh mobilní aplikace, jejímž uživatelům umožňuje s ohledem na zachování jejich soukromí snadno informovat své známé, že se mohou přidat k nákupu u nějakého prodejce, i jim nabízet nějaké své produkty. Podle návrhu pak byla aplikace realizována v programovacím jazyce Java a posléze testována lidmi, kteří běžně komunitní nakupování provozují.

Jelikož v kanálech určených pro prodej např. přebytků ze zahrádky budou uživatelé opakovaně nabízet svým kamarádům i známým převážně stejné produkty, v budoucnu by si tak pro ně aplikace mohla utvářet konečné množiny výrobků včetně jejich cen, které v nich již někdy byly prodejcem nabízeny. Při vytváření nových nabídek by pak každý uživatel mohl později už jenom z množiny jím již někdy prodávaných produktů vybírat pouze ty, jež by v ní chtěl aktuálně členům své komunity nabídnout, a rovněž by i u nich mohl editovat jejich cenu a uvádět, v jakém množství je má k dispozici. Příjemci těchto nabídek by pak nemuseli vypisovat nákupní seznam, neboť by jim mohl být zobrazen soupis nabízených produktů, v němž by si zaškrtli pouze ty, o něž by měli zájem, a taktéž by u nich uvedli, v jakém množství je požadují.

Ne každý člověk má však potřebu za každou cenu někomu něco prodávat. Proto by aplikace rovněž mohla lidem umožňovat uvést hodnotu jimi nabízených produktů např. v nějaké virtuální měně. Pokud by si pak nějaký člověk chtěl od svého známého něco zakoupit, jemuž dříve daroval kupř. jím upečený domácí chléb, nebylo by tedy zcela fér, kdyby mu měl zaplatit plnou cenu. Výsledná cena jím požadovaného produktu by tedy mohla být vypočítána z jeho reálné ceny, z níž by byla odečtena hodnota jím darovaného výrobku.

Literatura

[1] Guide to app architecture. *Android Developers* [online]. Revidováno 27.12. 2019 [cit. 30. prosince 2019]. Dostupné z: https://developer.android.com/. Path: Jetpack; Get started.

[2] Application Fundamentals. *Android Developers* [online]. [cit. 26. prosince 2019]. Dostupné z: https://developer.android.com/. Path: Docs; Guides; App Fundamentals.

[3] Introduction to Activities. *Android Developers* [online]. [cit. 26. prosince 2019]. Dostupné z: https://developer.android.com/. Path: Docs; Guides; Activities.

[4] Understand the Activity Lifecycle. *Android Developers* [online]. Revidováno 27.12. 2019 [cit. 27. prosince 2019]. Dostupné z: https://developer.android.com/. Path: Docs; Guides; Activities; The activity lifecycle.

[5] Fragments. *Android Developers* [online]. Revidováno 27.12. 2019 [cit. 29. prosince 2019]. Dostupné z: https://developer.android.com/. Path: Docs; Guides; Activities; Fragments.

[6] Intents and Intent Filters. *Android Developers* [online]. Revidováno 27.12. 2019 [cit. 31. prosince 2019]. Dostupné z: https://developer.android.com/. Path: Docs; Guides; Intents and Intent Filters.

[7] ViewModel Overview. *Android Developers* [online]. [cit. 31. prosince 2019]. Dostupné z: https://developer.android.com/. Path: Docs; Guides; Architecture Components; ViewModel.

[8] Save data in a local database using Room. *Android Developers* [online]. [cit. 1. ledna 2020]. Dostupné z: https://developer.android.com/. Path: Docs; Guides; App data & files; Save data in a local database; Overview.

[9] LiveData Overview. *Android Developers* [online]. [cit. 3. ledna 2020]. Dostupné z: https://developer.android.com/. Path: Docs; Guides; Architecture Components; LiveData.

[10] Defining data using Room entities. *Android Developers* [online]. Revidováno 27.12. 2019 [cit. 2. ledna 2020]. Dostupné z: https://developer.android.com/. Path: Docs; Guides; App data & files; Save data in a local database; Defining data using entities.

[11] Accessing data using Room DAOs. *Android Developers* [online]. [cit. 2. ledna 2020]. Dostupné z: https://developer.android.com/. Path: Docs; Guides; App data & files; Save data in a local database; Accessing data using DAOs.

[12] Certifikáty. *BADEKO* [online]. [cit. 5. prosince 2019]. Dostupné z: https://www.badeko.cz/. Path: Certifikáty.

[13] Evropská deklarace KPZ. *KPZinfo: KPZ znamená místní jídlo bez kompromisů* [online]. [cit. 1. prosince 2019]. Dostupné z: https://kpzinfo.cz/. Path: Co je KPZ?

[14] Principy KPZ. *KPZinfo: KPZ znamená místní jídlo bez kompromisů* [online]. [cit. 1. prosince 2019]. Dostupné z: https://kpzinfo.cz/. Path: Co je KPZ?

[15] Otázky o KPZ. *KPZinfo: KPZ znamená místní jídlo bez kompromisů* [online]. [cit. 1. prosince 2019]. Dostupné z: https://kpzinfo.cz/. Path: Co je KPZ?

[16] Nákupní skupinu může mít každý. *Scuk.cz* [online]. [cit. 3. prosince 2019]. Dostupné z: https://www.scuk.cz. Path: Pro organizátory.

[17] Alternativa ke klasickému nakupování. *Scuk.cz* [online]. [cit. 3. prosince 2019]. Dostupné z: https://www.scuk.cz. Path: O nás.

[18] S chutí podporujeme lokální výrobce a farmáře. *Scuk.cz* [online]. [cit. 3. prosince 2019]. Dostupné z: https://www.scuk.cz. Path: Pro prodejce.

[19] Gson: A Java serialization/deserialization library to convert Java Objects into JSON and back. *GitHub* [online]. 2008 [cit. 31. března 2020]. Dostupné z: https://github.com/google/gson.

[20] Retrofit: A type-safe HTTP client for Android and Java. *Square Open Source* [online]. 2013 [cit. 31. března 2020]. Dostupné z: https://square.github.io/.

[21] Komunitou podporované zemědělství. *Hnutí DUHA: Friends of the Earth Czech Republic* [online]. 2016 [cit. 1. prosince 2019]. Dostupné z: https://hnutiduha.cz/. Path: Naše práce; Zemědělství; Témata.

[22] The Linux Kernel Organization. *The Linux Kernel Archives* [online], 13. dubna 2017 [cit. 26. prosince 2019]. Dostupné z: https://www.kernel.org/. Path: About.

[23] Android: Retrofit. *VEnCa-X blog* [online], 28. října 2017 [cit. 31. března 2020]. Dostupné z: https://blog.venca-x.cz/. Path: Android.

[24] Global mobile OS market share in sales to end users from 1st quarter 2009 to 2nd quarter 2018. *Statista* [online]. 2019 [cit. 26. prosince 2019]. Dostupné z: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/.

[25] Scuk.cz: O komunitním nakupování s Kamilem Demuthem. *YouTube* [online], 28. května 2019 [cit. 3. prosince 2019]. Kanál uživatelky Margit Slimákové. Dostupné z: https://www.youtube.com/c/MargitTV.

[26] Kamil Demuth – Scuk.cz, 3.místo. *YouTube* [online], 1. března 2019 [cit. 3. prosince 2019]. Kanál Total Česká republika. Dostupné z: https://www.youtube.com/channel/UCRHxOyEPSre6sW8beTukBEQ.

[27] Callaham, J. The history of Android OS: its name, origin and more. *Android Authority* [online], 18. srpna 2019 [cit. 26. prosince 2019]. Dostupné z: https://www.androidauthority.com/.

[28] Lacko, L. *Mistrovství Android: Kompletní průvodce vývojáře*. 1. vyd. Brno: Computer Press, červenec 2017. 648 s. ISBN 978-80-251-4875-4.

[29] Mištová, J. MVVM. *Ackee* [online], 20. prosince 2018 [cit. 31. prosince 2019]. Dostupné z: https://www.ackee.cz/. Path: Blog.

[30] Schramm, D. *Mobilní aplikace pro inventarizaci majetku MU* [online]. Brno, 2019. [cit. 28. března 2020]. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Machač, Z. Dostupné z: https://is.muni.cz/th/v2psm/.

[31] Ujbányai, M. Úrovně Android API. In: *Programujeme pro Android*. 1. vyd. Redaktorka Zuzana Malečková. Praha: Grada Publishing, a.s., 2012, kap. 3, s. 47 – 49. ISBN 978-80-247-3995-3.

[32] Vávrů, J. a Ujbányai, M. Životní cyklus aktivity. In: *Programujeme pro Android: Druhé, rozšířené vydání*. 1. vyd. Redaktor Štěpán Böhm. Praha: Grada Publishing, a.s., 2013, kap. 3, s. 42 – 43. ISBN 978-80-247-4863-4.

[33] Šimlová, D. *Nakupujeme společně: Objevte 6 výhod komunitního nakupování* [online]. [cit. 1. prosince 2019]. 20 s. Dostupné z: https://www.nakupujemespolecne.cz/ebook-zdarma-6-vyhod.

Příloha A

Dotazník A

Mobilní aplikace pro sdílené nakupování 1/3

Zdravím všechny příznivce skupinových nákupů, má představa o aplikaci spočívá ve vytváření si vlastní jakési komunity tvořené jen z lidí, které já jakožto uživatel dobře znám a kterým chci nabízet společné nákupy v jistých obchodech (u jistých dodavatelů, pěstitelů atp.), případně nějaké věci (kupř. přebytky ze zahrádky, domácí marmelády, zavařeniny, med atp.) k odkoupení či k prostému darování.

V následující části dotazníku se nacházejí otázky zaměřené obecně na komunitu.

1. Jaké informace o Vaší osobě byste byl/a ochoten/ochotna poskytnout členům Vaší komunity, nebo též jaké informace jim musíte za účelem skupinových nákupů poskytnout?

☐ Profilová fotografie

☐ Jméno a příjmení

☐ E-mailová adresa

☐ Telefonní

☐ Adresa

☐ Číslo účtu

☐ Jiné . . .

2. Máte při provozování skupinových nákupů stanovené obecné požadavky, resp. jakási nepsaná pravidla týkající se např. toho, do kdy máte organizátorovi nákupu zaplatit Vámi požadované zboží, jak mu jej máte zaplatit (hotově či přes účet) atp.?

◯ Ano

◯ Ne

3. Vzhledem k tomu, že by profily uživatelů neměly být veřejně viditelné, je nutné každou osobu používající aplikaci jednoznačně identifikovat. Podle jakého identifikátoru byste tedy rád/a např. Vašeho kamaráda v aplikaci vyhledal/a?

☐ Uživatelské jméno

☐ E-mailová adresa

☐ Telefonní číslo

☐ Jiné . . .

4. Jakým způsobem by si měl uživatel přidat do své komunity nového člena?

◯ Uživatel vyplní uživatelské jméno, které jím hledaná osoba v aplikaci používá. Následně jí pošle žádost o „přátelství“, kterou může jak přijmout, tak i odmítnout.

◯ Uživatel vyplní e-mailovou adresu osoby, již si chce přidat do komunity, a následně jí pošle žádost o „přátelství“, kterou může jak přijmout, tak i odmítnout. V případě, že tato osoba nepoužívá aplikaci, jí bude zaslána e-mailová zpráva s pozvánkou k jejímu používaní.

◯ Uživatel vyplní telefonní číslo osoby, již si chce přidat do komunity, a následně jí pošle žádost o „přátelství“, kterou může jak přijmout, tak i odmítnout. V případě, že tato osoba nepoužívá aplikaci, jí bude zaslána SMS zpráva s pozvánkou k jejímu používaní.

◯ Jiné . . .

Mobilní aplikace pro sdílené nakupování 2/3

Vzhledem k tomu, že každý člověk může ostatním nabízet i ty věci ke koupi (jimi mám na mysli např. ovoce, zeleninu, vejce, ořechy, maso, med . . . prostě to, co se nám doma urodí, vyroste atp.), které pro někoho nejsou vzácné, nebo společné nákupy v obchodech (u dodavatelů, pěstitelů atd.), o které někteří nemusí mít zájem, je vhodné tyto nabídky každému na míru filtrovat. Každý uživatel by si tedy tvořil tzv. kanály (obdoba Twitteru), přičemž každý by byl zaměřený na něco do jisté míry konkrétního – řekněme, že bych si např. vytvořila tři kanály, v jednom bych nabízela společný nákup ve farmě XY, ve druhém nákup v internetovém obchodě nase-domaci-zoo.cz (název internetové stránky je smyšlený) prodávající krmivo pro zvířata a v posledním bych nabízela ovoce ze zahrádky. Každý ze členů mé komunity by se pak rozhodl, jaké kanály jej zaujaly a ty by „sledoval“. Když bych pak do svého patřičného kanálu zveřejnila nabídku na skupinový nákup s informací, že budu objednávat naší kočkounce jídlo, tato informace by se automaticky zobrazila všem zvířátkomilům v kanálu *Nabídky členů mé komunity* . . .

Následující část dotazníku je (kromě prvních dvou otázek) zaměřena na skupinové nákupy.

1. Pokud byste poslal/a jinému uživateli žádost o „přátelství“, jaké informace o Vaší osobě by v ní měly být uvedené? Nebo též jaké informace byste chtěl/a o osobě, která Vám žádost zaslala, vidět?

☐ Profilová fotografie

☐ Uživatelské jméno

☐ Jméno a příjmení

☐ E-mailová adresa

☐ Telefonní číslo

☐ Názvy Vámi (resp. danou osobou) vytvořených kanálů

☐ Nějaký (nepovinný) text obsahující např. informace, proč Vy/daná osoba žádáte/žádá o „přátelství“.

☐ Jiné . . .

2. Měla by být v nabídce (ať již na skupinový nákup, nebo věcí k zakoupení) uvedena informace, do kdy mohou potenciální zájemci sdělit organizátorovi, co si chtějí zakoupit? Tj. pokud nabízíte někomu společný nákup, stanovujete si datum, případně i čas, do kdy se Vám mají zájemci ozvat?

◯ Ano, je třeba určit alespoň datum.

◯ Ano, je třeba určit datum i čas.

◯ Ne, máme stanovené pravidlo, že nabídka platí jen určitou dobu od jejího zveřejnění.

◯ Ne, je jedno, kdy se ozvou.

◯ Jiné . . .

3. Měla by být v nabídce na skupinový nákup uvedena informace, zda již organizátor nákupu zboží objednal? Tj. pokud provozujete společný nákup, požadujete, aby Vám organizátor tuto informaci sdělil?

◯ Ano, chtěl/a bych to vědět vždy.

◯ Ano, ale pokud by tam ta informace nebyla uvedena, nevadilo by mi to.

◯ Ne, ale pokud by tam ta informace byla uvedena, nevadilo by mi to.

◯ Ne, je to zbytečné zatěžování organizátora nákupu.

◯ Jiné . . .

4. Pokud byste byl/a v roli organizátora, domlouval/a byste se na způsobu platby s každým zvlášť (individuálně)?

◯ Ano

◯ Ne

◯ Nevím

5. Jakým způsobem se s organizátory domlouváte na způsobu platby za Vaši objednávku?

☐ Máme stanovené pravidlo, že objednávku organizátorovi zaplatím ještě před jejím objednáním.

☐ Máme stanovené pravidlo, že objednávku organizátorovi zaplatím po jejím objednání.

☐ Máme stanovené pravidlo, že objednávku organizátorovi zaplatím až při jejím převzetí.

☐ Když mi organizátor nabídne skupinový nákup, rovnou mi i sdělí, do kdy, jak (hotově či přes účet) a v jaké výši mu mám zaplatit zálohu. Zbytek mu doplatím při převzetí zboží.

☐ Když mi organizátor nabídne skupinový nákup, rovnou mi i sdělí, do kdy a jak (hotově či přes účet) mu mám objednávku zaplatit.

☐ Na zaplacení objednávky se s organizátorem domlouvám až po té, co mi sdělí, že objednávku odeslal. Řekne mi, do kdy, jak (hotově či přes účet) a v jaké výši mu mám zaplatit zálohu. Zbytek mu doplatím při převzetí zboží.

☐ Na zaplacení objednávky se s organizátorem domlouvám až po té, co mi sdělí, že objednávku odeslal. Řekne mi, do kdy a jak (hotově či přes účet) mu mám objednávku zaplatit.

☐ Na zaplacení objednávky se s organizátorem domlouvám až po té, co mi sdělí, že je objednávka doručena. Řekne mi, do kdy, jak (hotově či přes účet) a v jaké výši mu mám zaplatit zálohu. Zbytek mu doplatím při převzetí zboží.

☐ Na zaplacení objednávky se s organizátorem domlouvám až po té, co mi sdělí, že je objednávka doručena. Řekne mi, do kdy a jak (hotově či přes účet) mu mám objednávku zaplatit.

☐ Jiné . . .

6. Pokud byste byl/a v roli organizátora, domlouval/a byste se na způsobu předání zboží s každým zvlášť (individuálně)?

◯ Ano

◯ Ne

◯ Nevím

7. Jakým způsobem se s organizátory domlouváte na způsobu převzetí Vaší objednávky?

☐ Organizátor rozešle e-mail s informací, že byla objednávka doručena. Na způsobu předání se domluvíme telefonicky či SMS zprávou.

☐ Organizátor rozešle e-mail s informací, že byla objednávka doručena, a rovnou do něj uvede, v jaké termíny si můžeme zboží převzít.

☐ Organizátor rozešle e-mail s informací, že byla objednávka doručena, a nabídne v něm, že nám zboží doveze.

☐ Organizátor rozešle SMS zprávy s informací, že byla objednávka doručena. Na předání se domluvíme telefonicky.

☐ Organizátor rozešle SMS zprávy s informací, že byla objednávka doručena, a rovnou do ní uvede, v jaké termíny si můžeme zboží převzít.

☐ Organizátor rozešle SMS zprávy s informací, že byla objednávka doručena, a nabídne v ní, že nám zboží doveze.

☐ S organizátorem se spojíme telefonicky. Sdělí mi, že objednávka dorazila, a rovnou se domluvíme na způsobu předání.

☐ Na předání se nedomlouváme. Zboží mi organizátor doveze do práce.

☐ Na předání se nedomlouváme. Zboží si převezmu až tehdy, když jdu k organizátorovi na návštěvu, nebo když on jde ke mně.

☐ Jiné . . .

8. Řekněme, že se Vám ve Vašem kanálu *Nabídky členů mé komunity* zobrazila nabídka na skupinový nákup v obchodě A, o niž byste měl/a zájem, a tak byste klikl/a na tlačítko *Přijmout nabídku*. Co by se mělo následně stát?

◯ Nabídka by se měla přesunout do kanálu *Přijaté nabídky*. Vy byste ale měli stále zůstat v kanálu *Nabídky členů mé komunity*.

◯ Nabídka by se měla přesunout do kanálu *Přijaté nabídky*, který by se Vám měl následně zobrazit se všemi přijatými nabídkami.

◯ Nabídka by měla po jejím přijetí zůstat ve stejném kanálu, tedy v kanálu *Nabídky členů mé komunity*.

◯ Jiné . . .

9. Když byste si zobrazil/a jednu konkrétní Vámi přijatou nabídku skupinového nákupu, měl by v ní být uveden seznam lidí, kteří tutéž nabídku přijali? Tedy lidí, kteří se též chtějí podílet na onom skupinovém nákupu . . . (Nutné poznamenat, že ne všechny lidi byste nutně znal/a, neboť každý uživatel nabízí skupinové nákupy své komunitě a ta se nemusí shodovat s Vaší.)

◯ Ano

◯ Ne

◯ Nevím

10. Pokud byste byl/a v roli organizátora, chtěl/a byste si u každého zájemce o skupinový nákup, resp. věci, jež nabízíte ke koupi, zaznamenat, zda Vám již nákup zaplatil?

◯ Ano

◯ Ne

11. Pokud byste byl/a v roli organizátora, chtěl/a byste si u každého zájemce o skupinový nákup, resp. věci, jež nabízíte ke koupi, zaznamenat, zda si již nákup převzal?

◯ Ano

◯ Ne

Mobilní aplikace pro sdílené nakupování 3/3

Děkuji Vám, že jste se prokousal/a až sem.

Níže jsou uvedené již nepovinné otázky. Pokud Vás však myšlenka této aplikace zaujala, níže můžete uvést jakékoli podněty, náměty, dojmy atd. týkající se tohoto tématu i dotazníku . . .

Moc Vám děkuji! A přeji Vám pěkný den. :-)

1. Pokud byste do budoucna měl/a zájem zodpovědět několik dalších otázek, nebo se třeba jen podívat na video vytvořeného prototypu aplikace, můžete zde na sebe nechat kontakt – e-mailovou adresu.

2. Do které věkové kategorie patříte?

- ◯ 18 let a méně
- ◯ 19 – 21 let
- ◯ 22 – 30 let
- ◯ 31 – 40 let
- ◯ 41 – 50 let
- ◯ 51 – 60 let
- ◯ 60 a více

3. Vaše podněty, náměty, dojmy . . . ?

Příloha B

Scénář pro testování

1. část

Tester A

Chcete se svým kamarádem (Testerem B) provozovat sdílené nákupy a rovněž mu chcete nabízet Vaše produkty ke koupi. Chcete si jej přidat do své komunity.

1. Zašlete Testerovi B žádost o spolupráci.

Tester B

Váš kamarád (Tester A) chce s Vámi provozovat sdílené nákupy a taktéž Vám chce nabízet jeho produkty ke koupi.

1. Zjistěte, zda Vám Tester A zaslal žádost o přátelství.
2. Přijměte žádost o přátelství, kterou Vám Tester A odeslal.

2. část

Tester A

Chcete svým přátelům pravidelně nabízet k zakoupení Váš domácí včelí med. Doma máte k dispozici několik velkých sklenic Javorového medu. Jednu sklenici chcete prodat za 170 Kč.

1. Vytvořte kanál *Včelí med*, v němž budete svým přátelům med nabízet.
2. Vložte do kanálu novou nabídku, v níž chcete kamarádům nabídnout *Javorový med* za výše uvedenou cenu.
3. Zjistěte, zda někdo Vaši nabídku přijal.

Tester B

Víte, že Váš kamarád (Tester A) má doma včelstvo. Chcete si od něj zakoupit nějaký med.

1. Začněte sledovat kanál Testera A, v němž nabízí svůj domácí med ke koupi.
2. Zjistěte, zda Tester A nabízí nějaký med ke koupi.
3. Přijměte nabídku Testera A, v níž Vám nabízí med.

3. část

Tester A

Víte, že Váš kamarád (Tester B) pravidelně nakupuje mouku u prodejce bio-mlyn.cz. Chcete zjistit, zda u něj bude zanedlouho nakupovat. Chcete se k nákupu přidat.

1. Začněte sledovat kanál Testera B, v němž nabízí společné nákupy u daného prodejce.
2. Zjistěte, zda Tester B bude u téhož prodejce zanedlouho nakupovat.
3. Přijměte nabídku společného nákupu u tohoto prodejce, kterou Tester B vytvořil.

Tester B

Pravidelně nakupujete mouku u prodejce bio-mlyn.cz. Za dva dny budete mouku u téhož prodejce kupovat. Chcete svým přátelům nabídnout, že se mohou k Vašemu nákupu přidat.

1. Vytvořte kanál *bio-mlyn.cz*, v němž budete svým přátelům nabízet společné nákupy u tohoto prodejce.
2. Vložte do kanálu novou nabídku, v níž chcete kamarádům nabídnout společný nákup.
3. Zjistěte, zda někdo Vaši nabídku přijal.

Příloha C

Dotazník B

1. Do které věkové kategorie patříte?
   - ◯ 18 let a méně
   - ◯ 19 – 21 let
   - ◯ 22 – 30 let
   - ◯ 31 – 40 let
   - ◯ 41 – 50 let
   - ◯ 51 – 60 let
   - ◯ 60 a více
2. Jakým způsobem jste se s aplikací seznámil/a?
   - ◯ Nainstaloval/a jsem si ji a vyzkoušel/a jsem ji.
   - ◯ Prošel/prošla jsem si průvodce aplikací.
   - ◯ Zhlédl/a jsem video aplikace.
3. Vyberte slova, která nejvíce postihují Vaše dojmy z aplikace:
   - ☐ Líbivé
   - ☐ Matoucí
   - ☐ Klidné
   - ☐ Přátelské
   - ☐ Stresující
   - ☐ Jednoduché
   - ☐ Nápomocné
   - ☐ Plýtvání mým časem
   - ☐ Pohodlné

- ☐ Příliš technické
- ☐ Přímočaré
- ☐ Šetří čas
- ☐ Nezajímavé
- ☐ Užitečné
- ☐ Důvěryhodné
- ☐ Nepředvídatelné
- ☐ Dobře použitelné
- ☐ Ucelené
- ☐ Složité
- ☐ Srozumitelné
- ☐ Intuitivní
- ☐ Lákavé
- ☐ Nekonzistentní
- ☐ Uspořádané

4. Je pro Vás uživatelské rozhraní aplikace, přehledné, srozumitelné a intuitivní? Případně co by se mělo zlepšit, aby aplikace byla více přehledná, srozumitelná či intuitivní?

5. Byl/a byste schopen/schopna začít pracovat s aplikací bez jakéhokoliv návodu, tedy bez jakéhokoliv průvodce aplikací?

6. Přidal/a byste do aplikace nějaké funkce? Jaké?

7. Co by se mohlo na aplikaci vylepšit?

8. Stalo se Vám při manipulaci s aplikací či při zhlédnutí videa, že se aplikace v nějakých okamžicích zachovala jinak, než byste očekával/a? O jaké situace se jednalo a jaké chování jste od aplikace očekával/a?

9. Odrazuje Vás na aplikaci něco od jejího používání? Kvůli čemu byste ji nepoužíval/a?

10. Myslíte si, že je aplikace v současném stavu použitelná? Pakliže ne, co by se mělo změnit, aby ji mohli uživatelé začít používat?

11. Myslíte si, že by mohla aplikace usnadnit organizátorům společných nákupů jejich organizaci? Jak? Pokud ne, jak by jim mohla práci usnadnit?

12. Sem můžete uvést nějaké Vaše připomínky, podněty, náměty i dojmy . . .