FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Mobilní aplikace na podporu udržitelnosti
Student:	Bc. Hana Fukalová
Vedoucí:	Ing. Tomáš Nováček
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Udržitelnost je téma, které je ve společnosti čím dál aktuálnější. Zero-waste styl života, komunitní úklidy, kompostování či swapy získávají na oblibě, ale stále chybí platforma, která by tyto aktivity cíleně podporovala a sdružovala.

1) Analyzujte populární aktivity na podporu udržitelnosti. Jako jednu z forem analýzy využijte dotazníkové šetření. Komunikujte také s organizacemi, které se problematikou zabývají (např. Trash Hero, Mramor, Kokoza).
2) Navrhněte mobilní aplikaci pro operační systém Android, která bude sdružovat a zobrazovat informace o různých aktivitách či důležitých bodech pro komunitu, která se udržitelnosti věnuje. Věnujte se i možnostem přidávání nových informací přímo od uživatelů (například míst, které jsou vhodná pro komunitní úklid).
3) Aplikaci naimplementujte.
4) Proveďte testování s uživateli.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 6. února 2023 v Praze.

Diplomová práce

MOBILNÍ APLIKACE NA PODPORU UDRŽITELNOSTI

Bc. Hana Fukalová

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Tomáš Nováček
4. května 2023

Odkaz na tuto práci: Fukalová Hana. *Mobilní aplikace na podporu udržitelnosti*. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěla bych poděkovat svému vedoucímu, Ing. Tomáši Nováčkovi, za podporu a cené rady při vypracovávání této diplomové práce. Dále bych chtěla poděkovat své rodině a přátelům za podporu při celém svém studiu. V neposlední řadě bych poté chtěla poděkovat 150 anonymním respondentům kteří vyplnili dotaz k teoretické části této práce, a také pěti testerům, kteří mi pomohli aplikaci v závěrečné fázi otestovat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 4. května 2023 .....

Abstrakt

Tato práce pojednává o procesu softwarového vývoje mobilní aplikace na podporu udržitelnosti. Součástí této práce je analýza domény swapu, v rámci které provádím dotazníkové šetření a na základě získaných poznatků a rešerše aplikací ze stejné domény určuji požadavky na aplikaci a případy užití aplikace. Dále se zabývám návrhem prototypu aplikace, který vychází z poznatků získaných při analýze. Vybírám vhodné technologie a navrhuji architekturu aplikace a databáze a vytvářím návrh uživatelského rozhraní. V další kapitole poté popisuji proces implementace prototypu, jaké problémy jsem v průběhu vývoje řešila a jakým způsobem jsem vyřešila různé implementační záludnosti. Následně v kapitole testování popisuji proces uživatelského testování s pěti uživateli a výčet nedostatků a jejich řešení, které byly v rámci testování nalezeny. Další sekce obsahuje poté návrh možných vylepšení do budoucna. V závěru práce poté shrnuji výsledek celého procesu tvorby projektu a jeho celkový přínos.

Klíčová slova Swap, udržitelnost, mobilní aplikace, Android, Firebase, analýza, návrh, implementace, uživatelské testování

Abstract

This master's thesis deals with the software development process of a mobile application supporting sustainability. The work includes an analysis of the swap domain, where I conduct a questionnaire survey and based on the knowledge gained and research of applications from the same domain, I determine the application requirements and use cases. Furthermore, I focus on the design of the application prototype, which is based on the analysis findings. I select appropriate technologies, propose application and database architecture, and create a user interface design. In the next chapter, I describe the implementation process of the prototype, the problems I encountered during development, and how I solved various implementation difficulties. Subsequently, in the testing chapter, I describe the

process of user testing with five users and list the shortcomings and their solutions that were found during the testing. I then propose possible improvements for the future. Finally, in the conclusion of the work, I summarize the result of the entire project creation process and its overall contribution.

Keywords Swap, sustainability, mobile applications, Android, Firebase, analysis, design, implementation, user testing

Seznam zkratek

FB	Facebook
MVC	Model–View–Controller
MVP	Model–View–Presenter
MVVM	Model–View–View Model
DI	Dependency Injection

Úvod

Zachování přírody a udržitelnější přístup k životu jsou témata, o nichž se v dnešní době stále více diskutuje a věnuje se jim větší pozornost. S narůstajícím počtem ekologických krizí se také stále více běžných lidí začíná angažovat v různých aktivitách, které mohou mít pozitivní dopad na životní prostředí. Ačkoli tyto kroky směřující k trvale udržitelnému způsobu života mohou vypadat malicherné, mohou mít velký vliv na celkové změny. Bez ohledu na to, zda se jedná o třídění domácího odpadu, zero-waste životní styl, kompostování, vegetariánství nebo jiné činnosti, všechny tyto aktivity mohou hrát klíčovou roli při transformaci konzumní společnosti v udržitelnou a ekologičtější komunitu.

V rámci přípravy na tuto práci jsem analyzovala různé aktivity na podporu udržitelnosti a snažila jsem se najít způsob, jakým bych mohla přispět k řešení této problematiky. Kontakovala jsem např. neziskovou organizaci Trash Hero, která se věnuje organizaci komunitních úklidů veřejných prostorů. Procházela jsem také společnosti, které organizují hromadné kompostování v Praze nebo zajišťují podporu handicapovaným dětem, ale všechny tyto neziskové organizace často již mají zaběhnuté webové stránky a informační systémy. Nakonec jsem se zaměřila na swap, který je mi myšlenkou blízký, a rozhodla jsem se vytvořit produkt, který by lidem usnadnil domluvu výměn a shánění předmětů na výměnu. Cílem je přispět k omezení konzumerismu dnešní doby a zároveň vytvořit nástroj, který by mohl lidem přinést užitek a pomoci udržet předměty v oběhu co nejdéle.

Swap

Swap je pojem označující jednu z činností, kterou mohou lidé zakomponovat do svého každodenního života a kterou mohou přispět k šetrnějšímu zacházení s našim životním prostředím. Swap je slovo převzato z anglického jazyka a v překladu znamená „výměna“. Hlavní myšlenkou swapu je omezit zbytečné vyhazování věcí, které jsou v dobrém stavu a pro které může někdo dále najít využití. Účastníci swapu si mezi sebou vyměňují své nepotřebné věci na oplátku za předměty, které aktuálně shánějí. V rámci swapu nemusí docházet pouze k výměně věcí. Občas

lidé nabízí také své služby či věci na zapůjčení, případně mnoho lidí daruje věci zdarma. Myšlenka swapu by měla napomoci zmírnění dnešního konzumerismu naší společnosti, kdy v mnoha případech je levnější koupit novou věc než nechat opravit starou. Za tuto nízkou výrobní cenu platíme ale našim životním prostředím a vykořisťováním pracovníků v chudších částech světa.

Cíl práce

Cílem této práce je vytvořit aplikaci, která usnadní účastníkům swapu organizaci výměn předmětů a hromadných swap akcí. V rámci práce provedu dotazníkové šetření na jehož základě vyhodnotím požadavky na aplikaci. Z analýzy tohoto problému bude poté vycházet celkový návrh aplikace, implementace jejího prototypu a následné uživatelské testování. Ve výsledku by měla práce obsahovat teoretickou část analýzy, návrhu, implementace a testování a praktickou část funkčního prototypu aplikace.

Kapitola 1
Analýza

Tato kapitola se věnuje identifikaci problémů, se kterými se potýkají členové swap komunity při používání dostupných nástrojů a aplikací k domlouvání swapu. Na základě poznatků z dotazníkového šetření jsem sestavila výčet požadavků na vyvíjenou aplikaci. V sekci rešerše se poté věnuji zkoumání již existujících aplikací a určuji, zda-li dané požadavky splňují. V závěru poté představuji případy užití aplikace a diagramy konkrétních aktivit.

Analýza dané domény a problémů s ní souvisejícími je důležitou fází v procesu softwarového vývoje. Poznatky získané z analytické části této práce budou následně použity při návrhu prototypu aplikace.

1.1 Dotazníkové šetření

Pro vyhodnocení požadavků na aplikaci jsem provedla dotazníkové šetření. Dotazník byl sestaven tak, aby oslovil jak aktivní účastníky swapu, tak i potenciální uživatele, kteří neměli o tomto fenoménu prozatím ponětí. Výsledná aplikace by měla tak přilákat nejen již aktivní swapery, ale i nové uživatele, kterým například nevyhovoval dosavadní proces, jakým swap probíhá, a proto se ho také neúčastnili.

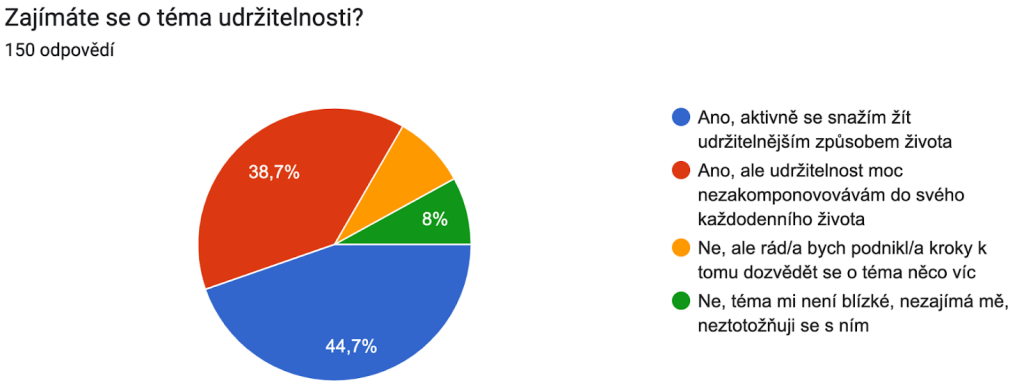
Demografické údaje

Dotazník jsem nasdílela mezi přátele na Facebooku a také ve facebookových skupinách a na platformách, kde se lidé kolem swapu sdružují. Celkem odpovědělo 150 respondentů, z toho 59,3 % byly ženy, 38 % muži a zbytek jinak se identifikující pohlaví nebo nechtěli odpovídat. Největší věková kategorie v zastoupení 74,4 % byli lidé ve věku od 18 do 30 let včetně, potom lidé mezi 31 a 40 lety v zastoupení 17,3 % a zbytek lidé pod 18 let nebo mezi 41 až 60 lety. Téměř polovina respondentů pocházela z Prahy, což mohlo být mimo jiné ovlivněno sdílením ve

facebookové skupině Swap Prague, která je největší takovouto skupinou v Česku. Zbytek krajů má zastoupení respondentů v přibližně podobných počtech.

Výsledky otázek ohledně swapu

První otázkou dotazníku bylo, zda-li se respondenti vůbec zajímají o téma udržitelnosti. Jak lze vidět z grafu 1.1, 92.1 % všech dotázaných se buď snaží aktivně žít udržitelným způsobem života nebo se o téma alespoň zajímá či chce podniknout určité kroky, jak zakomponovat udržitelnost do svého života. Všichni tito lidé mohou být potenciálními uživateli aplikace vyvíjené v rámci této práce.

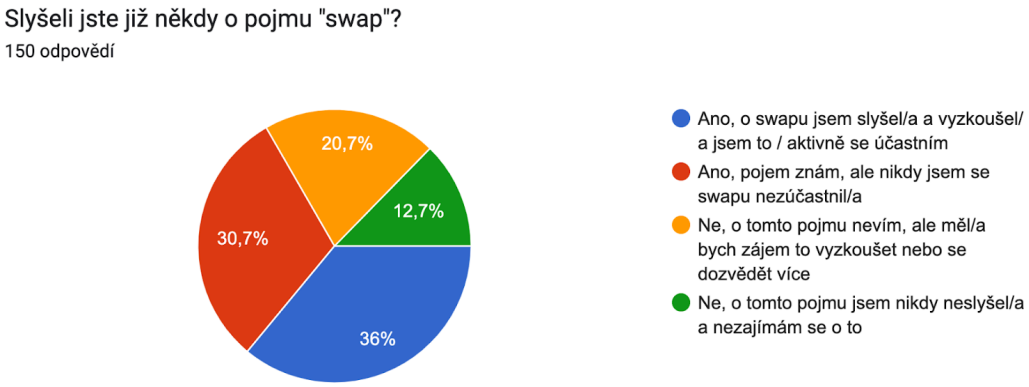


Obrázek 1.1 Výsledek první otázky ohledně zájmu o udržitelnost mezi respondenty dotazníku.

Další otázka se zabývala tím, jestli respondenti mají ponětí o pojmu swap, případně jestli se ho aktivně účastní nebo někdy zúčastnili. Na obrázku 1.2 lze vidět, že pouze 12.7 % všech dotázaných zodpovědělo, že o pojmu nikdy neslyšeli a zároveň se o toto téma vůbec nezajímají. Můžeme tedy předpokládat, že zbylých 87.4 % má ve větší či menší míře o swap určitý zájem, ať už se pouze chtějí jen dozvědět více či jsou aktivními účastníky.

Následující otázky byly dobrovolné a určené pouze pro respondenty, kteří se swapu již někdy zúčastnili nebo se pravidelně účastní.

Uživatelé měli uvést, jakým způsobem nejčastěji hledají věci na výměnu. Nejvíce zastoupeným způsobem bylo hledání přes facebookové skupiny, které využívá 64.1 % respondentů. S téměř vyrovnanými procenty poté lidé uvedli, že se účastní veřejně pořádaných swap akcí (33.3 %), používají platformu Vinted [1] (32.1 %) a využívají online bazarů jako je FB Marketplace [2], SBazar [3] nebo Bazoš [4] (29.5 %). Uživatelé měli také možnost uvést jiné způsoby, kterých využívají. Jeden z respondentů uvedl, že používá aplikaci Nevyhazuj to [5], která slouží k nabízení věcí zdarma za odvoz. Uvedl také, že aplikace ale není příliš funkční, často se seká nebo mizí chaty s ostatními uživateli.



Obrázek 1.2 Výsledek otázky týkající se zájmu respondentů o fenomén swapu.

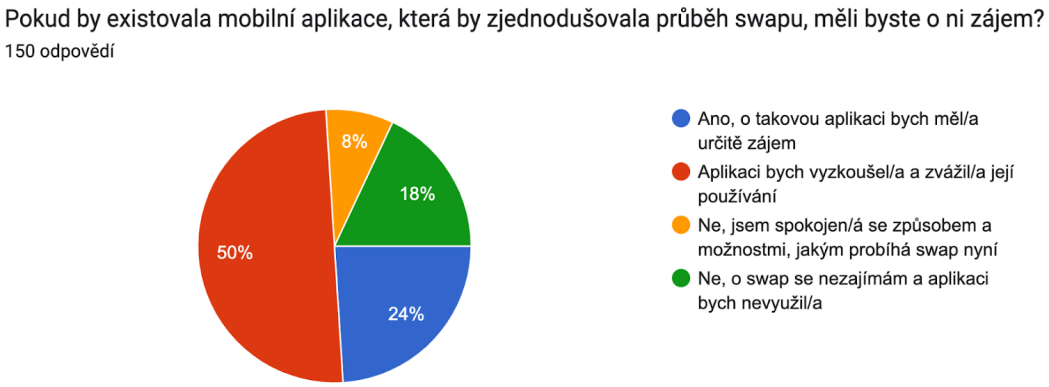
V následující otázce měli respondenti zodpovědět, zda-li jsou spokojeni se způsobem, jakým domlouvají swap věcí nyní. 40.3 % respondentů odpovědělo, že ne. V následující otázce ale uvedlo celkem 63.6 % dotázaných nějaký z faktorů, které jim nevyhovují na dosavadních způsobech, jakým se swapu účastní.

- **67.3 %** respondentů uvedlo, že jim vadí **nepřehlednost vyhledávání předmětů** a zároveň stejnému procentu uživatelů vadí i **nepřehledná komunikace** s ostatními účastníky swapu.
- **42.9 %** by potom ocenilo zjednodušení označení předmětů jako rezervovaných či již vyměněných.
- Jako další možnosti pak někteří uvedli, že by chtěli **zodpovědnější přístup uživatelů** ke swapu či **zjednodušení focení a nahrávání předmětů**, které zabere relativně dost času.

Další otázka byla opět povinná pro všechny respondenty. Dotázaní měli zodpovědět, zda-li by měli zájem o aplikaci, která by celý proces swapu zjednodušovala, ať už by se jednalo o přehlednější vyhledávání předmětů či jednodušší komunikaci s uživateli. 74 % respondentů by mělo o takovou aplikaci zájem či by ji chtěli alespoň vyzkoušet a zvážit její používání, jak znázorňuje graf na obrázku 1.3. Tento výsledek mluví poměrně kladně ve prospěch vývoje alternativní platformy pro výměnu předmětů ve swap komunitě.

V poslední otázce dotazníku měli respondenti uvést, jaké funkcionality by ocenili od aplikace, která by usnadňovala průběh swapu.

- **93.1 %** by ocenilo **vyhledávání předmětů podle názvu, lokality a kategorie.**
- **81 %** by se líbily uživatelské profily s **veřejným hodnocením uživatelů** na základě proběhlých swapů.



Obrázek 1.3 Zájem respondentů dotazníku o alternativní aplikaci k již existujícím způsobům domluvy swapu.

- **63.8 %** by využilo **přehledné album předmětů nahraných uživatelem** (ať už se jedná o přehled předmětů nahraných uživatelem samotným nebo zobrazení všech předmětů u profilu ostatních uživatelů).
- **52.6 %** by ocenilo **přehled veřejně pořádaných swap akcí** v okolí.

1.2 Požadavky na aplikaci

Určení požadavků na aplikaci slouží jako podklad k následnému návrhu architektury vyvíjeného prototypu. Na základě požadavků budou navrženy konkrétní funkcionality, které by měla aplikace obsahovat.

1.2.1 Funkční požadavky

Funkční požadavky představují, jaké funkcionality by měla aplikace pokrývat. Na základě těchto požadavků budou navrženy a implementovány prvky v prototypu aplikace, které budou uživateli jednoznačně viditelné.

- **FP1: Vytvoření uživatelského profilu a jeho správa**

  Uživatelé se budou moci do aplikace registrovat a vytvořit si uživatelský účet. Svá uživatelská data budou poté moci spravovat ve svém účtu.

- **FP2: Nahrávání a správa předmětů na výměnu**

  Uživateli bude umožněno nahrávat své předměty na výměnu. Ke každému předmětu bude možné vyplnit základní údaje jako název, fotografie předmětu, kategorie a případný popis. Uživatel bude také moci své předměty spravovat, tedy editovat je či mazat.

- **FP3: Vyhledávání předmětů**

  V aplikaci bude možné zobrazit přehled předmětů nahraných ostatními uživateli. Bude také možné vyhledávat předměty podle názvu a filtrovat je podle kategorie a lokace, případně dalších parametrů.

- **FP4: Vyhledávání a zobrazení profilů ostatních uživatelů**

  Uživatel bude moct vyhledat profily ostatních uživatelů dle jejich uživatelského jména. Bude si moct zobrazit jejich profil a přehled předmětů, které nabízejí na výměnu.

- **FP5: Posílání zpráv mezi uživateli**

  Uživatelé se budou moci v soukromých zprávách kontaktovat ohledně výměny předmětů či jiných informací týkající se swapu.

- **FP6: Veřejné hodnocení uživatelů**

  Na základě proběhlých výměn a komunikaci s ostatními členy budou uživatelé moci dávat hodnocení ostatním uživatelům, které bude sloužit jako určitá forma reputace uživatelů v aplikaci, což by mělo podpořit férovější jednání účastníků swapu.

- **FP7: Přehled veřejných swap akcí**

  Uživatelé si budou moct zobrazit přehled veřejně pořádaných swap akcí. Uživatelé budou také moci zakládat nové swap události, které si mohou ostatní uživatelé zobrazit a zúčastnit se jich.

1.2.2 Nefunkční požadavky

Nefunkční požadavky určují prvky aplikace, které nedefinují přímo funkcionalitu daného systému, ale ovlivňují celkovou funkčnost aplikace a uživatelův zážitek při jejím používání.

- **NP1: Mobilní aplikace**

  Aplikace bude vyvinuta pro mobilní zařízení, což poskytuje uživateli rychlost a flexibilitu při focení a nahrávání předmětů na výměnu.

- **NP2: Rychlost odpovědi na požadavek**

  Aplikace by měla na požadavek odpovědět do 2s. Podle výzkumu od Googlu, pravděpodobnost, že uživatel opustí aplikaci, která se načítá mezi 1 až 3 sekundami je 32 %. Pokud ale aplikaci trvá načtení mezi 1 až 5 sekundami, zvyšuje se pravděpodobnost opuštění aplikaci na 90 % [6].

- **NP3: Dostupnost aplikace**

  Aplikace by měla být dostupná 99,5 % času, aby byl zaručený příjemný uživatelský zážitek. Dostupnost 99,5 % času ročně znamená, že aplikace bude ročně nedostupná po necelé 2 dny. Měl by to být dostatek času na případné nutné údržby systému, které vyžadují nedostupnost aplikace.

1.3 Rešerše podobných aplikací

Ačkoliv je možné zprostředkovat swap i offline způsobem (např. účastnit se různých veřejně pořádaných akcí či se domluvit v rodině nebo mezi kamarády), většina organizace se děje online na internetu. Uživatelé k domluvě používají různé platformy. Jak už vyšlo najevo z dotazníkového šetření 1.1, nejpopulárnějším způsobem jsou pravděpodobně facebookové skupiny. Kromě tohoto způsobu existují i další platformy. Pro důkladnou analýzu řešeného problému je důležité se podívat i na dosavadní způsoby, jakým se lidé swapu zúčastňují, a vyhodnotit, které prvky jsou uživateli oceňovány a co jim u současných řešení chybí.

1.3.1 Facebookové skupiny

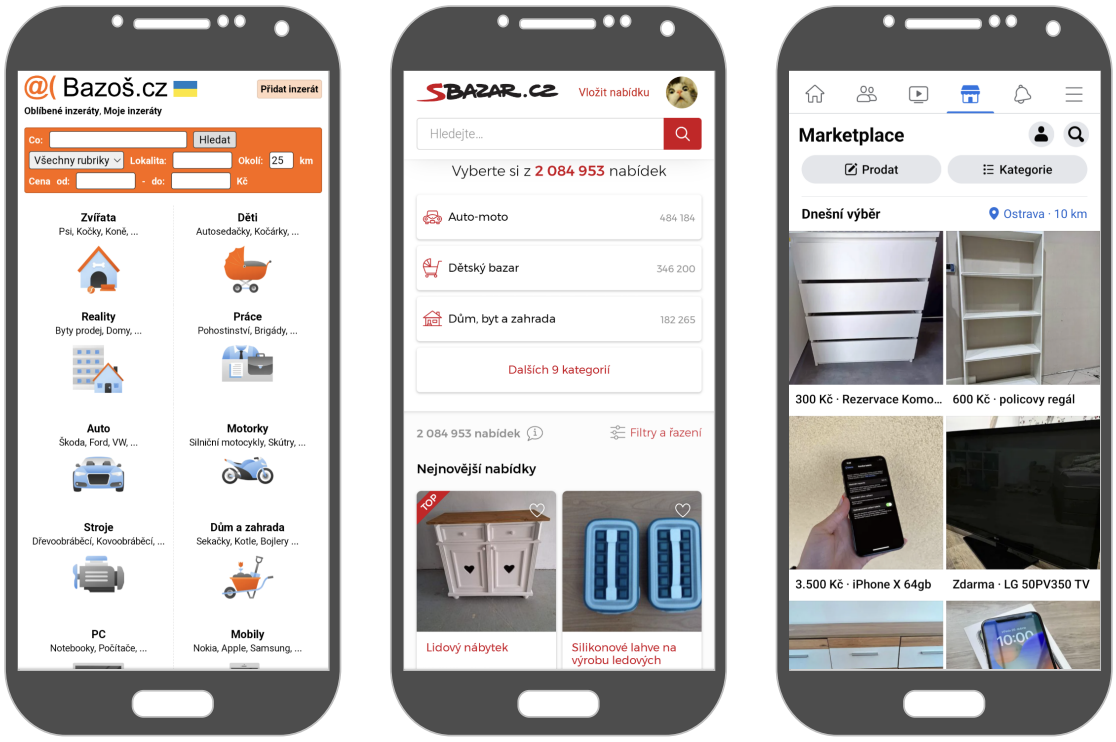
Nejčastějším online způsobem, jakým lidé v Česku vyhledávají a nabízejí předměty na swap, jsou skupiny na platformě Facebook. Tyto skupiny často čítají velké množství uživatelů. Například největší skupina, která sdružuje účastníky swapu v Praze, Swap Prague, má k dnešnímu dni 38.8 tisíc uživatelů. Další skupiny, shromažďující swapery v ostatních větších městech, mívají počet uživatelů v rámci tisíců.

Nabídka a poptávka předmětů a služeb na výměnu v těchto skupinách většinou probíhá tak, že uživatelé napíšou příspěvek s tím, co nabízí a co shánějí na oplátku a také případně specifikují místo předání a datum. Problémem tohoto přístupu je jeho nepřehlednost. Ve větších skupinách může přibývat 10 až 20 příspěvků denně, takže mnoho příspěvků se hned ztratí v záplavě nových. Vyhledávání konkrétních věcí na poptávku lze poté provádět jenom podle klíčových slov v textu příspěvků nebo podle fotek v hromadné fotogalerii. V tomto ohledu je využívání těchto skupin poměrně neoptimální.

Dalším faktorem je poté komunikace s ostatními účastníky. Může se stát, že se lidé domluví na osobním setkání a pak jedna strana nepřijde. V těchto skupinách nelze úplně jednoduše vyfiltrovat nespolehlivé uživatele. Čas od času se objeví ve skupině příspěvek, kde uživatel varuje ostatní swapery před konkrétním uživatelem, který nedorazil na místo předání, přestal komunikovat nebo se chová neférově.

1.3.2 Online bazary

K výměně předmětů na internetu využívají lidé i jiné online portály, které neslouží primárně ke swapu. Bazoš, Sbazar či Facebook Marketplace slouží primárně jako portály k nákupu a prodeji použitého zboží z druhé ruky. Občas lidé něco darují zadarmo nebo někdo shání věci výměnou. Tyto portály jsou přehlednější v ohledu vyhledávání a filtrování předmětů a následné komunikace s ostatními uživateli. Na FB Marketplace můžete po provedeném obchodě udělit uživateli i hodnocení. U těchto portálů je problémem, že slouží primárně k prodeji a nákupu a ne směnnému obchodu, proto může být pro uživatele poměrně nepřehledné hledat mezi všemi nabídkami ty, kde jsou ostatní ochotni i směnit místo prodeje. Na obrázku 1.4 jsou ukázky toho, jak stránky těchto bazarů vypadají v zobrazení na mobilním zařízení.

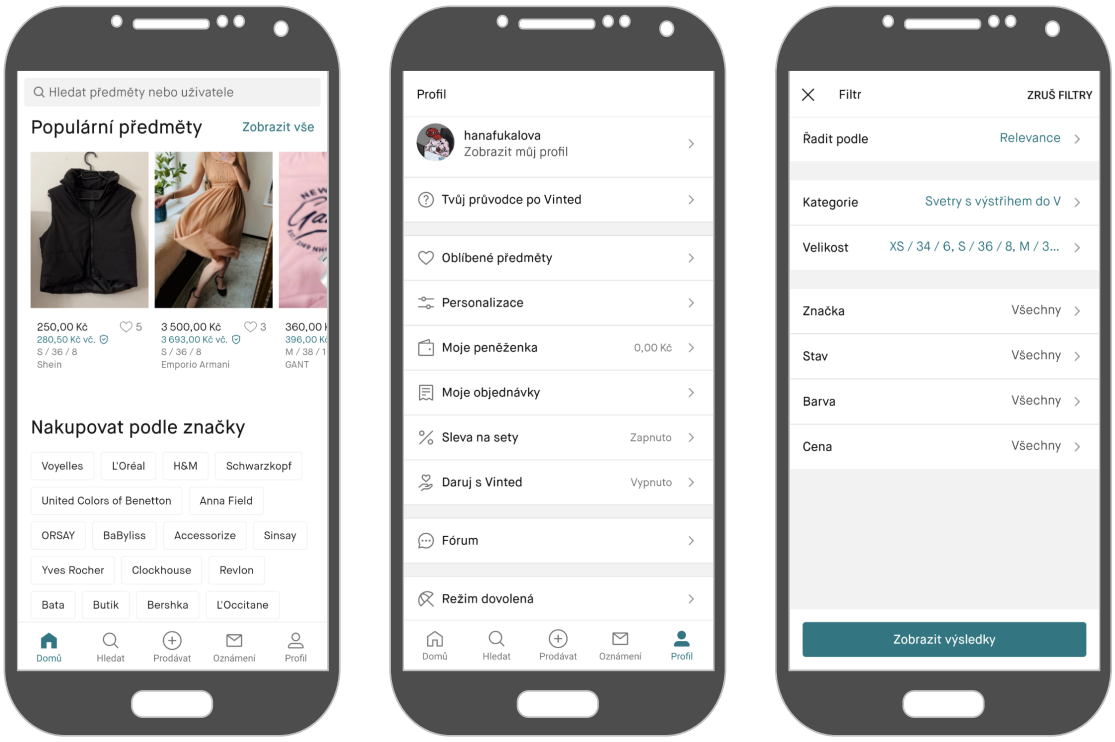


Obrázek 1.4 Ukázka webových stránek online bazarů v zobrazení na mobilním zařízení. Zleva Bazoš, SBazar a FB Marketplace.

1.3.3 Vinted

Vinted je internetová platforma sloužící k prodeji, nákupu a případné výměně oblečení a doplňků. V počátcích své existence aplikace umožňovala uživatelům i směňovat své oblečení a doplňky a poskytovala k tomu potřebné uživatelské rozhraní, kde si uživatel mohl vyfiltrovat pouze předměty, které chtějí ostatní směnit. S postupnou monetizací platformy v posledních letech ale z portálu úplně vymizela

oficiální podpora vyměňování předmětů a nyní je aplikace především o prodeji a nákupu, takže nově příchozí uživatelé o možnosti směny už ani nemají často ponětí. Vinted nově nabízí i objednání dopravy integrované přímo v aplikaci, čímž umožňuje uživatelům prodávat a nakupovat i mimo rámec svého bydliště, ale tuto funkcionalitu není možné využít, pokud by chtěli uživatelé své předměty pouze směnit. Portál je zaměřen pouze na oblečení, doplňky, případně i kosmetiku. Pokud uživatelé chtějí směňovat jiné předměty, mohou založit příspěvek ve veřejném fóru, které ale není zcela přehledné a příspěvky se v něm lehce ztratí. Na obrázku 1.5 jsou zobrazeny obrazovky Vinted mobilní aplikace.



Obrázek 1.5 Ukázka obrazovek mobilní aplikace Vinted. Zleva lze vidět postupně domovskou obrazovku s přehledem populárních předmětů, možnosti uživatelského profilu a vzhled vyhledávání a filtrování mezi předměty.

1.3.4 Leepa

Leepa je společnost propagující udržitelnou módu, která nabízí svým uživatelům mimo jiné také možnost prodat své oblečení v online second-hand obchodě. Součástí jejich online platformy je i webová aplikace, která umožňuje uživatelům vyměňovat předměty s ostatními uživateli, v podstatě tedy implementuje myšlenku swapu. Uživatelé si mohou vytvořit účet, nahrávat své věci na výměnu a vyhledávat a filtrovat věci ostatních uživatelů. Aplikace nabízí uživatelům také zajímavý prvek, jak nalézt věci na výměnu. Buď mohou prohlížet zeď s příspěvky ostatních

a nebo si mohou zapnout druhý mód, který funguje na podobném principu jako různé seznamovací aplikace. Uživateli je vždy zobrazena jedna položka, u které může rozhodnout kliknutím na určitá tlačítka, zda o ni má nebo nemá zájem. Pokud uživatel zvolí možnost, že má o předmět zájem, a zároveň majitel toho předmětu projevil zájem o něco z nabídky prvního uživatele, dojde k propojení a uživatelé se mohou v chatu domluvit na dalších detailech ohledně předání.

Platforma je dostupná pouze jako webová aplikace a také nenabízí uživatelům možnost udělovat hodnocení na základě zkušeností z výměny.

1.3.5 Výsledky rešerše

V tabulce 1.1 jsem na základě rešerše zkoumaných aplikací určila, zda-li tyto aplikace splňují funkční požadavky určené v sekci 1.2.1.

Aplikace	FP1	FP2	FP3	FP4	FP5	FP6	FP7
FB skupiny	√	P	X	P	P	X	X
Online bazary	√	P	X	P	√	P	X
Vinted	√	P	X	√	√	√	X
Leepa	√	√	√	P	√	X	X

Tabulka 1.1 Tabulka znázorňuje, zda-li jednotlivé aplikace splňují funkční požadavky specifikované v 1.2.1. √ značí, že ano, P splňují požadavek částečně, ale ne optimálně, X nesplňují.

1.4 Případy užití

Specifikace případů užití slouží k detailnějšímu popisu funkčních požadavků na aplikaci z pohledu uživatele. Každý funkční požadavek by měl být pokryt alespoň jedním případem užití, jinak je daná funkcionalita v aplikaci nepotřebná. Každý případ užití by měl poté náležet alespoň k jednomu funkčnímu požadavku, v opačném případě to znamená, že daný případ není implementován žádnou funkcionalitou v aplikaci. V tabulce 1.2 jsem znázornila toto vzájemné pokrytí případů užití a funkčních požadavků. Jak jednotlivé případy užití na sebe navazují je dále znázorněno v diagramu 1.6.

- PU1 – Registrace
- PU2 – Přihlášení
- PU3 – Zobrazení uživatelského profilu
- PU4 – Změna uživatelských údajů

- PU5 – Odhlášení
- PU6 – Zobrazení uživatelem nahraných předmětů
- PU7 – Nahrání předmětu
- PU8 – Zobrazení detailu uživatelova předmětu
- PU9 – Editace údajů předmětu
- PU10 – Smazání předmětu
- PU11 – Zobrazení přehledu nabízených předmětů
- PU12 – Vyhledávání předmětu
- PU13 – Filtrování předmětů podle kategorie a lokace
- PU14 – Zobrazení detailu předmětu
- PU15 – Zobrazení výsledků vyhledávání
- PU16 – Zobrazení profilů ostatních uživatelů
- PU17 – Vyhledání uživatelských profilů
- PU18 – Zobrazení profilu uživatele
- PU19 – Udělení hodnocení uživateli
- PU20 – Zaslání zprávy uživateli
- PU21 – Zobrazení všech konverzací uživatele
- PU22 – Zobrazení přehledu swap události
- PU23 – Založení veřejné swap události
- PU24 – Zobrazení detailu swap události

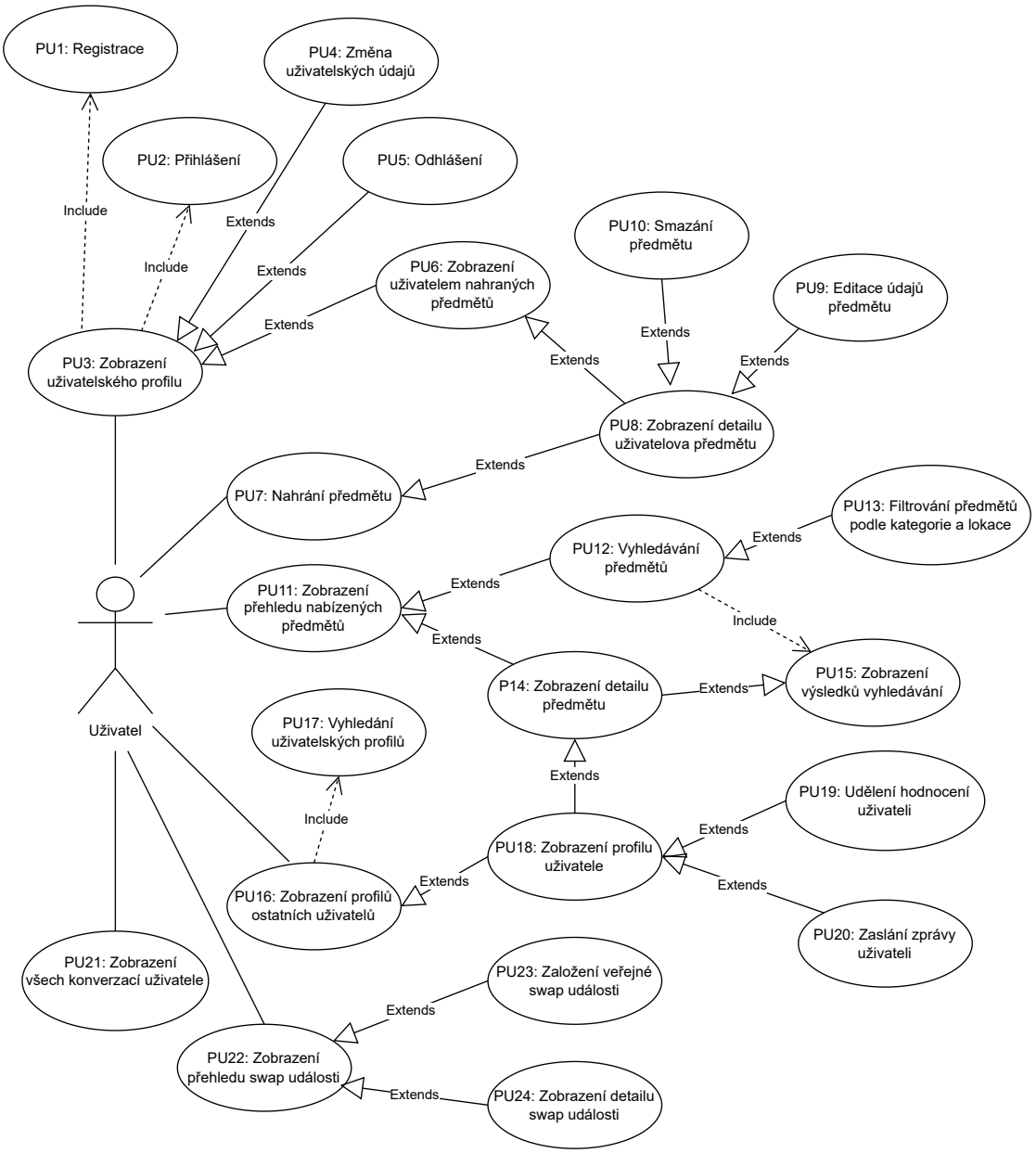
	FP1	FP2	FP3	FP4	FP5	FP6	FP7
PU1	√	-	-	-	-	-	-
PU2	√	-	-	-	-	-	-
PU3	√	-	-	-	-	-	-
PU4	√	-	-	-	-	-	-
PU5	√	-	-	-	-	-	-
PU6	-	√	-	-	-	-	-
PU7	-	√	-	-	-	-	-
PU8	-	√	-	-	-	-	-
PU9	-	√	-	-	-	-	-
PU10	-	√	-	-	-	-	-
PU11	-	-	√	-	-	-	-
PU12	-	-	√	-	-	-	-
PU13	-	-	√	-	-	-	-
PU14	-	-	√	-	-	-	-
PU15	-	-	√	-	-	-	-
PU16	-	-	-	√	-	-	-
PU17	-	-	-	√	-	-	-
PU18	-	-	-	√	-	√	-
PU19	-	-	-	-	-	√	-
PU20	-	-	-	-	√	-	-
PU21	-	-	-	-	√	-	-
PU22	-	-	-	-	-	-	√
PU23	-	-	-	-	-	-	√
PU24	-	-	-	-	-	-	√

Tabulka 1.2 Tabulka znázorňující vzájemné pokrytí funkčních požadavků ze sekce 1.2.1 a případů užití.

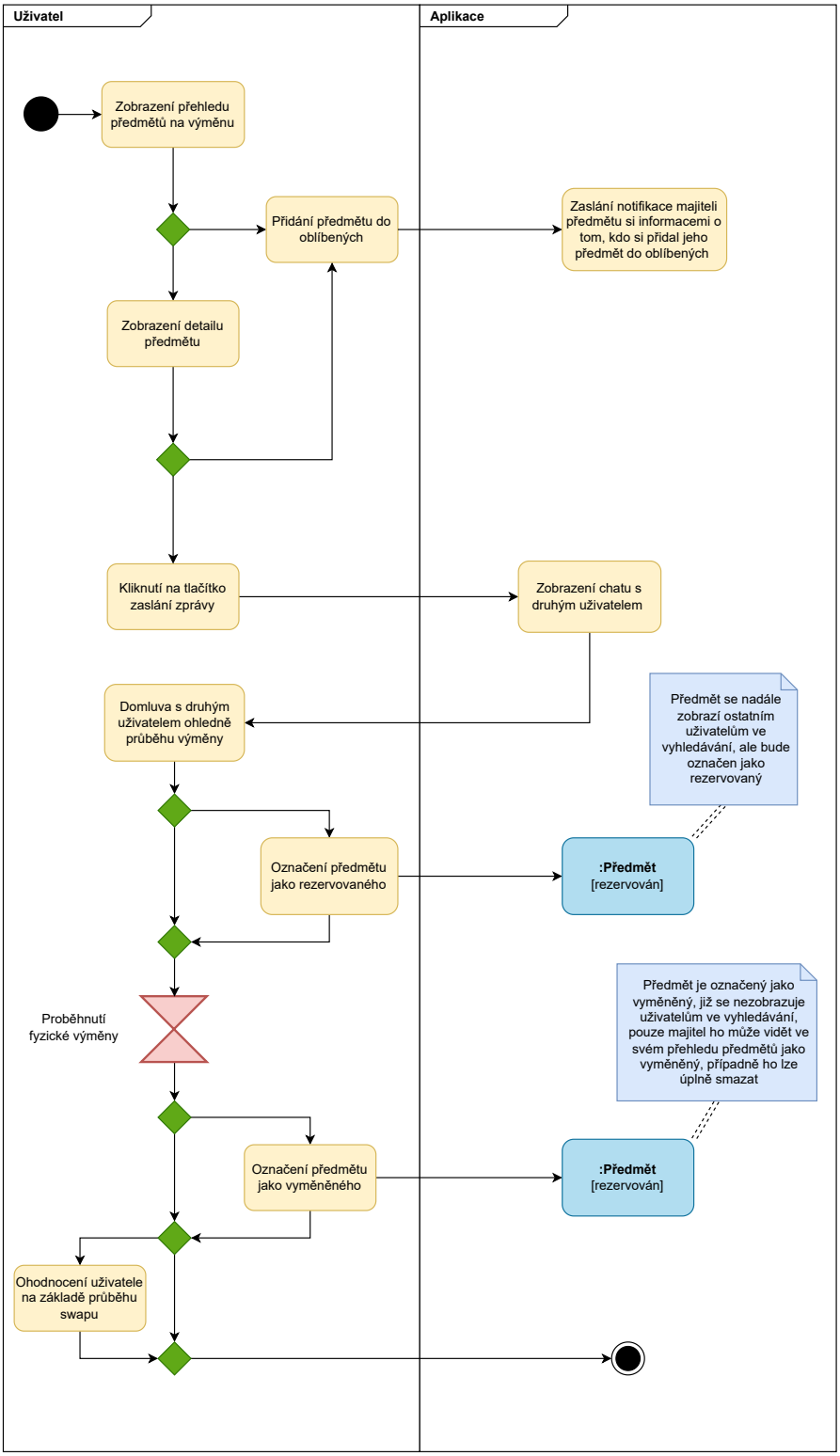
1.5 Diagram aktivit

Diagram aktivity slouží k detailnějšímu popisu komplexnějších případů užití aplikace. Některé z procesů v aplikaci mohou vyžadovat složitější proces, který nelze zobrazit pouze diagramem případu užití, proto jsou jednotlivé aktivity přehledněji znázorněny v diagramu.

V navrhované aplikaci se bude nacházet jeden komplexnější proces a tím je celkový proces výměny předmětů mezi uživateli. Uživatel podnikne kroky k vyhledání předmětu, o který by měl zájem, a poté si může přidat předmět do oblíbených, čímž příjde notifikace majiteli předmětu, nebo kontaktovat majitele předmětu přímo zásláním soukromé zprávy. Následně proběhne proces výměny a na závěr volitelné udělení hodnocení. Celý proces je znázorněn v diagramu aktivity 1.7.



Obrázek 1.6 Diagram případu použití.



Obrázek 1.7 Diagram aktivity výměny předmětu mezi dvěma uživateli v aplikaci.

Kapitola 2
Návrh

Tato kapitola se věnuje celkovému návrhu aplikace. V rámci návrhu jsem vytvořila doménový model, který znázorňuje jednotlivé entity a vztahy mezi nimi v doméně swapu tak, jak budou implementovány v aplikaci. Dále se v kapitole věnuji návrhu architektury aplikace a na jejím základě výběru vhodných technologií. Na závěr je v kapitole obsažen návrh uživatelského rozhraní, podle kterého se bude potom řídit implementace UI aplikace.

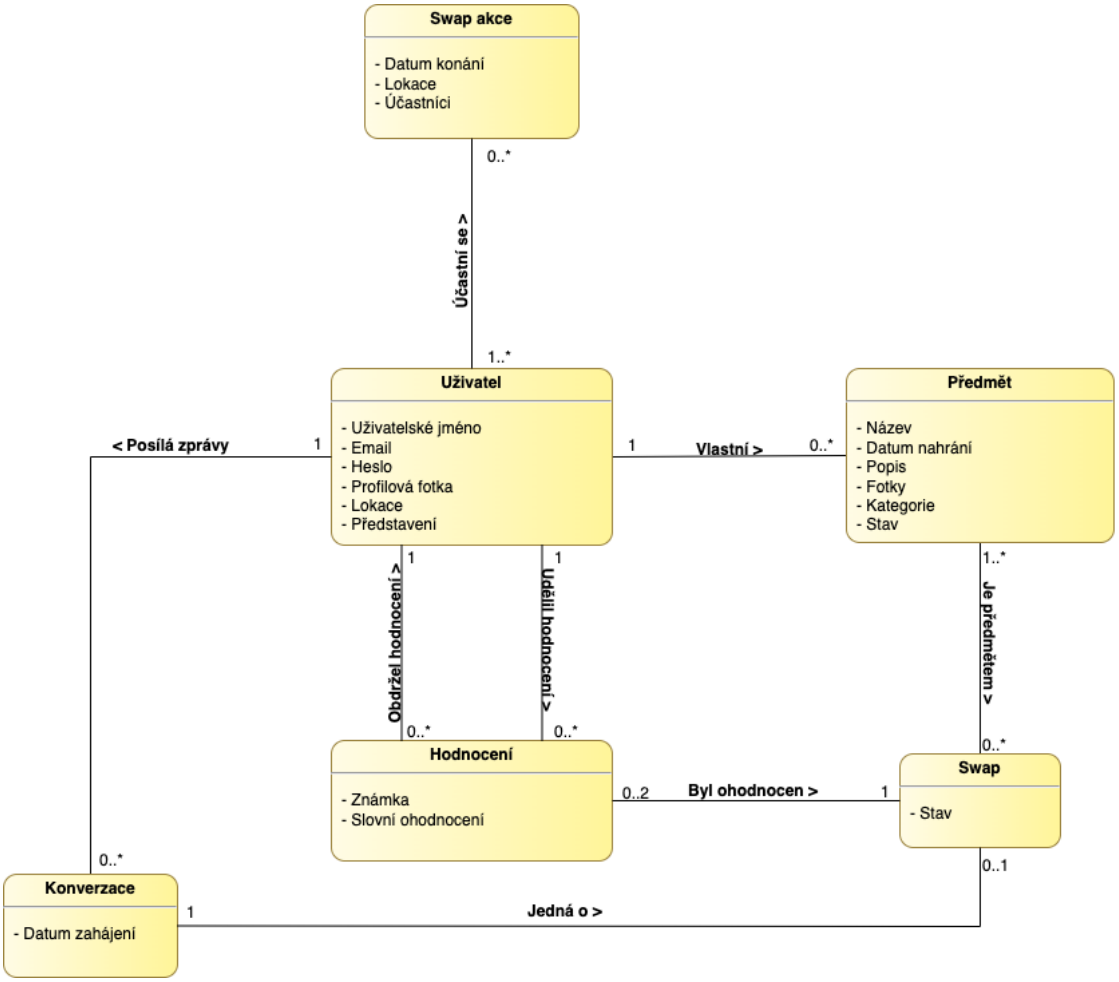
Fáze návrhu softwarového projektu je spolu s analýzou podstatnou částí celého procesu softwarového vývoje. Pokud je analýze a návrhu věnován dostatek času, čas samotné implementace softwaru se může poměrně velice zkrátit, případně se může předejít zbytečným chybám, které celý proces zdrží. Jelikož placení vývojářů může být vcelku drahou položkou v rozpočtu celého projektu, je v zájmu projektového manažera a zákazníka zbytečně fázi vývoje neprodlužovat.

2.1 Doménový model

Doménový model slouží k popsání domény navrhované aplikace pomocí entit a vztahů mezi nimi. Doménový model je nezávislý na jakékoliv implementační platformě a technologiích. Popisuje pouze entity v dané doméně tak, jak je lze logicky identifikovat v reálném světě. Při vytváření modelu jsem vycházela z vlastních znalostí domény swapu, funkčních požadavků na aplikaci a případů užití. Model je znázorněn na obrázku 2.1.

2.2 Technologie

Výběr vhodných technologií hraje důležitou roli při následném návrhu architektury aplikace. V této sekci se budu věnovat výběru platformy pro implementaci jak klientské aplikace a backendu, tak i dalších prvků aplikace.



Obrázek 2.1 Diagram znázorňující model domény swapu.

2.2.1 Android platforma

Počet uživatelů, kteří využívají pro přístup na internet mobilní zařízení, roste každým rokem. Pro srovnání, v roce 2015 bylo 31.16 % přístupu na webové stránky vygenerováno mobilními zařízeními. V roce 2022 to bylo již celých 59.16 % [7]. Mobilní aplikace rostou na popularitě díky kompaktnosti a praktičnosti mobilních zařízení, které lze jednoduše nosit u sebe.

Mobilní aplikace poskytují mnoho výhod oproti mobilním webovým aplikacím. Mobilní aplikace pracují optimálně s dostupnými zdroji operačního systému zařízení, jsou celkově rychlejší a umožňují uživateli interagovat s aplikací alespoň částečně i v offline módu díky lokálnímu úložišti dat a dalším funkcionalitám. Celkový uživatelský zážitek je příjemnější, jelikož mobilní aplikace jsou implementovány tak, aby se přizpůsobily různým rozměrům zařízení, pro což nemusí být webové stránky vždy optimalizovány. Mobilní aplikace také mohou těžit z různých funkcionalit zabudovaných přímo do mobilního zařízení, jako je kamera, seznam kon-

taktů, GPS a další. Mobilní aplikace také umožňují jednoduché upozornění na nové události v aplikaci pomocí notifikací. Podle průzkumu portálu eMarketer, strávili v roce 2022 uživatelé v USA 90.43 % času používáním mobilních aplikací při používání chytrých telefonů [8].

Na globální úrovni dominují trhu s mobilními operačními systémy Android a iOS, přičemž Android má převážný podíl se 72.21 %, iOS má 27.1 % [9]. Při vývoji prototypu aplikace v rámci této práce hrála roli nejen silnější uživatelská základna, ale také mé vlastní zkušenosti s vývojem aplikací pro Android platformu. S ohledem na všechny tyto faktory jsem se rozhodla implementovat klientskou část prototypu jako nativní mobilní aplikaci pro Android platformu..

2.2.2 Firebase

Pro implementaci backendové části aplikace jsem se rozhodla využít cloudové služby Firebase. Tato cloudová platforma nabízí širokou škálu nástrojů a služeb pro rychlý a efektivní vývoj mobilních a webových aplikací. Pro můj typ projektu, což je menší mobilní aplikace, je to ideální řešení, jelikož nevyžaduje složité a rozsáhlé backendové řešení. Protože se chci zaměřit spíše na implementaci klientské aplikace, potřebuji jednoduché, ale spolehlivé a rychlé řešení backendu.

Klíčové funkcionality, které by měla backend aplikace poskytovat, jsem identifikovala jako:

- Autentizace uživatelů
- Backendové funkce
- Databáze
- Uložiště pro multimédia
- Push notifikace

Firebase poskytuje služby pokrývající všechny funkcionality vyžadované v rámci této práce. Nabízí Spark plan, v kterém je většina služeb zdarma do určitého počtu aktivních uživatelů či invokací za měsíc [10]. V následujících odstavcích jsem popsala jednotlivé služby podrobněji.

2.2.2.1 Autentizace uživatelů

Firebase nabízí službu Authentication, díky které se může uživatel registrovat a přihlašovat do aplikace pomocí emailu a hesla, přes telefonní číslo nebo přes jiné poskytovale (Google, Apple, Facebook, Twitter, GitHub). Lze také integrovat službu s vlastním autentizačním systémem či vytvářet dočasné anonymní účty [11].

V rámci no-cost Spark plánu jsou všechny způsoby autentizace kompletně zdarma, kromě autentizace přes telefonní číslo, která má limit 10 000 autentizací/měsíc [10].

2.2.2.2 Backendové funkce

Vývoj backendových aplikací prošel během posledních dekád velkou transformací, od tradičního nasazování aplikací na fyzické servery, přes pronájmy virtuálních serverů až po nejnovější technologii – serverless funkce. Ty umožňují vývojářům přiřazovat prostředky podle aktuální potřeby a kompletně se vyhnout zodpovědnosti za správu infrastruktury, kterou přebírají cloudoví poskytovatelé. Díky tomu se vývojáři mohou soustředit čistě na psaní kódu a nemusí se starat o správu serverů. Oproti pronájmu cloudových serverů, kdy musí být vývojáři stále zodpovědní za správu a škálování infrastruktury, poskytují serverless funkce větší flexibilitu a jednoduchost. To se hodí pro případy, kdy je potřeba rychle a jednoduše vytvořit backendovou funkcionalitu bez složité infrastruktury. infrastruktury. Při vývoji prototypu aplikace v rámci této práce se chci soustředit spíše na tvorbu uživatelsky přívětivé klientské aplikace, kterou budou uživatelé využívat na svých mobilních zařízeních, proto bych ráda využila možnosti snadné implementace některých backendových funkcí přes cloudovou platformu.

Firebase nabízí Cloud Functions, což je služba, která poskytuje framework pro implementaci serverless funkcí. Tyto funkce lze využít například pro zpracování výpočetně náročných operací, které není vhodné provádět na klientském zařízení. Dále se hodí pro udržbu databáze a vyvolávání push notifikací na základě změn v databázi [12]. Firebase umožňuje bezplatné využívání Cloud Functions do 2 milionů invokací, 400 000 GB-sekund úložiště a 200 000 GHz CPU-sekund za měsíc.[10].

2.2.2.3 Databáze

Firebase nabízí dvě NoSQL databáze – Cloud Firestore a Firebase Realtime Database. Realtime Database ukládá data jako jednoduchý JSON strom, zatímco Firestore organizuje data do kolekcí a dokumentů, což je ideální pro komplexnější hierarchie dat. Firestore také poskytuje pokročilé dotazování a třídění nad daty a umožňuje efektivní získání potřebných dokumentů bez nutnosti získávat celý podstrom dat jako v Realtime Database. Realtime Database je vhodné pro jednoduše strukturovaná data, zatímco Firestore je doporučována pro organizaci komplexnějších datových struktur [13].

Firestore si účtuje primárně za operace prováděné nad databází jako je čtení, zápis a mazání, avšak s nižší sazbou než Realtime Database. Vývojáři mohou nastavit limity pro tyto operace, což jim umožní mít větší kontrolu nad náklady na používání databáze. Realtime Database je zpoplatněna na základě propustnosti a uložiště, ovšem s vyšší sazbou než Firestore.[13].

Na základě uvedených faktorů jsem se rozhodla pro využití Cloud Firestore jako datového úložiště pro svou aplikaci. Pro můj konkrétní případ užití je vhodnější díky možnosti komplexního strukturování dat a cenově se obě řešení jeví jako dostupná, zejména pro implementaci jednoduššího prototypu. Díky mé předchozí zkušenosti s Cloud Firestore bude také integrace databáze do aplikace jednodušší.

2.2.2.4 Úložiště pro multimédia

Pro vývoj prototypu aplikace je umožnění uživatelům nahrávat fotografie předmětů na výměnu klíčové. K tomuto účelu plánuji využít Cloud Storage od Firebase. Tato služba poskytuje možnost ukládání uživatelského obsahu s vysokou bezpečností a škálovatelností, kterou lze upravit podle potřeby [14].

S ohledem na požadavky aplikace a vývojový prototyp zatím zcela postačí bezplatný plán Spark, který nabízí úložiště o velikosti 5 GB, stahování 1 GB denně a až 20 000 operací stažení a 50 000 operací nahrání denně [10]. Pro budoucí provoz aplikace bude pravděpodobně potřeboa většího úložiště, ale Firebase nabízí příznivé cenové plány na základě velikosti úložiště a počtu provedených operací za den. Snadná integrace s ostatními službami, které chci v aplikaci použít, činí Cloud Storage jasnou volbou.

2.2.2.5 Push notifikace

Aby bylo možné uživatele informovat o změnách v aplikaci, je nutné implementovat push notifikace, což jsou notifikace, které jsou automaticky zasílány ze serverové strany, bez potřeby jakéhokoliv požadavku ze strany uživatele. Tyto notifikace mohou být vyvolány na základě změn v databázi nebo jako reakce na činnosti ostatních uživatelů v aplikaci. V mém případě mohou push notifikace například informovat uživatele o zájmu jiného uživatele o jeho nabízený předmět nebo o jakékoliv změně v průběhu swap akce, na kterou se již uživatel přihlásil.

Firebase nabízí službu Cloud Messaging pro implementaci push notifikací v aplikaci. Tato služba umožňuje posílat zprávy na klientská zařízení a může být integrována s Cloud Functions pro další pokročilejší využití [15]. Firebase také nabízí tuto službu kompletně zdarma [10].

2.2.2.6 Google Analytics

Kromě potřebných funkcionalit popsaných výše, disponuje Firebase také vynikající analytickou službou Google Analytics. Tato funkce umožňuje podrobnou analýzu využívání a chování uživatelů v integrované aplikaci, což může posloužit jak při testování aplikace, tak pro marketing aplikace. Tato funkce je také kompletně zdarma pro uživatele Firebase [16].

2.2.3 Chat SDK

Vzhledem k tomu, že uživatelé aplikace budou mít možnost chatovat o organizaci výměn předmětů a dalších záležitostech týkajících se swap komunity, plánuji implementovat v aplikaci funkci posílání zpráv v reálném čase. Jelikož hlavním účelem mé práce není vytvoření chatovací aplikace, využiji již existujících frameworků k implementaci této funkcionality.

Při hledání vhodné technologie jsem se soustředila převážně na tyto faktory:

- **Funkcionalita**

  V aplikaci budou k dispozici dva typy chatu – soukromé konverzace mezi dvěma uživateli a skupinové chaty, které se zaměří na organizaci swap akcí a další témata. Kromě toho je žádoucí, aby chat nabízel základní funkcionality, jakými jsou například možnost posílat obrázky a zobrazovat stav zpráv (např. zda byla odeslána, doručena a přečtena).

- **Kompabilita**

  Protože budu vyvíjet aplikaci s využitím Jetpack Compose – nového frameworku pro tvorbu uživatelského rozhraní v Android aplikacích, hledala jsem technologii, která nabízí Compose UI prvky pro implementaci chatu jako součást svého SDK. Dále jsem hledala řešení, které by bylo kompatibilní s Firebase Authentication a umožnilo tak synchronizaci s touto službou.

- **Cena**

  Sháněla jsem řešení, které by bylo ideálně pro menší projekty zdarma nebo za přívětivou cenu, neboť v současné fázi projektu nepočítám s žádným ziskem.

- **Aktuálnost a údržba SDK**

  Během hledání jsem kladla důraz na aktualitu a pravidelnou údržbu technologií. Dále jsem preferovala projekty s přehlednou dokumentací, napsanou ideálně v jazyce Kotlin.

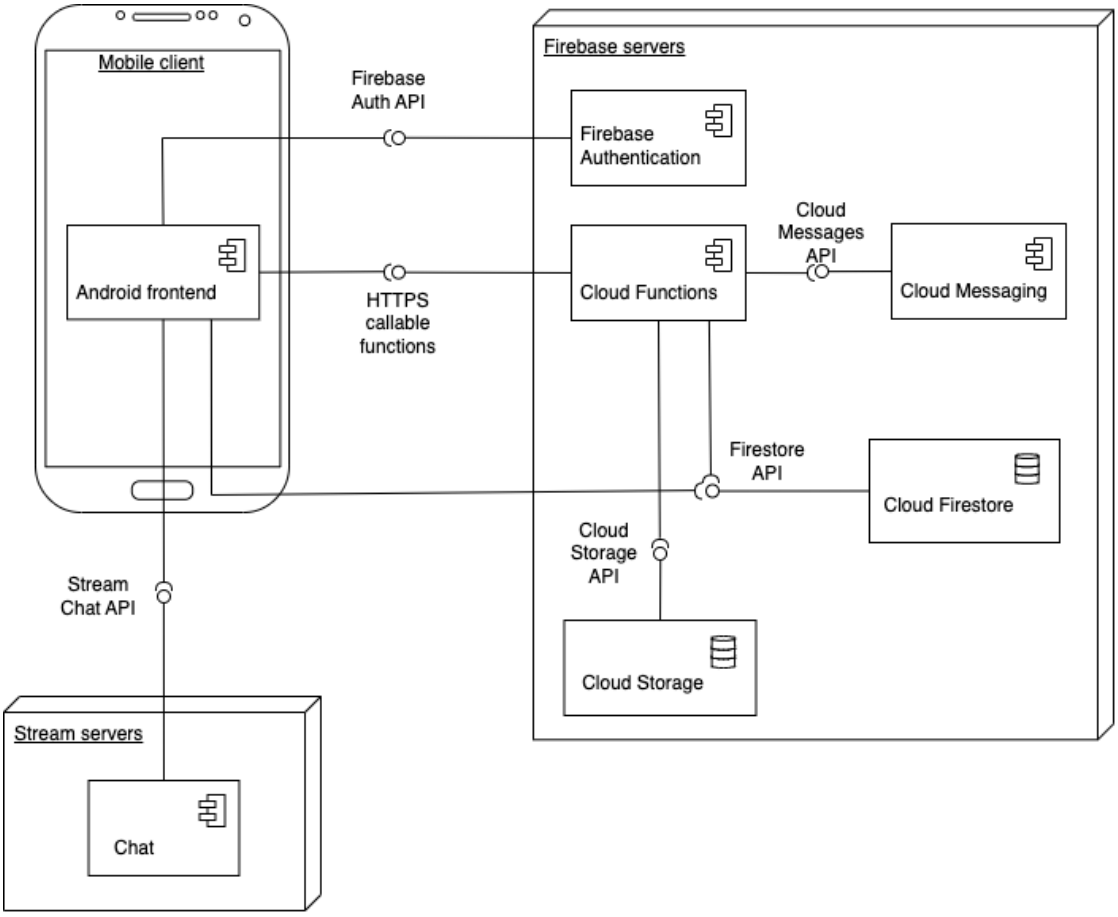
Po pečlivém zvažení mnoha možností jsem se rozhodla pro Chat SDK od Stream [17], které poskytuje jednoduchý způsob, jak implementovat chatovou funkcionalitu v Android aplikacích a přizpůsobit ji různým potřebám. Stream nabízí také SDK pro implementaci Compose UI komponent a spolupracuje s Firebase na rozšíření, které umožňuje snadnou autentizaci pomocí Firebase Authentication. Navíc je jeho velkou předností nabídka plánu pro menší společnosti, které mohou využívat SDK zdarma [18].

2.3 Architektura aplikace

Návrh architektury aplikace je klíčovým krokem v procesu softwarového vývoje, neboť ovlivňuje celkovou funkčnost, kvalitu a údržbu aplikace. Správně navržená architektura umožňuje efektivní vývoj, zjednodušuje údržbu a zvyšuje spolehlivost aplikace. Zahrnuje nejen volbu technologií, frameworků a databázových řešení, ale také rozdělení aplikace do logických modulů, definování vzájemných interakcí mezi nimi a vymezení rolí a pravomocí uživatelů. Díky správné architektuře mohou být v budoucnu snadno přidávány nové funkcionality, což umožní růst aplikace a přizpůsobení se požadavkům uživatelů. Navržení architektury před vývojem také

umožňuje předcházet možným problémům a chybám v implementaci a snižuje náklady na úpravy a údržbu aplikace v pozdější fázi vývoje.

Svou aplikaci jsem rozdělila na základě zvolených technologií na Android frontend aplikaci, která bude instalována na klientském mobilním zařízení. Tato aplikace bude přes API interagovat s Firebase backendovými službami. Aplikace bude také komunikovat se Stream Chat API, pomocí něhož budu implementovat funkcionalitu zasílání zpráv v aplikaci v reálném čase. Návrh této struktury je možné vidět na diagramu 2.2.



Obrázek 2.2 Návrh softwarové architektury rozdělený na klientskou Android aplikaci a backendové služby od Firebase a Stream.

2.3.1 Návrh architektury Android aplikace

Na poli softwarového vývoje existuje mnoho architektonických vzorů, které pokrývají různé domény aplikací. Aplikaci lze obecně rozdělit na serverovou a klientskou část, jak je popsáno v sekci 2.3. Dále je důležité zvolit vhodný architektonický vzor při implementaci obou těchto částí projektu. Jelikož backend aplikace budu

implementovat převážně pomocí služeb Firebase a Stream, zaměřím se v této sekci na návrh architektury Android klienta.

V Android vývoji jsou nejčastěji využívány tři návrhové vzory:

- Model–View–Controller (MVC)
- Model–View–Presenter (MVP)
- Model–View–View Model (MVVM)

Model-View-Controller

MVC je nejstarším z těchto architektonických vzorů a představuje základ pro následující dva. V této architektuře Model představuje datovou vrstvu, která buď sama poskytuje data nebo komunikuje s datovými úložišti. View obsahuje komponenty uživatelského rozhraní a Controller zprostředkovává komunikaci s uživatelem a aktualizuje Model a View. Controller a View jsou úzce svázány, každé View má jeden Controller. Jeden Model může být poté sdílen mezi více Controllery. Všechny komponenty mají na sebe vzájemně reference, jsou tedy navzájem závislé a úzce provázané [19].

Model-View-Presenter

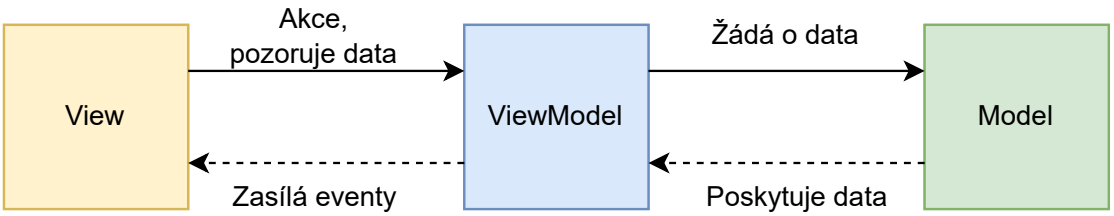
Tento architektonický vzor navazuje na MVC, snižuje však míru provázanosti jednotlivých komponent. Úloha Modelu zůstává stejná, nekomunikuje však již s View přímo ale přes Presenter. Presenter poté na základě dat z Modelu a vnitřní logiky aktualizuje View. View zprostředkovává komunikaci z uživatelem a zasílá informace o uživatelských akcích Presenteru, který aktualizuje na jejich základě Model. Tato změna cyklické závislosti mezi všemi třemi komponentami na lineární umožňuje lepší testovatelnost jednotlivých komponent a případnou výměnu některé z částí aplikace [20].

Model-View-View Model

MVVM představuje další vzor odvozený od MVC. Oproti MVP, MVVM přesouvá všechnu logiku chování View do View Modelu. View Model poskytuje stav, na jehož základě se View aktualizuje. Tento přístup umožňuje snadnější implementaci UI, oddělenou od jakékoliv business logiky aplikace. Implementace View vrstvy může být tak například delegována na designery, kteří nemají velké zkušenosti s programováním logiky aplikace, dokáží však pracovat s nástroji na tvorbu uživatelského rozhraní [19].

MVVM je vhodný architektonický vzor pro Android aplikace. Tento způsob rozdělení do vrstev se specifickými zodpovědnostmi je nativně podporován v Android frameworku, který nabízí množství užitečných architektonických komponent, včetně View Model třídy [21]. Při implementaci MVVM architektury v Android

aplikacích se obvykle používá jednosměrné řízení závislostí mezi jednotlivými komponentami. Komponenty View mají referenci na View Model, který získává data z Modelu a využívá publisher-observer návrhový vzor[1] pro publikování a čtení dat, na jejichž základě se aktualizuje stav View. Tento způsob tvorby závislostí umožňuje snadněji oddělit jednotlivé vrstvy programu a poskytuje efektivní správu a aktualizaci dat v rámci aplikace [22]. Komunikaci mezi těmito vrstvami jsem znázornila na diagramu 2.3. Díky použití MVVM architektury mohou vývojáři vytvořit flexibilní a snadno testovatelný kód. Studie z roku 2016 srovnávající MVC, MVP a MVVM vzory poukazuje na to, že MVVM vychází z těchto tří nejlépe z hlediska testovatelnosti, modifikovatelnosti a výkonu [19].



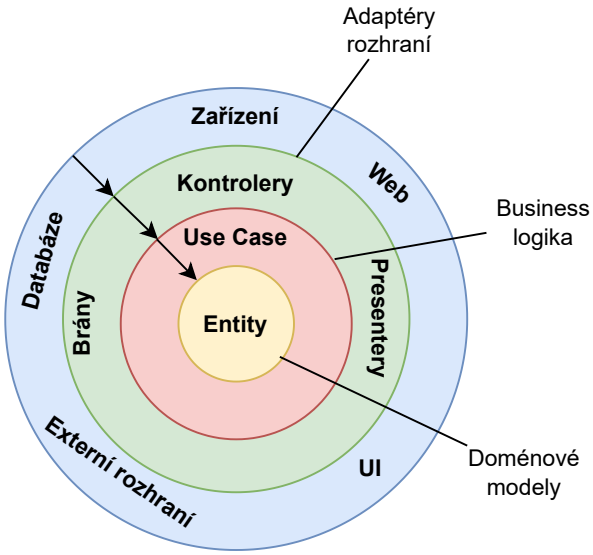
Obrázek 2.3 Znázornění implementace MVVM vzoru v Android aplikaci.

Clean Architecture

Clean Architecture je dalším z vývojových stupňů v softwarové architektuře. Tato architektura rozděluje komponenty do vrstev, které mají kruhový charakter, proto je také někdy nazývána jako „Onion Architecture“ (cibulová architektura). Hlavním pravidlem této architektury je, že závislosti kódu směřují pouze z vnějších vrstev do vnitřních. Kód vnitřních vrstev by neměl znát detaily o funkcích ve vnějších vrstvách. Znázornění jednotlivých vrstev lze vidět na diagramu 2.4. Tento princip umožňuje snadnou testovatelnost jednotlivých komponent kódu. Funkcionalita aplikace je jasněji čitelná přímo ze zdrojového kódu a hodí se obzvlášť na implementaci větších projektů, kde přibývají stále nové funkcionality [22].

Ve své aplikaci jsem se rozhodla spojit MVVM architektonický vzor s principy Clean Architecture. Aplikace bude mít tradiční View a View Model komponenty a také modelovou vrstvu, která bude pomocí rozhraní komunikovat s vnějšími službami. Komunikaci mezi datovou vrstvou a View Modelem bude zprostředkovávat několik jednotlivých Use Casů, kde každý případ užití zapouzdřuje jednu business logiku. Znázornění jednotlivých komponent a toku dat lze vidět na diagramu 2.5.

[1]Tento vzor popisuje funkcionalitu, kde jedna komponenta produkuje určitou informaci a ostatní komponenty se na ni registrují, aby vždy dostávaly aktuální data. Tento koncept v softwarovém vývoji bývá označován jako ”producent-konzument”.



Obrázek 2.4 Diagram znázorňující Clean Architecture principy. Přeložený a vycházející z diagramu zobrazeném v tomto článku [22].

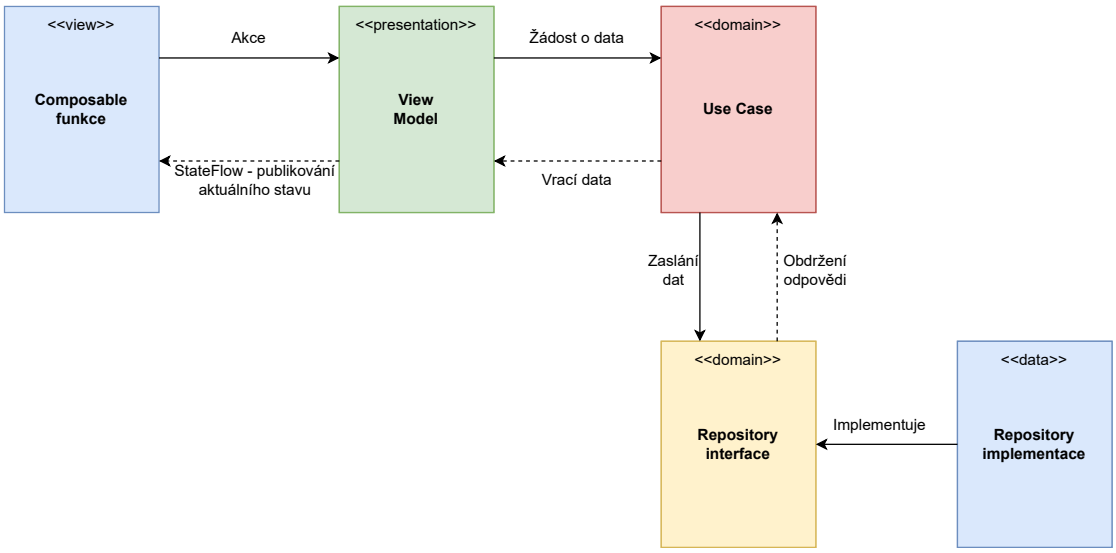
2.3.2 Návrh databáze

Databázi budu implementovat pomocí Cloud Firestore, což je NoSQL databáze. Datový model této databáze sestává z dokumentů ukládaných do kolekcí. Dokumenty mají dále členy namapované na konkrétní hodnoty. Tyto členy mohou být různých typů, od jednoduchých typů jako jsou řetězce a celá čísla až po vnořené objekty. Dokumenty také mohou obsahovat podkolekce [23].

Ve Firestore jsou definované tzv. root kolekce, tedy kořenové kolekce, které jsou na nejvyšší úrovni databáze. V mém návrhu struktury databáze 2.6 je 5 root kolekcí – Users (kolekce uživatelů), Items (kolekce předmětů), Reviews (kolekce hodnocení), Events (kolekce swap akcí) a Channels (data konverzací uživatelů). V těchto kolekcích jsou uloženy dokumenty pod unikátními ID. Struktura databáze je plochá, jednotlivé dokumenty neobsahují žádné podkolekce, pouze reference na dokumenty z ostatních kolekcí.

2.4 Uživatelské rozhraní

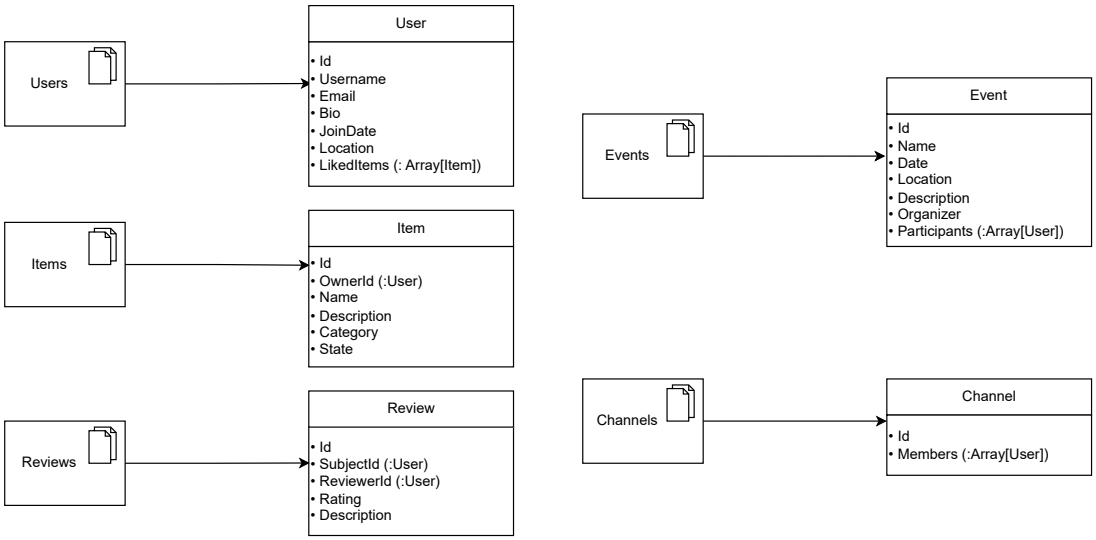
Před samotnou implementací aplikace je důležité rozvrhnout i to, jak bude vypadat uživatelské rozhraní aplikace, tedy obrazovky, které jsou viditelné uživateli a s kterými interaguje. Znázornění rozložení komponent na obrazovkách jsem rozkreslila ve wireframech. Wireframing je způsob návrhu uživatelského rozhraní aplikace na strukturální úrovni. Wireframe se obvykle používá k rozvržení obsahu a funkcionality na obrazovce s ohledem na potřeby uživatelů a jejich interakci s aplikací. Wireframy se využívají na začátku vývojového procesu k vytvoření základní struk-



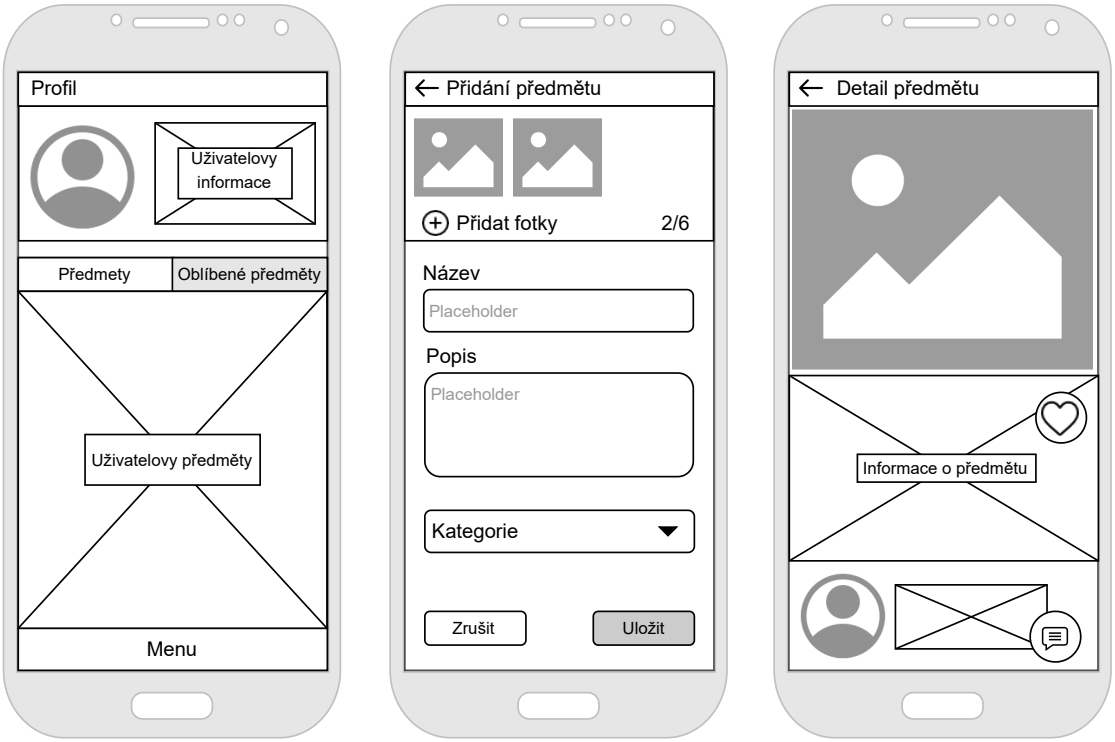
Obrázek 2.5 Diagram znázorňující implementaci MVVM vzoru v Android aplikaci s principy Clean architecture.

tury stránky před přidáním vizuálního designu a obsahu.

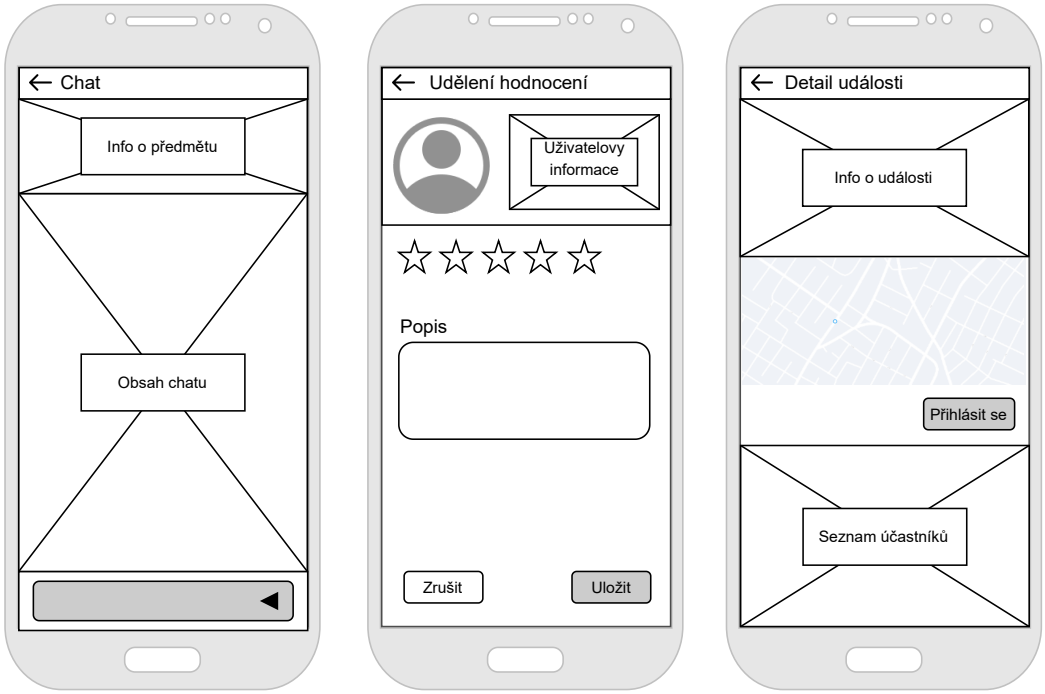
Na obrázku 2.7 je rozvržení komponent obrazovek profilu, přidání předmětu a detailu předmětu. Dále jsem navrhla vzhled obrazovky chatu, udělení hodnocení a detailu události 2.8. Aplikace bude obsahovat další obrazovky jako např. přihlášení, seznam předmětů či přehled událostí, tyto obrazovky však obsahují jednoduché komponenty jako seznamy předmětů, které nejsou nijak komplexní, proto jsem se rozhodla je z ukázek v této sekci vynechat.



Obrázek 2.6 Návrh NoSQL databázové struktury mé aplikace ve Firestore.



Obrázek 2.7 Zleva návrh obrazovek profilu, přidání předmětu a detailu předmětu.



Obrázek 2.8 Zleva návrh obrazovek chatu, udělení hodnocení a detailu události.

Kapitola 3
Implementace

V této kapitole se věnuji popisu procesu, jakým jsem implementovala prototyp aplikace, který je součástí praktické části této práce. První se věnuji implementaci klientské Android aplikace. Popisuji základní konfiguraci a strukturu projetku a propojení s Firebase. Dále se věnuji autentizaci uživatelů a UI tvořeného pomocí composable funkcí. Poté popisuji komunikaci mezi jednotlivými vrstvami aplikace, dependency injection a implementaci chatování v aplikaci. Na závěr pojednávám o cloud funkcích, které jsem implementovala jako backendovou část aplikace.

Po dokončení analýzy problému a návrhu řešení přichází v softwarovém vývoji fáze implementace. V této fázi programuji aplikaci na základě specifikací a návrhu. Zabývám se implementací Android aplikace, která bude komunikovat s backendovými službami poskytovanými Firebase a Stream. I když je důležité mít kvalitní návrh jako základ pro vývoj, může se stát, že bude v průběhu implementace potřeba něco změnit nebo upravit, co v návrhu nebylo předvídatelné. Implementace proto vyžaduje flexibilitu a schopnost přizpůsobit se novým požadavkům, které se mohou v průběhu vývoje objevit.

3.1 Android aplikace

Frontend projektu je implementován jako Android aplikace. Tato aplikace se skládá z jednotlivých vrstev, jež jsou popsány v kapitole 2.3.1. Podle tohoto návrhu budu implementovat v aplikaci jednotlivé komponenty.

3.1.1 Základní nastavení projektu

Na začátku implementace je nezbytné inicializovat projekt a přidat potřebné závislosti různých knihoven, které budou v aplikaci použity. Dále je třeba vhodně

rozvrhnout strukturu projektu a propojit Android projekt s Firebase a Stream projekty.

3.1.1.1 Struktura projektu

V aplikaci využívám principy Clean Architecture a dodržuji konvence, které jsou osvědčené v praxi a používají se ve firmě, kde pracuji. Tyto konvence a principy by měly dělat projekt přehlednějším a zlepšovat testovatelnost jednotlivých komponent. I když neexistují jednoznačná a univerzální pravidla pro implementaci principů Clean Architecture v Android vývoji, tyto konvence dodržují určité zásady, které se osvědčily na reálných projektech. Proto se řídím pravidly, která jsem si osvojila v praxi při vývoji Android aplikací.

Moduly

Projekt dělím do jednotlivých modulů, přičemž moduly spadají do jedné ze tří kategorií:

- **App module**

  Hlavní modul aplikace, ve kterém se inicializuje aplikace. Tento modul má závislosti na všechny ostatní moduly. Obsahuje např. App a Main Activity třídy, které jsou hlavním vstupním bodem aplikace. Dále obsahuje navigační komponenty aplikace a Service zprostředkovávající příjem notifikací.

- **Features**

  Feature moduly představují vždy jednu komplexní obrazovku aplikace. Např. jedna feature (neboli funkcionalita) je uživatelský profil, jiná feature je zobrazení přehledu předmětů na výměnu, další je obrazovka přidání nového předmětu atd. Feature modul může být závislý na libovolném množství Library modulů, neměl by však být závislý na App modulu či jiných Feature modulech.

- **Libraries**

  Library moduly představují např. části aplikace, které komunikují s backendem a jinými perzistentními datovými vrstvami. Dále se zde nachází moduly, které definují design celé aplikace nebo obsahují přepoužitelné UI komponenty. Library moduly by neměly být závislé na žádných jiných modulech.

Package

Jednotlivé komponenty v rámci těchto modulů dále dělím do packagů (balíčků). Každý modul může obsahovat jejich libovolnou podmnožinu, závisle na funkcionalitě modulu. Těmito package jsou:

- **System**

  System package obsahuje všechny komponenty implementující UI – Activity, Fragmenty a Composables, dále Services či App. Komponenty v system packagi jsou závislé na komponentách v presentation packagích.

- **Presentation**

  Obsahuje View Modely, View State data classes a jejich mapování z doménových datových tříd. Komponenty z tohoto package jsou závislé na domain package komponentách.

- **Domain**

  Obsahuje Use Casy a repository interfaces. Jeden Use Case představuje nějakou konkrétní funkcionalitu jako např. načtení uživatelových dat. V Use Casech se implementuje jakákoliv složitější logika nad daty získanými z datové vrstvy.

- **Model**

  Datové třídy doménových objektů. Nemají závislost na žádných jiných komponentách, představují základní entity aplikace jako např. uživatel, předmět, hodnocení apod.

- **Data**

  Implementace repository interfaců, které komunikují s backendem a ostatními zdroji dat. Dále package obsahuje mappery datových entit na doménové. Závislosti má pouze na model package.

- **Tools**

  Tools package obsahuje všechny komponenty infrastruktury, např. nástroje na parsování objektů, funkce na formátování dat apod.

- **Di**

  Obsahuje komponenty implementující dependency injection v projektu, které je popsáno v sekci 3.1.5.

3.1.1.2 Propojení s Firebase

Aby bylo možné využívat služby Firebase v Android projektu, bylo zapotřebí vytvořit Firebase projekt. Ten se zakládá na Firebase portálu a je propojený s Google účtem. Přes projekt, který jsem vytvořila, integruji do aplikace jednotlivé služby jako je Firebase Authentication nebo Firestore. Každou aplikaci, která by měla využívat služeb konkrétního Firebase projektu, je potřeba registrovat. Já jsem registrovala svou Android aplikaci. Při registraci Android aplikace jsem zadala jméno aplikace, název package, případně je možné přidat další konfigurace podle potřeby. V dalším kroku jsem stáhla google-services.json soubor, který je potřeba

vložit do projektu. Tento soubor specifikuje konfiguraci Firebase projektu propojeného s aplikací a obsahuje identifikační údaje projektu i klienta. Je proto vhodné nepushovat tento soubor na git, jelikož obsahuje soukromé API klíče, které by mohly být zneužity. V dalším kroku jsem nastavila potřebný google-services plugin v build.gradle projektu a také v build.gradle všech modulů, kde budou služby využívány. Poté jsem již přidala jen potřebné závislosti konkrétních služeb v modulech, kde jsou využívány.

3.1.2 Autentizace uživatelů

Případy užití aplikace vyžadují, aby bylo možné identifikovat jednotlivé uživatele, ukládat jejich data a omezit přístup k datům na základě autorizace. Ve své aplikaci využívám k autentizaci uživatelů služby Firebase Authentication, jak jsem určila při výběru technologií v kapitole návrhu 2.2.2.1.

Integrace autentizace do aplikace vyžaduje minimální počet kroků. Po přidání závislosti Firebase Auth SDK se dá v aplikaci přistoupit k instanci `Firebase.auth`, která zprostředkovává přístup ke všem funkcím autentizace pomocí Firebase. Ve své aplikaci využívám registrace a přihlášení uživatele pomocí emailu a hesla. Pro implementaci tohoto způsobu autentizace je zapotřebí pouze dvou funkcí:

- `FirebaseAuth.signInWithEmailAndPassword(email, password)`
- `FirebaseAuth.createUserWithEmailAndPassword(email, password)`

Komunikace pro přihlášení probíhá přímo mezi Android klientem a Authentication službou. Registraci implementuji přes Firebase Cloud Functions, jelikož je zapotřebí první zkontrolovat, zda-li uživatelské jméno, které uživatel zadal, není již zabrané. Pokud je vše v pořádku, uživatel je poté autentizován a je vytvořen jeho záznam ve Firestore databázi.

Po úspěšné registraci či přihlášení je vydán uživatelskému zařízení token, který je v zařízení uložen a používá se k autentizaci při dalších požadavcích na Firebase služby jako je Cloud Storage nebo Firestore. Tímto způsobem je umožněno, že uživatel zůstává přihlášený i po restartu aplikace. Všechny tokeny mají expirační dobu, proto v navigační komponentě aplikace kontroluji aktuální stav přihlášení uživatele a pokud přihlašovací token expiroval či uživatel si ještě nevytvořil uživatelský účet, je přesměrován na login obrazovku.

3.1.3 Jetpack Compose

V aplikaci využívám nový přístup k vytváření UI komponent pro Android pomocí Jetpack Compose. Oproti XML Views, které se v Android vývoji využívaly dříve, je Compose více intuitivní, vyžaduje méně kódu a jednotlivé komponenty jsou snadno přepoužitelné. Na rozdíl od deklarativního přístupu u XML views, kde bylo nutné jednotlivé komponenty nejdříve navázat na Activity nebo Fragment a pak na

základě obsahu měnit jejich stav, Compose využívá imperativního přístupu, který předem definuje vzhled obrazovky, který je poté měněn na základě aktuálního stavu [24]. Compose framework využívá inteligentního přístupu k rekompozici komponent, překresleny jsou vždy pouze komponenty, jejichž stav byl změněn, ostatní prvky na obrazovce se nepřekreslují. Díky tomu například nepředstavuje problém vkládání více layoutů do sebe, jelikož se vždy překreslují pouze prvky, jejichž stav byl změněn [25].

3.1.3.1 Navigace

Ve své aplikaci definuji jednotlivé obrazovky jako samostatné Composables funkce, které jsou vytvářeny v navigační komponentě. Navigační komponenta je hlavním UI prvkem aplikace, která je inicializována při vytvoření Main Activity. Navigační komponenta obsahuje `NavHost` Composable, v rámci které je vytvořen navigační graf a jsou v ní inicializovány všechny obrazovky aplikace. Je provázána s `TopBar`, který obsahuje informace o aktuálně zobrazené obrazovce a doplňkové akce (např. navigace zpět apod.), a `BottomBar`, který poskytuje UI pro navigaci mezi hlavními obrazovkami.

Ze strukturálního hlediska aplikace obsahuje dvě odlišné lokace – login obrazovku aplikace, kde se může uživatel přihlásit nebo registrovat, a poté hlavní obsah aplikace se všemi funkcionalitami, které jsou dostupné pouze autentizovaným uživatelům (např. zobrazení profilu, vyhledávání předmětů či přehled zpráv). Z logického pohledu a přehlednosti kódu by bylo vhodné tyto dvě lokace oddělit, avšak ve své implementaci jsem login obrazovku ponechala ve stejné navigační komponentě jako ostatní obrazovky. V původním řešení jsem měla vnořené dvě navigační komponenty v sobě, přičemž první obsahovala login obrazovku a poté druhou navigační komponentu se zbytkem obrazovek, kam byl uživatel přesměrován po přihlášení či registraci. Avšak pokud se uživatel odhlásil a měl být předsměrován zpět na login obrazovku, docházelo na pozadí stále k překreslování hlavního obsahu vnořené navigační komponenty, proto jsem zvolila následně řešení s ponecháním login obrazovky ve stejné navigační komponentě.

3.1.3.2 Composable obrazovky

Všechny Composables obsažené v navigační komponentě jsou jednotlivé obrazovky aplikace. Navigační komponenta obsahuje hlavní obrazovky, na které se dá navigovat z menu a poté sekundární obrazovky, do kterých se naviguje z hlavních obrazovek s případnými navigačními argumenty, jako jsou ID uživatelů či předmětů získaných z databáze. Hlavními obrazovkami jsou:

- Uživatelský profil se stručným přehledem uživatelových informací, seznamem uživatelových předmětů a seznamem oblíbených předmětů. Z této obrazovky se dá navigovat na detail profilu, zobrazující přehled dalších uživatelových dat

(popis a hodnocení uživatelů) nebo do nastavení profilu, kde si může uživatel změnit profilový obrázek nebo svůj popis, případně se odhlásit z aplikace.

- Přehled nabízených předmětů na výměnu ostatních uživatelů. V seznamu předmětů může uživatel vyhledávat nebo si může rozkliknout detail předmětu, z kterého je poté možné poslat uživateli zprávu ohledně předmětu, rozkliknout si profil majitele předmětu nebo si uložit předmět do oblíbených.
- Obrazovka přidání nového předmětu s nahráním fotek předmětu a vyplněním informací o předmětu.
- Přehled uživatelových konverzací s ostatními uživateli o konkrétních předmětech na výměnu.

Každá tato obrazovka má podobnou základní strukturu, ve které se pozoruje stav získaný z View Modelu a na jeho základě se vykreslí obsah obrazovky. Stav může nabývat hodnot Init, Loading, Empty nebo Fail State, pro něž jsem nadefinovala komponenty, které jsou sdílené napříč aplikací mezi všemi obrazovkami, jelikož tyto stavy se projevují skrz aplikaci stejně či velice podobně. Posledním typem stavu je stav úspěšného načtení dat konkrétní komponenty a pro ten se každá obrazovka liší na základě její dané funkcionality. Úryvek kódu 1 ukazuje právě jednu definici takovéto Composable funkce vykreslující UI. Na obrázcích 3.1 lze vidět screenshoty obrazovek profilu a přehledu předmětů tak, jak vypadají v prototypu aplikace. Na obrázcích 3.2 jsou potom zobrazeny obrazovky detailu předmětu a přidání nového předmětu.

3.1.4 Komunikace mezi vrstvami aplikace

Ke komunikaci mezi view vrstvou aplikace (tedy composable funkcemi) a ViewModelem využívám StateFlow, který umožňuje implementaci publisher-subscriber principu. Ve ViewModelu inicializuji MutableStateFlow, jenž aktualizuji na základě dat příchozích z datové vrstvy. Composables funkce vykreslující UI pozorují získávají při každé změně Flow aktuální data a na základě toho překreslují UI.

Komunikaci ViewModelů s datovou vrstvou probíhá poté přes Use Cases, jak jsem definovala v kapitole návrhu architektury Android klienta 2.3.1. Každý Use Case představuje nějakou jednotnou funkcionalitu jako je např. získání všech předmětů přidaných uživatelem do oblíbených. Na základě potřeby obsahují Use Cases reference na rozhraní repository, jejichž implementace poté komunikuje přímo s API backendových služeb jako je práve Firestore nebo Firebase Cloud Functions či Cloud Storage.

3.1.5 Dependency injection

Dependency injection (DI) je softwarový návrhový vzor, který slouží ke snížení vzájemného propojení mezi komponentami v softwarovém systému a zvyšuje mo-

```
@Composable
fun ItemDetailScreen(
    itemId: String,
    viewModel: ItemDetailViewModel,
    onScreenInit: (ScreenState) -> Unit,
    onNavigateBack: () -> Unit,
    navigateToProfileDetail: (String) -> Unit,
    navigateToChat: (String) -> Unit
) {
    val itemDetailState by viewModel.itemDetailState.collectAsState()

    TopBar(onScreenInit, onNavigateBack,)
    Box {
        ResolveState(
            itemDetailState,
            navigateToProfileDetail,
            navigateToChat
        )
    }
}

@Composable
fun ResolveState(
    state: ItemDetailState,
    navigateToProfileDetail: (String) -> Unit,
    navigateToChat: (String) -> Unit
) {
    when (state) {
        is Loading -> LoadingView(semiTransparentBlack)
        is Success -> ItemDetail(
                            state,
                            navigateToOwnerProfileDetail
                        )
        is Failure -> FailureView(state.message)
        is CreateChannelSuccess -> navigateToChat(state.channelId)
        is CreateChannelFailure -> FailSnackMessage(state.message)
        else -> Unit
    }
}
```

Výpis kódu 1 Ukázka kódu Composable funkce zobrazující obrazovku detailu předmětu.

dularitu, testovatelnost a znovupoužitelnost kódu. Při použití DI jsou závislosti komponent odděleny od samotné implementace těchto komponent. To znamená, že namísto toho, aby komponenta sama vytvářela a spravovala své závislosti, jsou jí tyto závislosti poskytnuty ze zdroje mimo samotnou komponentu. Díky DI může být kód snáze testovatelný, protože je možné snadno nahradit reálné závislosti komponent mock objekty nebo jinými implementacemi pro účely testování. DI také zlepšuje znovupoužitelnost kódu, protože závislosti jsou odděleny a mohou být snadno nahrazeny jinými implementacemi.

DI v projektu implementuji pomocí knihovny Koin. Využívám ji k injektování ViewModelů do Composable funkcí obrazovek, sdílení singleton instancí repository rozhraní a jejich implementací a definování factory pro UseCase komponenty. Každý modul (všechny Feature i Library moduly) definují svůj Koin modul, který je pak inicializován v App modulu a přepoužit v celé aplikaci.

3.1.6 Zasílání zpráv mezi uživateli

Aby byla aplikace pro uživatele opravdu využitelná, bylo zapotřebí implementovat možnost zasílání zpráv mezi uživateli ohledně předmětů, o které mají zájem. Tuto funkcionalitu jsem implementovala pomocí Stream Chat SDK. Toto SDK nabízí rychlé a intuitivní řešení, jak implementovat chatovací funkci do aplikací.

Po založení a nastavení parametrů Stream projektu v jejich konzoli a integraci do projektu Android klienta je zapotřebí vytvořit instanci Chat clienta s potřebnou konfigurací. Tuto instanci inicializuji jako singleton v hlavním App modulu, která se pak využívá na dalších místech projektu. Implementace jednotlivých obrazovek je poté zjednodušena díky předpřipraveným komponentám.

3.2 Cloud Functions

Pro implementaci komplexnějších funkcí v mé aplikaci, které vyžadovaly složitější proces než pouhé volání API a jednoduchou filtraci dat, jsem využila Firebase Cloud Functions. Cloud Functions jsou serverless funkce napsané v typescriptu, které jsou nasazeny na Firebase servery a umožňují implementaci backendových funkcionalit bez nutnosti správy fyzických serverů a škálování jejich prostředků. V mém případě jsem cloud funkce využila jen v omezené míře, protože Firebase nabízí jednoduché a intuitivní rozhraní pro přímou komunikaci mezi Android klientem a Firebase službami. Tyto funkce jsem využila pouze pro konkrétní případy, kterými jsou:

3.2.1 Registrace uživatele

Registraci uživatele implementuji jako cloud funkci, jelikož jsem chtěla klientskou aplikaci odštítit od potřeby řešit jednotlivé kroky celého procesu. Nejprve proběhne

kontrola, zda-li požadované uživatelské jméno není již obsazené. Poté probíhá autentizace pomocí Firebase Auth a až poté je vytvořen záznam uživatele v Users kolekci ve Firestore. Tento záznam obsahuje unikátní ID, které je uživateli přiděleno během autentizace. Toto ID je pak v klientské aplikaci dostupné pomocí instance Firebase Auth a umožňuje získání dalších uživatelových dat uložených v databázi. Díky tomuto postupu zaručuji, že každý uživatel má přístup pouze k vlastním datům a je ochráněn před neoprávněným přístupem ostatních uživatelů. Ukázka kódu je k vidění v 2.

3.2.2 Vytvoření předmětu

Další funkcionalitou je vytvoření záznamu o předmětu. Nejprve je tento záznam vytvořen v databázi a poté je jeho ID uloženo u záznamu uživatele, který předmět nahrál. Výsledek této funkce je poté předán zpět do Android klienta, skrze který probíhá nahrávání obrázků předmětu do Cloud Storage. Je totiž nezbytné uložit download URI obrázků k záznamu předmětu a to není možné získat přes cloud funkce. Po dokončení nahrávání obrázků je pak aktualizován záznam předmětu s odpovídajícími download URI jeho fotek. Tento postup umožňuje snadné získání informací o předmětu, včetně jeho obrázků, jelikož download URI jsou uloženy přímo u záznamu předmětu a není tedy nutné je stahovat zvlášť z Cloud Storage při získávání dat předmětu, což je časově náročnější než query nad Firestorem.

3.2.3 Notifikace přidání předmětu do oblíbených

Cloud funkce lze také využít pro reakci na změny ve Firestore databázi. V mém případě je využívám k detekci změn stavu oblíbenosti předmětu. Při přidání předmětu do oblíbených se ID předmětu přidá do pole „likedItems“ u daného uživatele v databázi. Moje cloud funkce tedy monitorují změny v tomto poli u všech uživatelů a v případě, že dojde ke změně, získají příslušná data o předmětu a o jeho majiteli. Na základě těchto dat je pomocí Cloud Messaging odeslána notifikace majiteli předmětu o tom, že jeho předmět byl přidán do oblíbených jiným uživatelem. Tento postup umožňuje efektivní a přesnou reakci na změny v databázi a zajišťuje tak rychlou a spolehlivou komunikaci mezi uživateli.

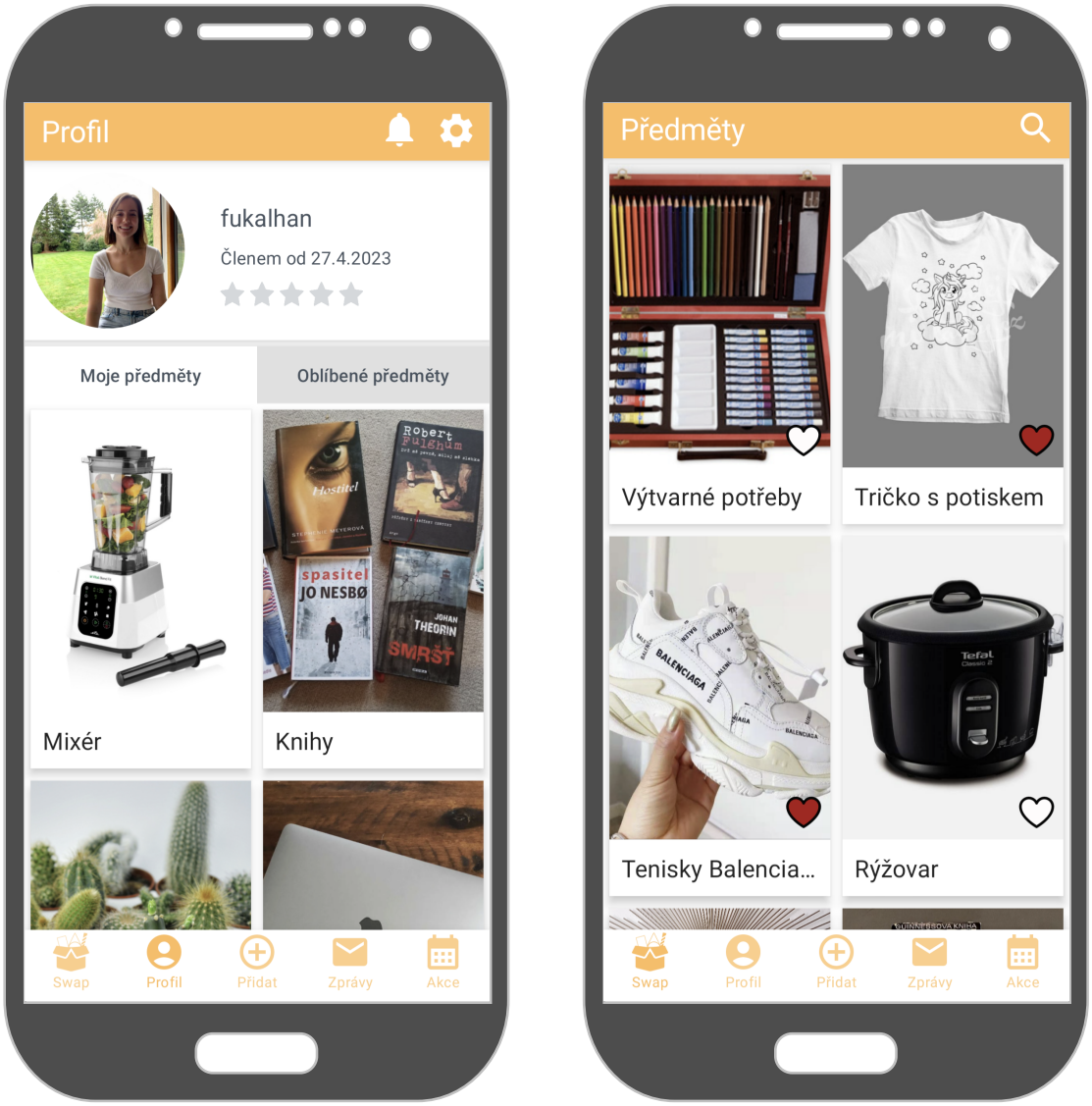
3.2.4 Rozšíření pro Firebase Auth a Stream

Jako poslední funkcionalitu využívám cloud funkce k reakci na události spojené se Stream Chatem, který používám pro implementaci chatovací funkcionality mezi uživateli. Pomocí tohoto SDK je snadné integrovat Firebase Authentication s autentizací ve Stream Chatu. Tyto externí funkce reagují na události vytvoření a smazání záznamu uživatele v Firebase Auth a na základě těchto událostí vytvářejí

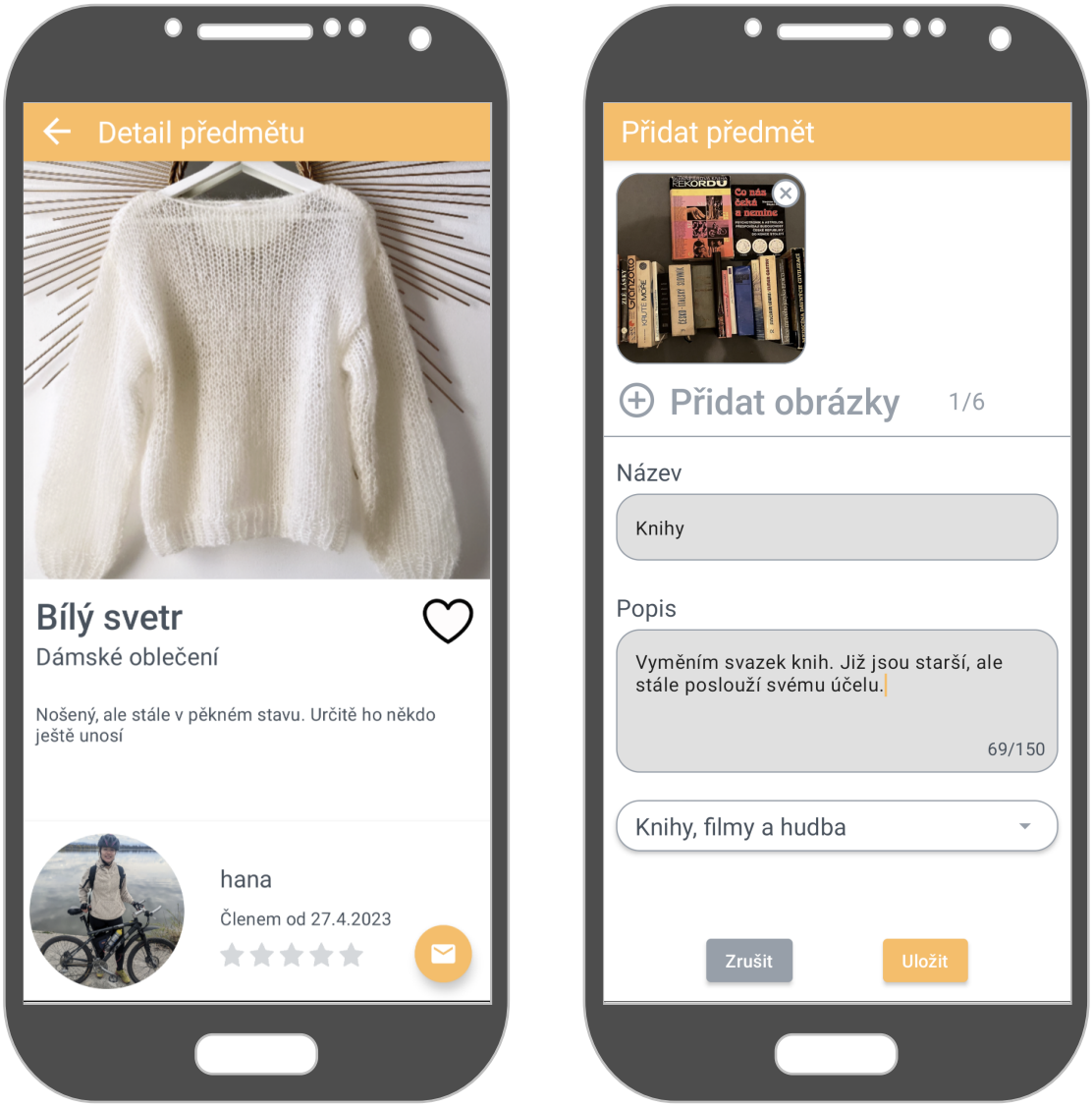
nebo odstraňují uživatelský účet ve Stream Chatu. Tato funkcionalita zajišťuje jednoduchou a spolehlivou synchronizaci mezi Firebase Auth a Stream Chatem.

```
export const createUser = functions.https.onCall(async (data) => {
  const { username, email, password } = data;
  try {
    // Check if the username is already taken
    const snapshot = await admin.firestore().collection("users")
      .where("username", "==", username)
      .get();

    if (!snapshot.empty) {
      // Username is already taken
      return {result: "USERNAME_TAKEN"};
    } else {
      // Create a new user account with Firebase Authentication
      const userCredential = await admin.auth().createUser({
        email: email,
        password: password,
      });

      // Create a new user record in the Users collection
      const newUser = {
        id: userCredential.uid,
        username: username,
        profilePic: defaultProfilePic,
        email: email,
        joinDate: userCredential.metadata.creationTime,
      };

      const userRef = admin
        .firestore()
        .collection("users")
        .doc(userCredential.uid);

      await userRef.set(newUser);
      return {result: "SUCCESS"};
    }
  } catch (error: any) {
    ...
  }
});
```

Výpis kódu 2 Ukázka kódu deploynuté serverless funkce `createUser`, která slouží k autentizaci uživatele a vytvoření uživatelského záznamu.



Obrázek 3.1 Obrazovky uživatelova profilu a přehledu předmětů v prototypu aplikace.



Obrázek 3.2 Obrazovky detailu předmětu a přidání nového předmětu v prototypu aplikace.

Kapitola 4
Testování

V této kapitole se věnuji testování finálního prototypu aplikace, kdy se zaměřuji na uživatelské testování s pěti vybranými uživateli, kteří představují skupinu potenciálních uživatelů výsledného produktu. Společně s nimi procházím testovací scénáře a na závěr analyzuji výsledky testování a navrhuji možná vylepšení budoucích verzí aplikace.

Testování aplikací představuje zásadní část procesu softwarového vývoje, kdy je ověřována kvalita a správnost funkcionality výsledného produktu. Cílem testování je odhalit chyby, nedostatky a potenciální problémy aplikace, které by mohly vést k neuspokojivému uživatelskému zážitku a snížení efektivity produktu. Testování zahrnuje různé typy testů, jako jsou testy funkcionality, výkonu, bezpečnosti, uživatelského rozhraní a další. Jeho účelem je zaručit, že aplikace bude plně funkční, spolehlivá a bezpečná pro uživatele, a to především v reálných podmínkách. Správně provedené testování aplikace přispívá ke zvýšení kvality výsledného produktu, minimalizaci rizik a nákladů spojených s opravou chyb a zvýšení spokojenosti uživatelů. Ve své práci se zaměřuji na uživatelské testování aplikace.

4.1 Uživatelské testování

Uživatelské testování aplikace představuje důležitou fázi v procesu vývoje softwaru, která umožňuje vývojářům získat cennou zpětnou vazbu od potenciálních uživatelů produktu. Prostřednictvím této metody testování mohou vývojáři zjistit, jak uživatelé produkt používají, s čím mají největší potíže, co je pro ně příliš složité či nepřehledné a jak lze případně produkt vylepšit. Toto testování přináší řadu výhod, jelikož je aplikace otestována přímo koncovými uživateli produktu, kteří mohou pomoci například s odhalením problémů s uživatelským rozhraním, jako jsou matoucí popisky, obtížně ovladatelná tlačítka, navigace a další prvky, které uživatelům komplikují práci s produktem. V rámci tohoto testování prochází jednotliví testující uživatelé předpřipravenými scénáři případů užití aplikace.

4.1.1 Testovací scénáře

Testovací scénáře jsem volila tak, aby si testující uživatelé prošli všechny hlavní funkcionality aplikace. Zvolila jsem 8 scénářu, které jsou popsané ve výčtu níže.

1. Zaregistrujte se do aplikace. Změňte si profilový obrázek a popis profilu.
2. Přidejte jeden svůj předmět na výměnu. Můžete přidat předmět s libovolnými údaji.
3. Chcete najít mixér na výměnu, ale ještě nejste rozhodnuti, zda-li chcete tyčový mixér nebo normální (plnění z vrchu). Vyhledejte tedy všechny možnosti a všechny výsledky si přidejte do oblíbených.
4. Rozhodli jste se pro jeden z mixérů, které jste si uložili do oblíbených. Pošlete tedy zprávu jeho majiteli o tom, že byste měli o předmět zájem.
5. Odhlaste se z aplikace a znovu se přihlašte s předem určenými uživatelskými údaji. Zkontrolujte zprávy a odepište případným zájemcům o některý z vašich předmětů.
6. Vyberte si jednu konverzaci ohledně vámi nabízeného předmětu a projděte celým procesem výměny až k finálnímu udělení hodnocení uživateli (v tomto kroku předstíráme pro účel testování, že došlo k domluvě na fyzickém setkání a proběhlo předání předmětů v mezičase).
7. Chtěli byste se zúčastnit nějaké veřejné swapovací akce. Podíváte se tedy na přehled nabízených událostí a přihlásíte se na libovolnou z nabízených.
8. Chtěli byste si založit vlastní swapovací akci zaměřenou na výměnu pokémonových kartiček. Přidáte tedy událost s vhodně zvoleným názvem a popisem. Uspořádat byste ji chtěli příští víkend někde v Praze.

4.1.2 Testující uživatelé

Celkem jsem aplikaci otestovala s 5 uživateli. Snažila jsem se volit uživatele s různou mírou zkušeností s jakoukoliv formou swapu.

- **Uživatel 1**

  Muž, 25 let, žádné předchozí zkušenosti se swapem. Povoláním programátor a zároveň student IT.

- **Uživatel 2**

  Žena, 52 let, má předchozí zkušenosti s výměnou předmětu mezi přáteli. Povoláním programátorka.

- **Uživatel 3**

  Žena, 27 let, má zkušenost s výměnou oblečení přes platformu Vinted. Povoláním programátorka.

- **Uživatel 4**

  Muž, 25 let, Žádné předchozí zkušenosti se swapem. Povoláním frontend vývojář.

- **Uživatel 5**

  Muž, 25 let, s výměnou věcí má minimální zkušenosti. Povoláním backend developer.

4.1.3 Průběh testování

Testování jsem prováděla buď ve fyzické přítomnosti uživatelů nebo testování probíhalo přes video hovor. Na začátku jsem uživateli stručně představila doménu aplikace a jaký je její účel a poté jsme přešli na vykonávání jednotlivých scénářů. Uživatelé vykonávali scénáře samostatně, pouze v případě, že se v aplikaci vyskytla nějaká chyba nebo si nevěděli dlouho rady, jsem do průběhu testování zasáhla a poskytla jim nápovědu nebo vysvětlení. Na závěr jsem s nimi provedla rozhovor o jejich celkových dojmech z aplikace, o tom, co bylo problémové a kde vidí případné nedostatky. Průběh každého testování je nahrán jako záznam obrazovky zařízení s audiem komentáře testu. Tyto záznamy jsou dostupné v příloze práce.

Průběh testování s jednotlivými testovacími uživateli popisuji v následujících sekcích, příčemž podávám u toho výčet chyb a nedostatků aplikace, které byly testováním odhaleny. Každému nalezenému problému uděluji hodnocení od 1 do 5, které definuje závažnost problému následujícím způsobem:

- 1 – Designový nedostatek aplikace, neovlivňuje nijak funkčnost aplikace, uživatelé se mohou nad ním však pozastavit.
- 2 – Nedostatek aplikace, který může způsobovat delší čas vykonávání jednotlivých případů užití, než by bylo očekávané, avšak nijak nenarušuje funkcionalitu aplikace.
- 3 – Problém s tímto stupněm hodnocení způsobuje uživateli nepříjemný zážitek, ale nijak neovlivňuje celkovou funkčnost aplikace.
- 4 – S chybou tohoto stupně je aplikace stále částečně použitelná, avšak uživatel musí hledat složitá řešení pro svoje potřeby a uživatelský zážitek je nepříjemný.
- 5 – Závažná chyba v aplikaci, bez jejíhož řešení není aplikace plně funkční.

Po hodnocení problému následně navrhuji možné řešení daného nedostatku, které by bylo možné implementovat v budoucích verzích aplikace.

4.1.3.1 Testování s uživatelem 1

S uživatelem 1 jsme prošli všechny testovací scénáře. Průběh testování byl poměrně hladký, uživatel komentoval své kroky při průchodu aplikací a ve většině případech mu bylo hned jasné, kde danou funkcionalitu hledat.

Při testování jsme nalezli následující nedostatky:

- Při nastavení profilového obrázku a popisu profilu uživatel nejprve zkoušel kliknout na profilový obrázek na hlavní stránce profilu a poté přešel ještě na stránku detailu profilu a až poté nalezl nastavení, kde údaje zdárně změnil. Jak vyplynulo z následné konverzace o tomto zádrheli, problémem bylo, že aplikace, kterou používal k nahrávání obrazovky, mu zrovna částečně překrývala ikonu nastavení v pravém horním rohu aplikace, jinak říkal, že by si nejspíš tlačítka všiml dříve.

  Závažnost: 2

  Řešení: Je možné změnit funkcionalitu v profilu a umožnit uživateli měnit tyto údaje přímo z hlavní stránky profilu nebo jeho detailu při kliknutí na profilový obrázek a popis.

- Uživatel byl lehce zmaten při úkolu, kdy se měl přihlásit na libovoulnou akci z nabízených veřejných událostí, ale nakonec se zdárně na akci přihlásil. Problém byl v tom, že používal již předpřipravený účet k účelům testování, který byl již na jednu z událostí přihlášen, a vybral si náhodně zrovna tuto. Uživatel na základě toho nabyl dojmu, že se mu zobrazují jenom události, na které se již přihlásil a proto chvíli v aplikaci hledal, kde se dá přihlásit na nové události. Po vysvětlení, že si zrovna vybral jednu událost, kde již byl přihlášen, dořešil úlohu rychle.

  Závažnost: 3

  Řešení: Aplikace by měla umožňovat uživatelům filtrovat mezi událostmi a zobrazit si pouze události, na které je nebo není uživatel již přihlášen.

4.1.3.2 Testování s uživatelem 2

Testování s uživatelem 2 objevilo více nedostatků aplikace. Kromě závažnějších chyb, které popisuji níže, byl celkový průběh testování pomalejší, jelikož uživatel chtěl volit popisy předmětů a všechny informace pravdivě a gramaticky správně a proto vyplňování určitých informací zabralo delší dobu, což se ale týkalo spíše celkové schopnosti ovládat moderní smartphony.

Nedostatky, které byly nalezeny, jsou:

- Při změně profilového obrázku uživatel našel záložku nastavení ihned, avšak chvíli váhal, zdali změna profilového obrázku probíhá kliknutím na něj.

  Závažnost: 2

Řešení: Uživatel sám navrhl, že by mu přišlo vhodné, aby u obrázku byla přítomna určitá ikona, která by indikovala, že kliknutím na obrázek je možné jej změnit.

- Vyhledávání předmětů proběhlo vcelku hladce, avšak při přidání výsledku vyhledávání do oblíbených uživatel odhalil závažnou chybu, kdy přidání předmětu do oblíbených zobrazí opět seznam všech předmětů a ne pouze výsledky vyhledávání.

  Závažnost: 5

  Řešení: Je potřeba implementovat funkcionalitu tak, aby se po přidání předmětů do oblíbených z výsledků vyhledávání neaktualizoval celý seznam, ale zobrazily se pouze výsledky vyhledávání.

- U vyhledávání předmětů byl uživatel zmaten tím, že je vyhledávání citlivé na diakritiku. Toto zjevně není u aplikací žádané, jelikož uživatelé očekávají, že při psaní bez diakritiky se zobrazí i předměty pojmenované s diakritikou.

  Závažnost: 4

  Řešení: Implementace vyhledávání bez citlivosti na diakritiku.

- U vyplňování textových údajů je softwarová klávesnice zarovnaná těsně pod textová pole, takže vyplňovaný text lze vidět, ale nejde vidět obsah obrazovky pod ním. Uživatel byl chvíli zmaten, jak schovat klávesnici po dopsání textu, což je sice spíše o uživatelově schopnosti používat smartphony, ale bylo by vhodné udělat rozhraní více přívětivé.

  Závažnost: 3

  Řešení: Implementace zarovnání softwarové klávesnice pod obrazovku.

- Uživatel byl lehce zmaten z ikon tlačítek v navigačním menu.

  Závažnost: 1

  Řešení: Přidání textového popisu ke každému tlačítku menu.

4.1.3.3 Testování s uživatelem 3

Uživatel prošel všemi testy, ačkoliv v průběhu testování byla nalezena závažná chyba, kterou jsem ihned opravila, aby bylo možné pokračovat v průběhu testu. Uživatel procházel aplikací poměrně rychle až na pár problémů, které jsou vypsány níže.

- Uživateli se nezobrazily jeho konverzace.

  Závažnost: 5

  Řešení: Příčinu závady v zobrazení konverzací jsem ihned odhalila a opravila. Docházelo k nesprávnému filtrování zpráv, které by se měly uživateli zobrazit.

- Uživatel měl problém s nalezením nastavení profilového obrázku a popisu profilu. První se snažil změnit obrázek kliknutím na něj na hlavní stránce profilu a potom i v detailu profilu. Po těchto dvou neúspěšných pokusech potom vyzkoušel nastavení, kde obrázek změnil.

  Závažnost: 2

  Řešení: Přesunout změnu profilových údajů přímo na hlavní stránku profilu nebo do detailu profilu.

- Při změně popisu profilu přišlo uživateli odhlašovací tlačítko špatně umístěné. Po editaci popisu uživatele aplikace vybízela spíše ke kliknutí na tlačítko odhlášení než na tlačítko uložení změn.

  Závažnost: 3

  Řešení: Přesunutí tlačítka odhlášení na vhodnější pozici.

- Uživateli přišly barvy spodního menu málo kontrastní, což znesnadňovalo určení, na které obrazovce z menu se uživatel právě nachází.

  Závažnost: 1

  Řešení: Zvolit více kontrastní barvy ikon spodního menu, aby bylo jasně rozlišitelné, která položka je aktuálně zvolena.

4.1.3.4 Testování s uživatelem 4

Testování s uživatelem 4 proběhlo hladce. Uživatel měl lehké problémy pouze u prvního testovacího scénáře a dále v závěrečném rozhovoru diskutoval vhodnější zvolení umístění jedné z komponent v aplikaci. Následuje výčet nedostatků nalezených při testování.

- Uživatel se snažil u úkolu se změněním profilového obrázku a popisu nejprve změnit obrázek kliknutím na profil a následně v detailu profilu. Po nenalezení dané funkcionality v těchto obrazovkách přešel do nastavení, kde údaje zdárně změnil.

  Závažnost: 2

  Řešení: Tento problém se vyskytl také u většiny z předchozích testerů. Bylo by nejspíše vhodné přemístit možnost změny profilových údajů přímo do záložky profilu.

- Uživatel v závěrečném rozhovoru podotkl, že ačkoliv neměl s nalezením záložky oblíbených předmětů žádný problém, očekával by ji nejspíš někde jinde, např. na domovské obrazovce spolu s přehledem dalších informací podstatných pro uživatele.

  Závažnost: 1

Řešení: Je možné zvolit jiné rozložení obrazovek, například místo hlavní obrazovky profilu vytvořit domovskou obrazovku s přehledem aktualit v aplikaci a profil dát do jiné záložky menu.

4.1.3.5 Testování s uživatelem 5

Průběh testovacími scénáři s uživatelem 5 proběhl vcelku hladce bez větších obtíží. Vyskytl se pouze jeden problém, o kterém se uživatel zmínil v závěrečném rozhovoru, který je popsaný níže.

- Při výběru místa konání u založení události bylo pro uživatele zprvu matoucí, že musí scrollovat níže, aby se zobrazila nápověda možných adres, po chvíli ale na to přišel.

  Závažnost: 2

  Řešení: Upravit vhodně rozvržení obrazovky tak, aby list nápovědy možných adres nebyl překrytý zobrazenou softwarovou klávesnicí.

4.1.4 Vyhodnocení testů

Uživatelské testování hodnotím celkově jako velice přínosné, jelikož mi umožnilo odhalit jak skryté chyby v aplikaci, tak i designové nedostatky, které by mohly znepříjemňovat uživateli zážitek při používání aplikace. Dané nedostatky jsem u každého testu ohodnotila, navrhla jejich řešení a případně některé chyby a problémy rovnou opravila. V následujícím výčtu popisuji problémy o jednotlivých stupních závažnosti od nejkritičtějších k nejméně závažným.

- Celkově se vyskytly dva problémy závažnosti 5, příčemž oba problémy byly skryté chyby aplikace, které jsem opravila ihned po jejich nalezení, aby bylo možné pokračovat v hladkém průběhu testování s dalšími uživateli. Jednalo se o problémy načítání seznamu předmětů poté, co uživatel vyhledal předměty a přidal si některý do oblíbených, a náhodně se projevující nefunkčnost zobrazení konverzací uživatele, která pravděpodobně vycházela z chybného filtrování konverzací na základě typu konverzace.

- Problém se stupněm hodnocení 4 jsem nalezla pouze jeden a tím byla citlivost na diakritiku při vyhledávání předmětů na výměnu podle klíčových slov. V realitě bylo žádané, aby vyhledávání nebylo citlivé na diakritiku, jelikož je většina uživatelů zvyklá psát bez diakritiky. Problém jsem opět ihned vyřešila přepsáním pravidel filtrování.

- Nedostatky s hodnocením 3 byly v aplikaci nalezeny tři, přičemž dva z těchto nedostatků jsem ihned vyřešila, oba se týkaly nevhodného zarovnání komponent na obrazovce při zobrazení softwarové klávesnici telefonu na displeji.

Třetím z nedostatků byla chybějící možnost jakkoliv filtrovat a vyhledávat ve swapovacích událostích. Tento problém by bylo možné adresovat v budoucích verzích aplikace.

- Problémů stupně 2 bylo nejvíce s počtem pět. Všechny až na jeden se týkaly stejného problému, kterým byla změna profilového obrázku. Kromě jednoho testujícího uživatele se všichni zbývající domnívali, že by měl jít profilový obrázek změnit přímo po kliknutí na obrázek v přehledu profilu nebo jeho detailu. Všichni uživatelé nakonec našli změnu profilových údajů v nastavení, avšak bylo by za vhodné zvážit do budoucna změnu rozložení komponent v obrazovkách či případné přesunutí určitých funkcionalit do jiných obrazovek. Posledním problémem s hodnocením 2, který se netýkal nastavení profilového obrázku, bylo překrytí seznamu nápovědy adres při vytváření swapovací události softwarovou klávesnicí. Obrazovka, jež tuto komponentu obsahuje, je scrollovací, uživatel tedy má možnost zobrazit si seznam i s otevřenou softwarovou klávesnicí, avšak nemusí to být pro uživatele vždy intuitivní, proto by bylo vhodné zvážit opravu v budoucích verzích.

- Nedostatky s hodnocením 1 byly tři. Dva se týkaly vzhledu spodního menu aplikace, přičemž v rámci opravy tohoto nedostatku jsem přidala popisky ikon menu, které mohly být původně pro některé uživatele nejasné. Dalším problémem bylo poté zvolení lokace přehledu oblíbených předmětů uživatele na obrazovce profilu, přičemž uživatel zmínil, že ačkoliv neměl s nalezením záložky žádný problém, očekával by tento přehled spíše na nějaké domovské obrazovce aplikace. Tento návrh stojí také za zvážení do budoucích verzí aplikace.

Celkem se při testování se všemi uživateli vyskytlo 14 problémů, přičemž některé problémy se opakovaly u většiny uživatelů. Opravila jsem celkem 7 z těchto nedostatků, přičemž jsem vyřešila všechny chyby stupně 4 a 5 a všechny problémy stupně 3 až na jeden. Řešení zbývajících problémů zakomponuji v sekci o návrhu budoucích vylepšení.

4.2 Vylepšení do budoucna

Z uživatelského testování vyplynulo hned několik nedostatků, které by bylo vhodné adresovat v budoucích verzích aplikace. Jednalo se převážně o umístění funkcionality nastavení profilových údajů a poté o možnost vyhledávání a filtrování ve swapovacích událostech. Tyto problémy jsou méně závažného charakteru, avšak jako budoucí možná vylepšení stojí určitě za zvážení.

Dalšími možnými funkcionalitami, které by bylo vhodné zvážit v dalších verzích aplikace, je implementace více druhů notifikací uživatele. V aplikaci aktuálně chodí uživateli notifikace, když si přidá jiný uživatel některý z jeho předmětů do oblíbených. Uživatel si tak může zobrazit profil druhého uživatele a vybrat si

na oplátku něco z jeho předmětů. V aplikaci bych chtěla implementovat funkcionalitu, aby při vzájemném přidání předmětů do oblíbených, kdy uživatel 1 si přidá do oblíbených předmět uživatele 2 a naopak, přišly uživatelům upozornění, že došlo k vzájemnému matchi o zájem o předměty a z notifikace by byl uživatel rovnou navigován do konverzace o dané výměně.

Další funkcionalitou, kterou bych chtěla implementovat v dalších verzí aplikace, je vytvoření skupinového chatu akce při založení nové swapovací události, kam by byli uživatelé automaticky přidání po přihlášení na akci. Funkcionalita vytvoření skupinového chatu je v prototypu aplikace již implementovaná, avšak kvůli problémům s integrací Stream Chat SDK docházelo k chybnému filtrování zobrazených konverzací, proto jsem byla nucena od této funkcionality v prototypu upustit. Do dalších verzích by bylo ovšem vhodné funkcionalitu implementovat.

Také funkcionalita vyhledávání uživatelů v aplikaci a editace a mazání uživatelových předmětů byla plánovaná, ale nakonec jsem ji z důvodu nedostatku času a problémů při vývoji v aplikaci neimplementovala. Tyto funkcionality jsou podstatné pro kvalitní uživatelský zážitek, proto jsou naplánované na implementaci v budoucí verzi aplikace.

Poslední funkcionalitou, kterou by bylo určitě vhodné implementovat v následující verzi aplikace, je lokalizace uživatele. Tuto funkcionalitu jsem původně plánovala implementovat, jelikož je podstatná pro nalezení předmětů na výměnu a událostí v okolí uživatele, kvůli problémům při implementaci jsem ji však musela z prototypu aplikace vypustit. V budoucích verzích aplikace je zajisté nutné lokalizaci uživatele implementovat.

Kapitola 5
Závěr

Cílem této práce bylo vytvořit funkční prototyp aplikace, která se zabývá některou formou činnosti na podporu udržitelnosti. V rámci této práce měla být provedena analýza konkrétní domény pomocí dotazníkového šetření a následný návrh a implementace prototypu aplikace. V závěru mělo být provedeno uživatelské testování aplikace a jeho vyhodnocení. V tomto ohledu si myslím, že moje práce splnila všechny body zadání.

Po předběžné konzultaci a průzkumu, který zmiňuji v úvodu práce, jsem se rozhodla pro implementaci aplikace na podporu činnosti swapu. V analytické části práce jsem vytvořila dotazník, který vyplnilo 150 respondentů a z něhož vyplynuly důležité poznatky pro následné určení požadavků na aplikaci a jejích případů užití. Následně jsem provedla návrh aplikace, v kterém vytvářím model domény swapu a volím vhodné technologie k implementaci prototypu aplikace. Tato část byla poté důležitým bodem pro následné zvolení architektury aplikace, návrhu databázové struktury a načrtnutí uživatelského rozhraní. Po fázi návrhu aplikace jsem se pustila do implementace prototypu, přičemž v rámci této části práce docházelo i k určitým změnám v návrhu na základě poznatků získaných při vývoji. Po dokončení prototypu jsem provedla uživatelské testování s 5 uživateli, v rámci kterého byly odhaleny určité chyby a nedostatky aplikace, z nichž polovinu jsem opravila a řešení zbylých, nekritických problémů, popisuji v sekci o budoucích vylepšeních s případnými návrhy nových funkcionalit a zlepšení v dalších verzí aplikace.

Bibliografie

1. *Vinted* [online]. [B.r.]. Dostupné také z: www.https://www.vinted.cz/.
2. *FB Marketplace* [online]. [B.r.]. Dostupné také z: https://www.facebook.com/marketplace/.
3. *SBazar* [online]. [B.r.]. Dostupné také z: https://www.sbazar.cz/.
4. *Bazoš* [online]. [B.r.]. Dostupné také z: https://www.bazos.cz/.
5. *Nevyhazujto* [online]. [B.r.]. Dostupné také z: https://www.bazos.cz/.
6. AN, Daniel. *Find out how you stack up to new industry benchmarks for mobile page speed* [online]. 2018. Dostupné také z: https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/mobile-page-speed-new-industry-benchmarks/.
7. *Percentage of mobile device website traffic worldwide from 1st quarter 2015 to 4th quarter 2022* [online]. [B.r.]. Dostupné také z: https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/.
8. WURMSER, Yoram. *The Majority of Americans' Mobile Time Spent Takes Place in Apps* [online]. [B.r.]. Dostupné také z: https://www.insiderintelligence.com/content/the-majority-of-americans-mobile-time-spent-takes-place-in-apps.
9. COUNTER, Stat. *Mobile Operating System Market Share Worldwide* [online]. [B.r.]. Dostupné také z: https://gs.statcounter.com/os-market-share/mobile/worldwide.
10. FIREBASE. *Firebase Pricing* [online]. [B.r.]. Dostupné také z: https://firebase.google.com/pricing.
11. GOOGLE. *Firebase Authentication* [online]. [B.r.]. Dostupné také z: https://firebase.google.com/docs/auth.
12. FIREBASE. *What can you do with Cloud Functions?* [Online]. [B.r.]. Dostupné také z: https://firebase.google.com/docs/functions/use-cases#notify_users_when_something_interesting_happens.

13. FIREBASE. *Choose a database: Cloud Firestore or Realtime Database* [online]. [B.r.]. Dostupné také z: https://firebase.google.com/docs/firestore/rtdb-vs-firestore.
14. FIREBASE. *Cloud Storage for Firebase* [online]. [B.r.]. Dostupné také z: https://firebase.google.com/docs/storage.
15. FIREBASE. *Firebase Cloud Messaging* [online]. [B.r.]. Dostupné také z: https://firebase.google.com/docs/cloud-messaging.
16. FIREBASE. *Google Analytics* [online]. [B.r.]. Dostupné také z: https://firebase.google.com/docs/analytics.
17. STREAM. *Design + Build: Any Chat Use Case* [online]. [B.r.]. Dostupné také z: https://getstream.io/chat/.
18. STREAM. *Stream Pricing* [online]. [B.r.]. Dostupné také z: https://getstream.io/pricing/.
19. T., Lou. *A comparison of Android Native App Architecture: MVC, MVP and MVVM*. 2016. Dostupné také z: http://example.com/paper. Dis. pr. Eindhoven University of Technology.
20. AKHTAR N., Ghafoor S. *Analysis of Architectural Patterns for Android Development*. 2021. Dostupné také z: https://www.researchgate.net/publication/352021976_Analysis_of_Architectural_Patterns_for_Android_Development. Dis. pr. Fatima Jinnah Women University.
21. GOOGLE. *ViewModel overview* [online]. [B.r.]. Dostupné také z: https://developer.android.com/topic/libraries/architecture/viewmodel.
22. MISHRA, Rishu. *MVVM (Model View ViewModel) Architecture Pattern in Android* [online]. [B.r.]. Dostupné také z: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/.
23. FIREBASE. *Cloud Firestore* [online]. [B.r.]. Dostupné také z: https://firebase.google.com/docs/firestore.
24. JALAN, Nishant Aanjaney. *4 reasons Jetpack Compose is better than XML* [online]. [B.r.]. Dostupné také z: https://medium.com/@cybercoder.naj/4-reasons-jetpack-compose-is-better-than-xml-ac0efd12db28.
25. GOOGLE. *Thinking in Compose* [online]. [B.r.]. Dostupné také z: https://developer.android.com/jetpack/compose/mental-model#recomposition.

Obsah přiložených souborů

- `user-test-recordings` ..... záznamy z uživatelského testování
- `apk` ..... adresář se spustitelnou formou implementace
- `Swap` ..... zdrojové kódy implementace klient aplikace
- `SwapBackend` ..... zdrojové kódy implementace backendových funkcí
- `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `thesis.pdf` ..... text práce ve formátu PDF