VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO NETRADIČNÍ SEZNAMOVÁNÍ LIDÍ

UNCONVENTIONAL DATING MOBILE APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAN DEMEL**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JIŘÍ HYNEK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Demel Jan**
Program: Informační technologie
Název: **Mobilní aplikace pro netradiční seznamování lidí**
Unconventional Dating Mobile Application
Kategorie: Informační systémy

Zadání:

1. Prostudujte problematiku online seznamování lidí. Proveďte průzkum a srovnání existujících aplikací a služeb.
2. Seznamte se s principem tvorby multiplatformních aplikací pro mobilní zařízení.
3. Detekujte skupinu lidí, která nepoužívá moderních aplikací a služeb pro seznamování lidí. Analyzujte důvody, proč tato skupina lidí nepreferuje tento druh seznamování lidí, analyzujte její požadavky. Korespondujte tyto požadavky s existujícími aplikacemi a službami.
4. Navrhněte řešení pro skupinu uživatelů z bodu 3.
5. Navržené řešení implementujte.
6. Proveďte uživatelské testování aplikace na vhodném vzorku lidí a vyhodnoťte výsledky. Nastiňte reálné nasazení aplikace.

Literatura:

- Johnson, J.: *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Burlington: Morgan Kaufmann Publishers/Elsevier, 2010, ISBN: 978-0-12-375030-3.
- React Native: *Documentation of React Native* [online]. 2019 [cit. 2020-09-16]. Dostupné z: https://reactnative.dev/docs/getting-started

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

Vedoucí práce: **Hynek Jiří, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 26. října 2020

Abstrakt

Cílem této práce je vytvořit novou seznamovací aplikaci, která napravuje nedostatky stávajících řešení. Jako hlavní nedostatky se ukázaly neosobní seznamování a souzení na základě vnějšího vzhledu. Ty jsou napraveny užitím geolokačních nástrojů pro vyhledání spojení v blízkém okolí, psychologického modelu MBTI a omezením času, po který si uživatelé mohou psát. V práci je postupně rozebrána problematika současných aplikací, návrh řešení dílčích problémů a poté popis implementace samotné aplikace. Ta se skládá z multiplatformní mobilní aplikace a API. Samotná multiplatformní aplikace byla implementována jako mobilní nativní aplikace použitím React Native. API bylo implementováno jako REST API. Aplikace využívá služeb Firebase pro uchovávání dat, zasílání notifikací a autentizaci uživatelů. Pro vyhledávání vhodných spojení v okolí je použito lokalizační API zařízení.

Abstract

The aim of this thesis is to create a new dating app that corrects the shortcomings of existing solutions. It turned out that the main shortcomings are impersonal dating and judging based on appearance. These are corrected by using the psychological model called MBTI and limiting the amount of time that users can chat. The thesis gradually discusses the issue of current applications, the design of solutions for partial problems and lastly the description of the implementation itself. The implementation consists of a cross-platform mobile application and an API. The cross-platform application itself was implemented as a mobile native app using React Native. The API was implemented as a REST API. The app uses Firebase services for data storage, sending notifications and user authentication. The localization API of the device is used to search for suitable connections in the vicinity.

Klíčová slova

seznamování, aplikace, React Native, MBTI, Firebase, geolokace, Android, iOS, TypeScript, Node.js, Koa

Keywords

dating, app, React Native, MBTI, Firebase, geolocation, Android, iOS, TypeScript, Node.js, Koa

Citace

DEMEL, Jan. *Aplikace pro netradiční seznamování lidí.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jiří Hynek, Ph.D.

Aplikace pro netradiční seznamování lidí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Hynka, Ph. D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Demel
10. května 2021

Poděkování

Chtěl bych poděkovat své rodině – především svým rodičům Silvii a Zdeňkovi za to, že mi umožnili studovat vysokou školu a postarali se o to, aby se mi dostalo patřičného vzdělání. Také bych chtěl poděkovat, že mi vždy bylo umožněno studovat to, pro co jsem měl vášeň a čemu jsem věřil. Byla to právě rodina, která mě mnohokrát dokázala podržet, když jsem několikrát chtěl se vším „seknout“. Bez jejich pomoci by tato práce s největší pravděpodobností nevznikla. Rovněž bych chtěl poděkovat vedoucímu této práce Jiřímu Hynkovi, který mi po celou dobu práce věnoval čas i v jeho osobním prostoru a sám přicházel s nápady, jak práci vylepšit. Vždy mi poskytnul cenné informace a rovněž patří k lidem, bez kterých by tato práce nevznikla. Při obtížích s implementací mi byl oporou Bc. Oliver Rýdzi. Tomu bych chtěl poděkovat, že se mi rovněž věnoval ve svém osobním volnu a při mých urgentních problémech s kódem mu nevadilo, že je sobota večer. V neposlední řadě bych chtěl poděkovat mé přítelkyni Natálii za to, že se mnou měla pevné nervy a ačkoliv to se mnou neměla mnohokrát v době psaní práce jednoduché, stála vždy při mně.

Obsah

Kapitola 1

Úvod

Žijeme v digitální době. Každým dnem vzniká na světě spousta aplikací, které nám usnadňují každodenní život tím, že nám pomáhají například organizovat čas, ovládat domácnost, vyhledávat informace nebo organizovat osobní finance. Celosvětově rovněž vzniká velká spousta aplikací, které mají za úkol sbližovat lidi – seznamky. Tyto aplikace využívají miliony uživatelů, ovšem ne všichni jsou s nimi spokojeni a část uživatelů si stěžuje na nevhodné a ofenzivní chování, či na ignoraci ze strany protějšku. Žádná z těchto aplikací nevede uživatele k tomu, aby se co nejrychleji sešel se svým protějškem. Diskuze se můžou táhnout i týdny a ve výsledku ani nemusí dojít na osobní setkání, které má při seznamování stěžejní roli. Toto, osobní zkušenost a fakt, že lidé se mnohem víc poznají osobně, než po chatu, mě inspirovalo k vymyšlení konceptu aplikace. Tato aplikace by uživatele spojovala s jinými uživateli v jejich okolí, přičemž by kladla důraz na to, aby neztráceli čas tím, že si spolu budou zbytečně dlouho psát a zároveň maximalizovala šanci, že si spolu budou rozumět.

Cílem mé práce je vytvořit mobilní aplikaci pro nejpoužívanější operační systémy na telefonech, která pomůže lidem navázat osobní kontakt při seznamování, který je dnes v těchto aplikacích opomíjen. Tato aplikace bude podporovat *Myers-Briggs Type Indicator* [22] pro určení vhodnosti protějšku, přičemž výběr protějšku bude možný i zcela náhodně. Koncept aplikace jsem postavil na osobní zkušenosti a průzkumu, který jsem udělal. Když takovou aplikaci začne používat větší množství lidí a bude ve svém úmyslu úspěšná, mohla by do značné míry ovlivnit způsob, jakým se lidé z generace Z[1] seznamují.

V následující kapitole 2 rozeberu základní teorii z oblasti psychologie a filozofie seznamování lidí. Rovněž krátce popíšu historii moderních prostředků pro seznamování a popíšu rozdíly mezi osobním a neosobním seznamováním. Následně obecně popíšu typy aplikací a rozeberu současnou situaci. V kapitole 3 zmapuji existující mobilní aplikace a způsob, jakým jsou využívány. Popíšu, jaké vstupní data vyžadují od uživatele, jaké technologie využívají a jak funguje vyhledávání protějšku. Poté v kapitole 4 vysvětlím způsoby a současné trendy, jakými je možné implementovat aplikace určené pro více platforem. Kapitola 5 bude rozebírat analýzu požadavků uživatelů a porovnání s již existujícími implementacemi. Na základě těchto požadavků poté v kapitole 6 navrhnu řešení, které je bude splňovat a detailně popíšu implementaci v kapitole 7. Způsob, jakým jsem výslednou implementaci testoval, popíšu v kapitole 8.

[1] Lidé narození od roku 1995

Kapitola 2

Seznamování lidí

2.1 Filozofický úvod

Již od pravěku měli lidé potřebu se shlukovat a vytvářet společenství. Tato potřeba měla kořeny při hledání potravy. Tehdy se vytvářely skupiny lovců a sběračů, což jsou z dnešního pohledu jedny z nejjednodušších společenství, které známe [9]. Přežít v tehdejších podmínkách bylo o poznání složitější, než při těch dnešních. V přírodě čekala velká spousta nástrah a člověk musel být neustále ve střehu. Tehdejší tlupy spolu vyrážely na lov, během kterého spolupracovaly a domlouvaly se různými skřeky a pohnutky, přičemž mnohokrát nasazovaly vlastní život. Mimo to bylo třeba udržovat oheň, který měl v pravěku zásadní roli [3].

Zvrat přišel s objevením zemědělství. Člověk přišel na to, že mu zemědělství přináší stabilní a předvídatelný přísun potravy. Také bylo jednodušší skladovat zrní, než maso, které se velice rychle zkazí. Bohužel s tím, že jídlo déle vydrží, se pojil problém toho, že jídlo bylo třeba mnohem víc chránit. Lidé se tedy začli shlukovat a stavět příbytky blízko sebe. Zde někdy můžeme hovořit o budování prvních vesnic [30]. Z těchto vesnic postupně vznikaly města a z měst dnešní velkoměsta.

Zajištění potravy a instinkt přežít ale nejsou jedinou motivací pro člověka seznamovat se. Když se podíváme na Maslowovou pyramidu lidských potřeb, což je teorie pěti hirearchicky seřazených lidských potřeb, zjistíme, že na třetím stupni se nachází potřeba lásky, přijetí a spolupatřičnosti [17]. Nejdůležitější motivací pro člověka seznamovat se je ovšem potřeba reprodukce a zachování rodu [27]. Dá se říct, že tyto potřeby z velké části ovlivňují chování člověka a mohou být uspokojeny právě vztahy, rodinou, přátelstvím, ale i náboženstvím nebo náboženskými uskupeními. Vyhledáváme společnost, abychom se necítili osamoceni a nepropadali depresím, potřebujeme lásku, abychom se cítili milováni a obklopujeme se přáteli, kteří nás pochopí.

2.2 Historie seznamování

Lidstvo se v průběhu svého vývoje seznamovalo mnoha způsoby. Jednoduše je můžeme rozdělit na osobní a neosobní (rozvedené v sekcích 2.2.1 a 2.2.2). Obě tyto kategorie mají svá pro i proti, ovšem člověku přirozenější je z historického hlediska osobní seznamování. Ne vždy byly totiž dostupné technologie, které umožňovaly dvěma libovolně vzdáleným lidem spolu komunikovat. Než lidstvo došlo k online seznamování tak, jak ho známe dnes, vymyslelo mnoho jiných způsobů založených na osobním i na neosobním seznamování. Mimo jiné se má za to, že k vyhledávání nových cest k seznámení přispěly také současné moderní pří-

stupy k manželství, které mají za následek i to, že se mladí lidé berou později, nebo omezení možnosti seznámit se na pracovišti jako důsledek sexuálního obtěžování [12]. Můžeme tedy předpokládat, že budoucnost seznamování bude opět ovlivněna současnými trendy.

Jak jsem již zmínil v sekci 2.1, člověk od pradávna žil v tlupách a nějaký druh seznamování pro něj byl tedy naprosto přirozený. Postupně, s vývojem lidstva přicházely nové způsoby, jak se s někým seznámit. Pro zajímavost můžeme říct, že ve středověku určovali dětem partnery převážně rodiče. Ty mnohdy neměly na výběr a ačkoliv s rozhodnutím svých rodičů nesouhlasily, musely se mu podřídit. Pro potřeby této práce je nám historicky nejbližší způsob seznamování se pomocí inzerátu v novinách. Ty fungovaly (a do určité míry stále fungují) tak, že osoba, která hledá svůj protějšek, nechala vytisknout svůj seznamovací inzerát do novin a následně čekala na kontaktování od některé z osob, které si daný inzerát přečetly. Pokud někoho inzerát zaujal, měl možnost použít například poštu a domluvit si schůzku. První inzeráty tohoto druhu se objevovaly již v 17. století [1]. Tehdy se v inzerátech velice často objevovaly informace o majetku a věnu. Byli to primárně muži, kdo tyto inzeráty umisťovali do novin, jelikož u žen to bylo nepřípustné. Tyto inzeráty v novinách postupně převzaly některé magazíny a v určitě formě se objevovaly například i v televizích – u nás například v televizi Óčko[1]. Pokud jste chtěli vytvořit ve vysílání této televize inzerát, stačilo zaslat SMS ve specifickém tvaru s přiloženým vzkazem na číslo, které se dívákům v pravidelných intervalech ukazovalo ve spodní liště vysílání.

Obrázek 2.1: Inzerát na televizi Óčko. Zdroj: https://www.youtube.com/watch?v=dJt0I9KDUqU

Přibližně ve stejné době se v Česku podobné inzeráty rozmohly na internetu. Samotný nápad, že by se lidé mohli seznamovat online implementoval jako jeden z prvních Terrence „Lee“ Zehrer, když v roce 1994 vymyslel server Kiss.com[2]. O pár let později, v roce 1998,

[1] https://ocko.tv/
[2] www.kiss.com

vznikla česká internetová seznamka Seznamka.cz[3]. Na této seznamce si lidé jednoduše vytvořili profil a procházeli podobně sestavené profily napříč celou sítí. Poté se mohli kontaktovat, domluvit se na schůzce a sejít se. Jedná se o jednu z prvních seznamek, které v Česku vznikly. Krátce na to v roce 2000 ve světě vznikla seznamka eharmony[4]. Nejpodstatnější věcí, kterou se od všech ostatních seznamek lišila, byl způsob, jakým se lidé na platformě mohli seznámit. Jedná se o první aplikaci, která uživatelům nabídla výběr protějšku na základě algoritmu. Po registraci na síti byl uživatel vyzván k vyplnění dotazníku, který měl za cíl získat o uživateli dostatek informací. Poté na základě výsledku těchto dotazníků vytvářel propojení s jinými uživateli sítě a nabízel možnost začít si chatovat.

Mimo online prostor přišel v roce 1998 americký rabín Yaacov Deyo s revolučním nápadem – *speed dating* [8], kterým chtěl zvýšit množství židovských svateb. Ten spočívá v tom, že k ženám, které jsou rozesazeny každá k vlastnímu stolu postupně přichází muži. Po tom, co se usadí, mají zpravidla 3-8 minut na to, aby se poznali s protějškem. Po uplynutí této doby se muži přesunou k jinému stolu. Poté, co se všichni vystřídají, dají organizátorům vědět, který z protějšku mu byl sympatický. Organizátoři následně dají těm, u kterých došlo k vzájemné shodě, kontakt na protějšek [8].

Od doby, kdy se mezi mladými lidmi rozvinul trend online technologií, se objevovaly náznaky, že by se lidé jednou mohli seznamovat po internetu. Jedna z prvních aplikací, která se v Česku pro tyto účely používala byla například ICQ[5]. Svou roli hrál i Skype[6], který oproti ICQ umožňoval i videochat. Tyto dva nástroje byly delší dobu používány jako jedny z hlavních – než přišel Facebook[7]. Ten měl v sobě zabudovaný chat a jelikož se jednalo o nejrychleji rostoucí sociální síť, mnoho mladých lidí používalo právě jej. Používání moderních technologií pomáhala i situace s vývojem HW. Začaly vznikat chytré telefony a tablety. Člověk tedy mohl mít chat se svou vysněnou partnerkou/partnerem v podstatě všude s sebou. Další podstatná změna přišla s nástupem GPS technologií v mobilních telefonech. Od té doby bylo možné zároveň vytvářet spojení a psát si s lidmi ve vašem okolí.

2.2.1 Osobní seznamování

Pokud se bavíme o osobním seznamování, v průběhu let se do společnosti adaptovaly mnohé způsoby – seznamování se na diskotékách, seznamování se na různých společenských, či kulturních událostech, seznamování se na již zmíněném *speed dating* a nebo v barech, či restauracích. Tento typ seznamování umožňuje pozorovat řeč těla. Již víme, že z řeči těla se dá vyčíst o člověku opravdu mnoho [7]. Dalo by se říct, že z řeči těla poznáme, v jaké náladě se člověk aktuálně nachází, jestli je přikloněný k diskuzi s námi, nebo dokonce, zda k nám má nějaké sympatie. Pro mnohé lidi může být samotná řeč těla atraktivní [7]. Ačkoliv neverbální komunikace sama o sobě vypovídá o člověku mnoho, jejím doplňkem je verbální komunikace. I tu ale můžeme pozorovat výhody osobního kontaktu. Když slyšíme někoho mluvit, můžeme z tónu a intonace jeho hlasu poznat například, zda je sebevědomý, nebo zda věci, které říká, myslí vážně [28]. Z hlasového projevu můžeme dokonce odhadnout mentální kapacitu daného člověka [26]. Tento způsob seznamování nám příjde přirozeně lepší, jelikož se s ním lidé setkávali od pradávna.

[3] https://www.seznamka.cz/
[4] https://www.eharmony.co.uk/
[5] https://icq.com/
[6] https://www.skype.com/cs/
[7] https://www.facebook.com

Mezi nejdůležitější faktory řeči těla patří:

- oční kontakt,
- postavení těla,
- výraz tváře,
- gestikulace rukou

Všechny výše zmíněné faktory ovlivňují to, jakým způsobem na nás člověk působí. Pokud k nám například někdo příjde, má ramena vysunutá dopředu a hlavu zklopenou dolů, víme, že tento člověk nejspíš nebude sebevědomý. Nemusí nám být příjemné, když se s námi někdo baví a příliš mává rukama. Pokud se s někým bavíme a on se místo na nás dívá do země, může nám připadat lhostejný.

2.2.2 Neosobní seznamování

Naproti tomu neosobní seznamování tyto důležité faktory postrádá. Během toho, co si s někým dopisujeme, sice máme k dispozici jeho nepsaný projev, ten je ale oproti mluvenému slovu, či gestikulaci, relativně nevypovídající. Přesvědčit se o tom můžeme například ve studii, která vznikla v roce 2003 na Iowské státní univerzitě [16]. V této studii vybrali 64 studentů, kteří se do té doby neznali, ze kterých utvořili dvojice. Těmto dvojicím bylo poté náhodně určeno, zda spolu mají komunikovat přes internet, nebo z očí do očí. Výsledkem bylo, že páry, které měly za úkol se seznamovat offline, byly celkově spokojenější s jejich interakcí, méně se hádaly a připadaly si více propojení.

Bohužel ne vždy máme pro osobní seznamování prostor. Může se stát, že lidé, kteří jsou velmi vytížení, preferují neosobní typ seznamování před tím, než se s někým sejdou. Důvodem může být, že čas, který stráví s člověkem, kterého neznají, považují za zbytečně ztracený. Rovněž lidé, kteří jsou spíše introverti než extroverti, mohou upřednostňovat tento typ seznamování, jelikož se mohou před protějškem stydět a dopisování jim může být příjemnější. Další důležitou věcí je, že při dopisování máme více prostoru přemýšlet nad tím, co chceme napsat a můžeme se tím vyvarovat některým zbytečným konfliktům, které velice často můžou vzniknout právě díky tomu, že psaný projev postrádá emoce. Částečně tyto emoce mohou nahradit tzv. emotikony[8]. Z těch ale není vždy zřejmá nálada autora, jelikož různé emotikony můžou být chápány odlišně. Například emotikona „:)“ může být kromě svého zamýšleného vyjádření radosti vnímána také jako emotikona pro vyjádření ironie.

[8] Vyjádření nálady znaky, které svým seskupením připomínají výraz tváře

2.3 Typy aplikací

Na trhu se postupem času objevila velká spousta aplikací. Některé z nich se zachovaly, jiné zanikly v důsledku toho, že je konkurenční aplikace „převálcovaly“, nebo protože o ně nebyl dostatečně velký zájem. Zajímavé je ale zaměření těchto aplikací a samotná cílová skupina uživatelů. Samotné aplikace můžeme rozdělit do čtyř typů [23]. Jednotlivé aplikace můžou spadat i do více, než jednoho typu. Zmíněnými typy jsou:

1. **Geolokační aplikace** – aplikace využívající polohu uživatele pro určení shody.
2. **Aplikace založené na algoritmu pro vyhledávání shody** – využívají jako vstupní data uživatele dotazníky.
3. **Tradiční aplikace** – jedná se o mobilní verze již zaběhlých seznamek.
4. **Aplikace zaměřené na určité skupiny** – vyznačují se tím, že cílí pouze na určitý segment zákazníků.

Do typu 4 spadají například aplikace, které jsou určené primárně pro homosexuály. Jelikož seznamování na standardních aplikacích by pro ně mohlo být složité, přešli na tento typ platforem. Typickým zástupcem této skupiny je aplikace Grindr[9]. Existuje také druh seznamovacích aplikací, které berou ohled na politický názor uživatele a nabízí mu protějšky s podobným politickým přesvědčením – například MapleMatch[10]. Ta rovněž spadá do typu 4. Dále tu jsou aplikace, které mají za cíl jednoduše najít partnera na jednu noc, nebo kamaráda/ku s výhodami. Jako příklad můžeme uvést aplikaci Pure[11]. Touto aplikací se také (poněkud nechtěně) stal Tinder[12] – zde je ovšem na vině komunita. Rovněž existují aplikace pro muže, kteří se stydí a ocenili by, kdyby jim žena napsala první. Nejznámějším zástupcem z této skupiny je aplikace Bumble[13]. Řešení je i pro lidi, jejichž kamarádi nejsou spokojení s opakovaným špatným výběrem partnera. Ship[14] jednoduše umožňuje vašim přátelům vybrat ideálního partnera pro vás podle jejich představ – slogan aplikace je „Randěte s někým, koho Vaši přátele již mají rádi“. Pure, Tinder, Bumble i Ship jsou zástupci typu 1, jelikož všechny využívají technologie pro určení polohy[15] jako hlavní parametr k nalezení shody.

Velice specifickou skupinou aplikací jsou aplikace zaměřené na vyhledávání tzv. *sugar daddy*. Přes tyto aplikace hledají muži primárně mladší dívky, které jim nabízejí „randění“. Chodí s nimi do kina, nakupovat, na večeře a nebo do baru na skleničku. Jednoduše jim dělají společnost. *Sugar baby*[16] by měla mít dobré vychování, znát etiketu, umět se chovat na veřejnosti, přičemž se počítá i s určitou úrovní inteligence. *Sugar daddy* se poté každý měsíc odvděčí své *sugar baby* pěněžitou odměnou, jejíž výše je na domluvě. Obdobné aplikace existují i pro muže. Ti si hledají své *sugar mummy*, které jim následně pomáhají se studiem

[9] https://www.grindr.com/
[10] https://www.maplematch.com/
[11] https://pure.app
[12] https://tinder.com
[13] https://bumble.com/
[14] https://getship.co/
[15] Může se jednat o GPS, WiFi sítě, nebo technologie počítání vzdálenosti zařízení na základě síly signálu mezi vysílači
[16] Dívka, kterou *sugar daddy* na platformě hledá

a kupují pěkné věci. Zástupci této skupiny jsou například aplikace Suger[17], nebo Sugar momma dating cougar[18].

Mezi aplikace založené na algoritmu (typ číslo 2), které byly vytvořeny jako dodatek již existující, funkční platformy (typ čislo 3), patří eHarmony[19]. Ta uvádí, že používá svůj 20 let prověřený algoritmus pro nalezení ideální shody.

Co se týče míry informací, které musíme seznamkám poskytnout a které seznamky poskytnou našemu protějšku, jedná se velice často o ty samé. Základními informacemi, které musíme seznamce většinou poskytnout jsou:

- jméno (nemusíme nutně sdělovat příjmení),
- datum narození,
- pohlaví,
- sexuální orientace

Některé od nás navíc vyžadují vyplnit seznam našich koníčků, preference co se týče našeho ideálního partnera a nebo dotazník (obvykle typ číslo 2), který má seznamce pomoct odkrýt naše potřeby tak, jak si je sami neuvědomujeme.

Ačkoliv existují i seznamky, které nutně nevyžadují fotku, drtivá většina z nich ji považuje za nutnost a není možné na nich vytvořit profil bez fotky. Rovněž většina aplikací umožňuje uživateli napsat krátký popis sebe sama. Existují ovšem i aplikace, které vyžadují, aby jim uživatel sdělil, na kolik přibližně odhaduje svůj majetek. Tyto aplikace slouží převážně velmi bohatým lidem, kteří si zakládají na tom, aby člověk, se kterým randí, byl bohatý a samostatný. Jednou z těchto aplikací je třeba Luxy[20].

Většina aplikací využívá technologie sledování zařízení pro určení uživatelovy polohy a zpřesnění výsledků tak, aby uživateli byly nabízeny protějšky z jeho okolí. V době, kdy se ve světě rozšiřoval virus SARS-CoV-2 způsobující nemoc covid-19, některé ze zaběhlých seznamek od tohoto upustily a povolily uživateli se „přesunout“ do jakéhokoliv místa na světě – konkrétně to zavedl například Tinder. Uživatel tak dostal možnost bavit se s lidmi z opačného konce světa. Existují i implementace, které na polohu uživatele neberou ohled, ty ale ve výsledku působí spíše jako telefonní seznamy, než jako aplikace, přes které bychom měli najít svého ideálního partnera.

Co se týče kalkulace vhodnosti protějšku, tak zde se aplikace liší podstatně více. Některé moderní aplikace mají jako jediná kritéria pro výběr partnera věk (požadovaný věk protějšku si uživatel sám nastaví) a polohu, jiné počítají, zda se k sobě dva lidi hodí na základě shody v dotazníkách. Výběr ideálního partnera může být ovšem i naprosto náhodný. Algoritmizace problému seznamování a hledání ideálního protějšku je velice složitá. Jedná se totiž o proces, který obsahuje velkou spoustu proměnných. Existují i aplikace, které počítají vhodnost protějšku na základě *Myers-Briggs Type Indicator*, který je podrobněji vysvětlen v sekci 2.5. Takové aplikace určí uživateli na základě jeho odpovědí v krátkém dotazníku jeden z 16 typů osobností. Na základě tohoto typu osobnosti poté určí ideální typy osobností pro jeho protějšek a nabízí mu uživatele, kteří tento typ osobnosti mají.

[17] https://play.google.com/store/apps/details?id=com.dating.suger&hl=en_US&gl=US
[18] https://play.google.com/store/apps/details?id=com.hookup.hookup.hookup&hl=es_GT
[19] https://www.eharmony.co.uk/
[20] https://www.onluxy.com/

2.4 Charakteristiky současných aplikací

V současnosti většina aplikací, které vznikají za účelem seznámení dvou lidí těží z výhod použití lokačních technologií. Většina z nich pracuje s myšlenkou přizpůsobení výběru protějšků na základě naší současné polohy. Použití těchto technologií s sebou ovšem nese velké množství nařízení ať už od samotné Evropské unie nebo jiných subjektů, které mohou bojovat za práva uživatelů a ochranu uživatelských údajů. Není například povolené uchovávat v databázích historii polohy uživatele. Pro nás největší změny ve způsobu uchovávání informací o uživateli přinesla v roce 2016 Evropská unie schválením obecného nařízení o ochraně osobních údajů dnes známého spíše pod zkratkou GDPR[21].

Mezi nejpopulárnější aplikace, které dnes využívají lokačních technologií patří Tinder[22], Badoo[23] a nebo OkCupid[24]. Většina těchto aplikací rovněž používá téměř totožný způsob výběru protějšku – takzvané *„svajpování“*. Jedná se o způsob výběru protějšku. Název je odvozen od anglického slova *swipe*. Jedná se o tah prstem doleva, či doprava na základě toho, zda se nám nabízený protějšek líbí. Zpravidla svajpnutí doleva znamená, že nelíbí, svajpnutí doprava, že líbí.

Všechny známější aplikace mají nastavený určitý systém monetizace[25]. Uživatel, který si zaplatí nějakou úroveň předplatného (většina aplikací nabízí možnosti měsíčního předplatného) dostane typicky výhody v podobě zobrazení protějšků, které mu daly *„lajk“*[26], zvětšení dosahu svého profilu, možnost neomezéno *svajpování*, na některých platformách možnost super lajků[27] a nebo zrušení reklam. Na některých platformách jsou tyto systémy poněkud nevyvážené, jelikož uživatelé, kteří si je předplatí, jsou značně zvýhodněni oproti neplatícím uživatelům, což z těchto služeb dělá takzvané *pay-to-win*[28] služby.

Současné aplikace s sebou nesou také řadu různých rizik. Tím, že mnohé z nich používají polohu, se můžou stát nástrojem pro různé kriminální činy. Poněkud nebezpečný může být i fakt, že nevíme, s kým si píšeme. Tuto hrozbu dokázal dobře odkrýt český dokumentární film V síti[29] režisérů Víta Klusáka a Barbory Chalupové, který ukázal, jak se na dnes již zaniklé sociální síti Lidé.cz[30] predátoři nebáli oslovovat mladé, mnohdy nezletilé dívky a lákat je na osobní setkání. Na výše zmíněných aplikacích se také velice často objevují profily, které ve skutečnosti nejsou pravé a na kterých se osoba pouze za někoho vydává. Nemilé můžou být i zjištění, že se za nás někdo vydává na seznamovací aplikaci, kterou jsme v životě nepoužili.

Je ovšem důležité říct, že seznamky tak, jak fungují dnes, mají i velkou spoustu výhod. Jedna z výhod se například ukázala v době celostátního lockdownu v České republice. Lidé měli okamžitou možnost psát si s lidmi z jejich okolí a zachovat nějakou úroveň sociální interakce, byť jen minimální. Další výhodou online seznamování může být, že neztrácíme zbytečný čas konverzováním s lidmi, se kterými si zřejmě nerozumíme (ne každý má odvahu se na schůzce jednoduše zvednout ze židle a odejít pryč). Online seznamování s lidmi může

[21] https://www.gdpr.cz/gdpr/
[22] https://tinder.com/
[23] https://badoo.com/cs/
[24] https://www.okcupid.com/
[25] Proces zpeněžení aplikace
[26] Vyjádření zalíbení
[27] Unikátní typ lajku, který zdůrazňuje zalíbení a většinou je protějšku zobrazen zvlášními grafickými prvky
[28] Služba, ve které uživatel získává značné výhody zakoupením předplatného, nebo jednotlivých nabízených balíčků.
[29] https://www.csfd.cz/film/720753-v-siti/prehled/
[30] https://www.lide.cz/

být také příjemnější pro ty z nás, kdo se trochu více stydíme. V samotné diskuzi jsou pak lidé otevření i co se týče toho, co od vztahu očekávají a to platí nejen pro stydlivé lidi. Obrovskou výhodu dnešních seznamovacích aplikací můžeme pozorovat, když se ocitneme ve městě, kde nikoho neznáme. Na většině těchto platforem se můžeme velice rychle seznámit například s místními obyvateli, kteří nás můžou provést nám neznámým městem a ukázat místní zvyklosti.

Každopádně můžeme říct, že seznamovací aplikace se staly nedílnou součástí seznamování. To se promítá i do předpokladu vývoje hodnoty trhu online seznamování, u kterého se v roce 2025 předpokládá s hodnotou přibližně 3.5 miliardy dolarů [24]. Toto podtrhuje i razance, se kterou se ve světě vyvíjejí nové seznamovací aplikace.

2.5 Myers-briggs indikátor

Myers-briggs indikátor typů je osobnostní test na určení toho, jak lidé vnímají svět kolem sebe a reagují na něj. Vytvořila jej Katharine Briggsová spolu s Isabel Myersovou a je založený na studii Carla Junga, který ve své práci definoval základní typy osobností [11]. Autorky poté práci rozšířily a definovaly 4 základní klíče k určení osobnostního typu. Na základě tohoto osobnostního typu poté lze obecně definovat chování, které by měl testovaný člověk vykazovat. Zmíněnými základními kliči jsou:

1. **vnímání okolního prostředí** – extraverze (E) , introverze (I),
2. **získávání informací** – smysly (S), intuice (N),
3. **zpracování informací** – myšlení (T), cítění (F),
4. **životní styl** – usuzování (J), vnímání (P)

Osobnostní typ se poté vytváří kombinací těchto klíčů, kdy z každého klíče vybereme jednu možnost a seřadíme je za sebe přesně ve výše naznačeném pořadí. Vznikne tak 16 možných kombinací – tedy 16 osobnostních typů. Jednotlivým osobnostním typům jsou přiděleny názvy, které reflektují význam daného typu. Například:

- Velitel **ENTJ** – nezávislý, efektivní a strategický typ člověka. Jsou to dobří, efektivní organizátoři.
- Obhájce **ESFJ** – kamarádský a spolehlivý typ. Snaží se pomáhat druhým.
- Advokát **INFJ** – idealistický, organizovaný a nezávislý typ. Má rád spolupráci a vyhledává intelektuální konverzace.
- Kutil **ISTP** – analytický, spontánní a akční typ. Má rád dobrodružství.

Existují studie, ve kterých nalezneme pravděpodobnost kompatibility dvou osobnostních typů. Na obrázku 2.2 jsou tyto pravděpodobnosti doplněny o barevnou signalizaci. Čím vyšší číslo, tím větší pravděpodobnost vytvoření spojení. Můžeme například říct, že je daleko větší pravděpodobnost, že si spolu budou rozumět ENTJ a INFJ, než ENTJ a ISTP [21]. Ačkoliv je Myers-Briggs indikátor typů často zpochybňován z důvodů, že je vágní a obecný, dnes se používá v pedagogice, kariérním poradentstvím ale i v managementu [19, 18].

	ESTP	ISFP	ISTP	ESFP	ESTJ	ESFJ	ISTJ	ISFJ	ENFJ	INFJ	ENFP	INFP	ENTJ	INTJ	ENTP	INTP
ESTP	0,040	0,296	0,506	0,506	0,296	0,296	0,506	0,296	0,714	0,506	0,506	0,506	0,296	0,714	0,867	0,506
ISFP	0,296	0,110	0,506	0,139	0,296	0,296	0,139	0,296	0,296	0,867	0,867	0,506	0,714	0,714	0,506	0,506
ISTP	0,506	0,506	0,259	0,296	0,867	0,506	0,714	0,867	0,139	0,296	0,714	0,296	0,139	0,506	0,714	0,714
ESFP	0,506	0,139	0,296	0,460	0,506	0,867	0,714	0,506	0,506	0,296	0,296	0,714	0,506	0,139	0,296	0,296
ESTJ	0,296	0,296	0,867	0,506	0,680	0,714	0,867	0,952	0,296	0,139	0,506	0,139	0,296	0,296	0,506	0,506
ESFJ	0,296	0,206	0,506	0,867	0,714	0,840	0,867	0,714	0,296	0,506	0,506	0,506	0,714	0,051	0,139	0,139
ISTJ	0,506	0,139	0,714	0,714	0,867	0,867	0,940	0,867	0,506	0,296	0,296	0,296	0,506	0,139	0,296	0,296
ISFJ	0,296	0,296	0,867	0,506	0,952	0,714	0,867	0,940	0,296	0,139	0,506	0,139	0,296	0,296	0,506	0,506
ENFJ	0,714	0,296	0,139	0,506	0,296	0,296	0,506	0,296	0,840	0,506	0,139	0,506	0,714	0,714	0,506	0,506
INFJ	0,506	0,867	0,296	0,296	0,139	0,506	0,296	0,139	0,506	0,680	0,714	0,714	0,867	0,506	0,296	0,296
ENFP	0,506	0,867	0,714	0,296	0,506	0,506	0,296	0,506	0,139	0,714	0,460	0,296	0,506	0,506	0,714	0,296
INFP	0,506	0,506	0,296	0,714	0,139	0,506	0,296	0,139	0,506	0,714	0,296	0,250	0,506	0,506	0,296	0,714
ENTJ	0,296	0,714	0,139	0,506	0,296	0,714	0,506	0,296	0,714	0,867	0,506	0,506	0,110	0,296	0,139	0,139
INTJ	0,714	0,714	0,506	0,139	0,296	0,051	0,139	0,296	0,714	0,506	0,506	0,506	0,296	0,030	0,867	0,867
ENTP	0,867	0,506	0,714	0,296	0,506	0,139	0,296	0,506	0,506	0,296	0,714	0,296	0,139	0,867	0,110	0,714
INTP	0,506	0,506	0,714	0,296	0,506	0,139	0,296	0,506	0,506	0,296	0,296	0,714	0,139	0,867	0,714	0,250

Obrázek 2.2: Graficky znázorněná pravděpodobnost vytvoření spojení v sociální síti [21]

Kapitola 3

Existující mobilní aplikace

Jak již bylo zmíněno v kapitole 2, na světě vzniká velká spousta aplikací. O většině těchto aplikací se s největší pravděpodobností nikdy nedozvíme. Důvody jsou různé. Může to být slabý marketing, převálcování giganty, špatný návrh a nebo celkově špatná myšlenka aplikace. V této kapitole postupně analyzuji dnes nejpoužívanější mobilní aplikace, které jsou používány pro seznamování lidí.

U každé z aplikací uvádím, jaké informace od uživatele vyžaduje pro vytvoření profilu, jak vypadá základní použití dané aplikace a jak funguje vyhledávání protějšků z uživatelské perspektivy. Dále zhodnocuji přívětivost a intuitivnost uživatelského rozhraní. Veškerý popis aplikací bude založen na produkční verzi aplikace dostupné v obchodě App Store[1] na zařízení iPhone X s verzí operačního systému iOS 14.4. Dílčí části uživatelského rozhraní se můžou pro jiné operační systémy a mobilní zařízení lišit.

3.1 Tinder

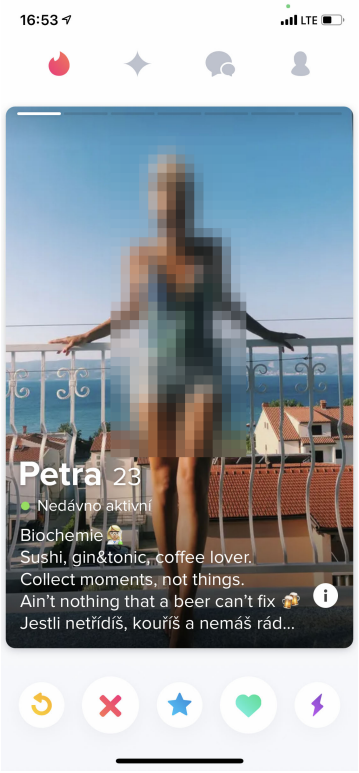
Začneme s dnes nejpoužívanější aplikací pro seznamování v USA [29] dostupnou pro platformy iOS a Android. Tato aplikace spadá do typu 1 – tedy využívá polohu zařízení pro vyhledání shody. Je přístupná pro lidi starší 18 let a pro vytvoření profilu na této seznamce je potřeba vyplnit své telefonní číslo, datum narození, jméno, pohlaví, nahrát fotku a vybrat, co na platformě hledá. Uživatel má možnost si v aplikaci vytvořit účet třemi způsoby:

- přihlášení prostřednictvím Apple ID,
- přihlášení přes Facebook,
- přihlášení prostřednictvím telefonního čísla

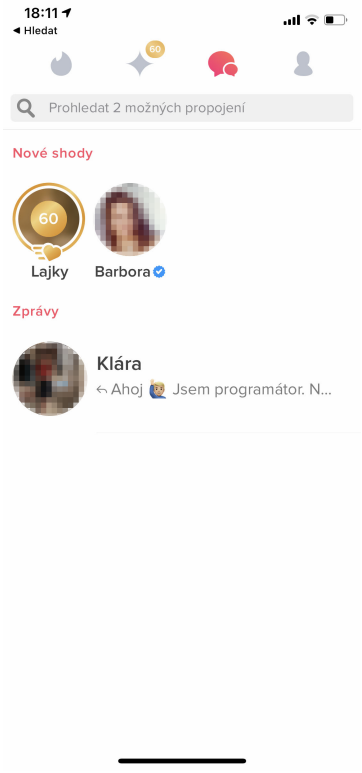
Při zadávání data narození je třeba mít se na pozoru. Pokud totiž uživatel zadá, že nedovršil plnoletosti, Tinder automaticky zablokuje vytváření profilu z daného telefonního čísla, dokud uživatel nedovrší plnoletosti (obrázek 3.4). Tato blokace účtu se nedá zvrátit.

Po registraci aplikace nabídne uživateli svou hlavní obrazovku 3.1. Na této stránce se uživateli zobrazují protějšky z jeho okolí. Pokud chceme vidět další nebo předcházející fotku uživatele, musíme kliknout do levé, či pravé části aktuálně zobrazované fotky. Pro zobrazení profilu musíme kliknout na fotku přibližně někam doprostřed. Uživatel může zobrazený protějšek ohodnotit celkem třemi způsoby:

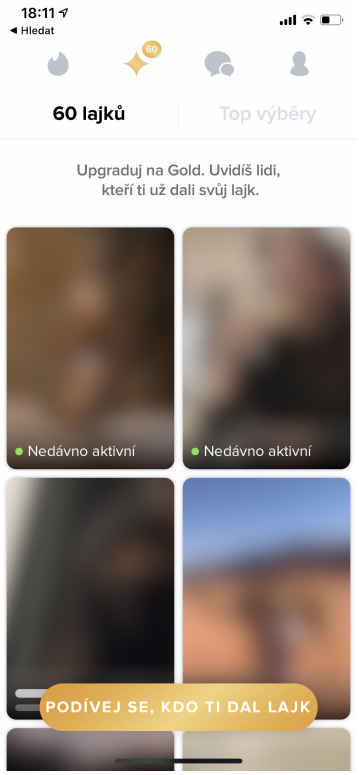
[1] https://www.apple.com/cz/ios/app-store/



Obrázek 3.1: Hlavní stránka tinderu. Obrázek byl anonymizován.



Obrázek 3.2: Chaty na Tinderu. Obrázek byl anonymizován.



Obrázek 3.3: Ukázka zpoplatněné funkcionality na Tinderu

- „Líbí se mi“ – potáhnutí prstem z levé strany do pravé,
- „Nelíbí se mi“ – potáhnutí prstem z pravé strany do levé,
- „Super lajk“ – potáhnutí prstem ze spodní strany nahoru

Pokud uživatel zvolí, že se mu vybraný protějšek líbí a zároveň protějšek uživateli již v minulosti dal lajk, vytvoří se spojení. V takovém případě lze poté zahájit s potenciálním partnerem/partnerkou chat, jak můžeme vidět na obrázku 3.2 v horní liště „Nové shody“. V případě, že protějšek zatím nedal lajk uživateli, aplikace nám bude dál ukazovat vybrané potenciální partnery/partnerky. Pokud se uživatel rozhodne dát někomu super lajk, musí brát na vědomí, že super lajků mají všichni uživatelé pouze přidělený počet. Ti, co si neplatí žádnou variantu předplatného, mají dostupný pouze jeden super lajk za den. Oproti tomu předplatitelé mají dostupných až 5 super lajků každý den. Super lajk je oproti klasickému lajku graficky zvýrazněn. Navíc super lajk je viditelný uživateli při procházení potenciálních partnerů na hlavní obrazovce (obrázek 3.1. Tímto způsobem na sebe tedy můžeme upozornit.

Ve stejné liště, kde se nachází „Nové shody“ vidíme na prvním místě poutač na protějšky, které nás lajkly. Tento poutač ovšem vede pouze na stránku s rozmazanými fotkami 3.3. Zde vidíme příklad zpoplatnění Tinderu. Uživatelé, kteří si předplácí některý z programů Tinderu využívají kromě neomezeného počtu lajků a viditelnosti lidí, kterým se líbí, také benefity v podobě neomezeného počtu lajků nebo upřednostnění ve frontě.

Obrázek 3.4: Zablokovaný uživatelský profil na přibližně 12 let

Jak již bylo zmíněno na začátku této sekce, Tinder využívá technologií pro určení polohy zařízení, aby vybral vhodné protějšky v okolí. Kromě této metriky používá také věk a pohlaví. Uživatel má v nastavení svého profilu možnost vybrat si okruh, ve kterém má Tinder hledat a ideální věk partnera/partnerky. Zbytek je náhodný.

3.2 Bumble

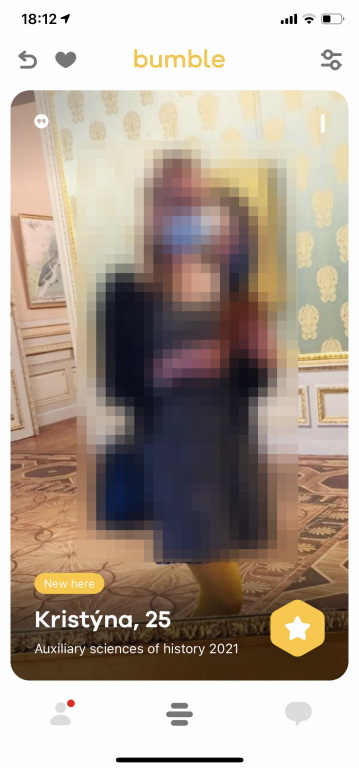
Jedná se o druhou nejpoužívanější aplikaci [29]. Liší se především konceptem chatování a udržování propojení.

Při registraci je uživatel požádán o jméno, nahrání fotky, datum narození, výběr pohlaví, výběr pohlaví protějšku a výběr módu, ve kterém aplikaci bude používat. Kromě módu „Date“ jsou zde i módy „BFF“ – zaměřené na vyhledávání nových přátel a „Bizz“ – pro hledání pracovních příležitost.

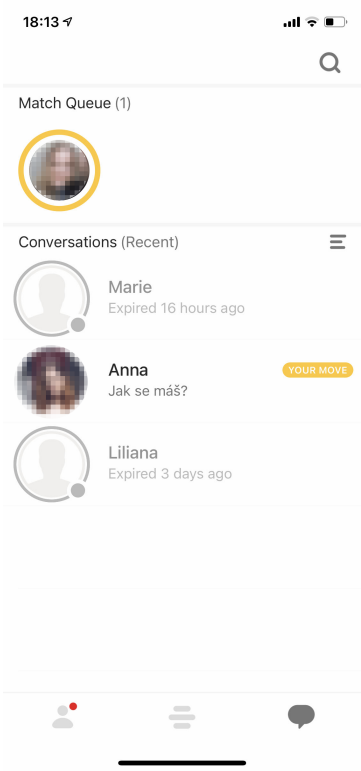
Způsob, jakým uživatel interaguje s hlavní obrazovkou (obrázek 3.5) je totožný s aplikací Tinder 3.1. Zásadní změnou je, že v případě shody muže a ženy, musí udělat první krok žena a napsat muži. Nesmí ale se zprávou dlouho otálet, jelikož po vytvoření shody má pouze 24 hodin na úvodní zprávu. Po uplynutí této doby chat bez úvodní zprávy zmizí a již nikdy nebude možné opětovně vytvořit toto spojení.

V kontextu typů, jež byly definovány v sekci 2.3 spadá Bumble do typu 1. Rovněž tedy používá lokaci zařízení pro určení vhodných protějšků. Narozdíl od Tinderu zde ale najdeme uživatele z celé země a někdy se nám dokonce objeví zahraniční uživatelé. Poloha tedy není použitá tak, že by filtrovala výsledky v určitém okruhu. Jde spíš o nějaké určení přibližné lokace. Do algoritmu na Bumble navíc vstupují proměnné ve filtru. Uživatel má možnost nastavit si určité vlastnosti, které u protějšku očekává, jako například vzdělání,

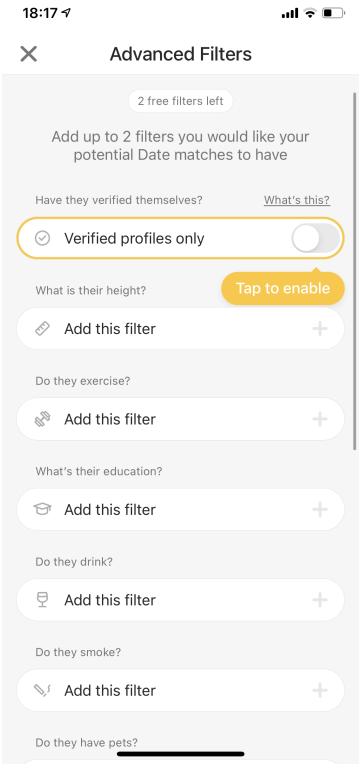
výšku, zda pijí, zda cvičí, nebo jaké jsou znamení 3.7. Bumble následně nabízí uživateli pouze ty výsledky, které tomuto filtru odpovídají.



Obrázek 3.5: Hlavní stránka aplikace Bumble. Obrázek byl anonymizován.



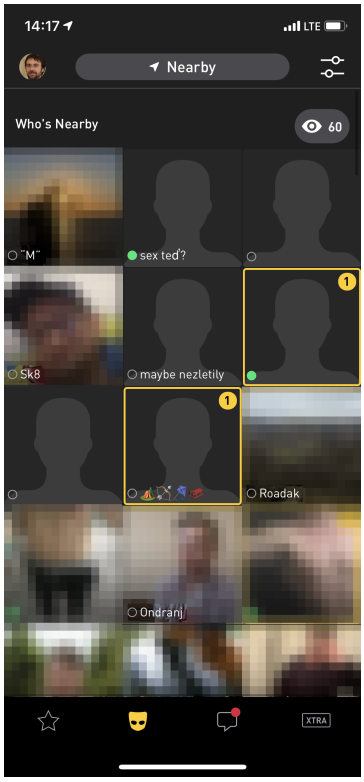
Obrázek 3.6: Chaty na Bumble. Obrázek byl anonymizován.



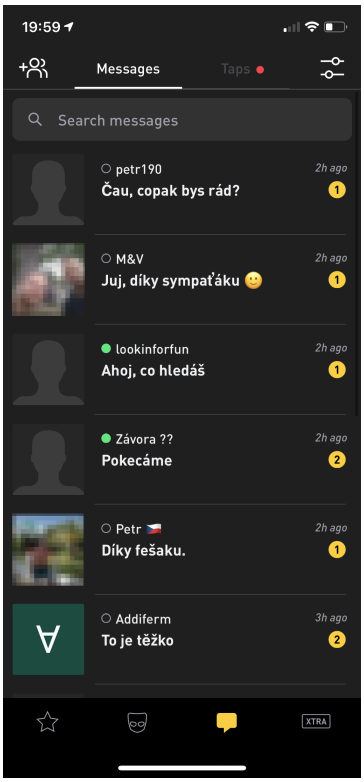
Obrázek 3.7: Filtrování výsledků

3.3 Grindr

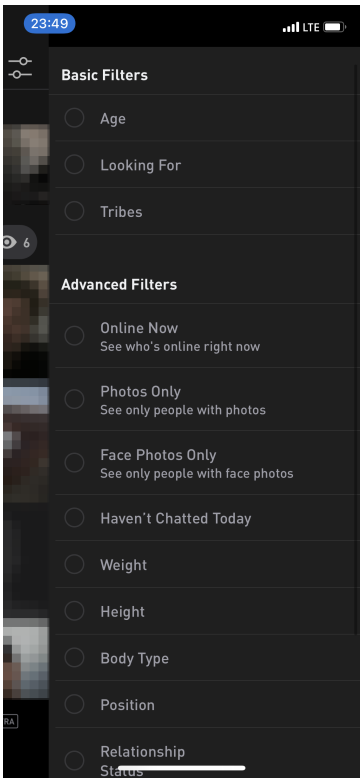
Jedná se o nejrozšířenější seznamku pro homosexuály, můžeme tedy říct, že spadá do typu 4. K vytvoření profilu na seznamce stačí zadat jméno, datum narození, nahrát profilovou fotku a vybrat pohlaví. Poté se zobrazí hlavní stránka aplikace Grindr. Na té nalezneme mřížku sestavenou z profilů uživatelů, kteří jsou v našem okolí 3.8 — aplikace tedy spadá také do typu 1. Pokud uživatel nechce zobrazovat výsledky z jeho okolí, může přepnout z režimu „nearby“ do režimu manuálního nastavení polohy a prozkoumat uživatele v jiném městě. Na rozdíl od aplikací jako Tinder 3.1 nebo Bumble 3.2 Grindr nefunguje na principu svajpování. Uživatel může kdykoliv napsat jiným uživatelům, kteří se nachází v jeho blízkosti a jsou zobrazeni na hlavní stránce. Počet potenciálních partnerů, které uživatel vidí na hlavní stránce je limitován na 102. Způsob, jakým můžeme na Grinderu někomu vyjádřit zalíbení, jsou plamínky. Tyto plamínky můžeme rozdat všem uživatelům, které máme na hlavní stránce. Ovšem jednomu konkrétnímu uživateli můžeme dát pouze jeden plamínek za 24 hodin. Uživatel se může kdykoliv podívat na plamínky, které dostal od ostatních uživatelů v záložce „Taps“, kterou vidíme na obrázku 3.9. Navíc se mu uživatelé od kterých dostal plamínek zobrazí se žlutým rámečkem kolem profilové fotky 3.8.



Obrázek 3.8: Hlavní stránka aplikace Grindr. Obrázek byl anonymizován.



Obrázek 3.9: Chaty na Grindr. Obrázek byl anonymizován.



Obrázek 3.10: Filtrování výsledků. Obrázek byl anonymizován.

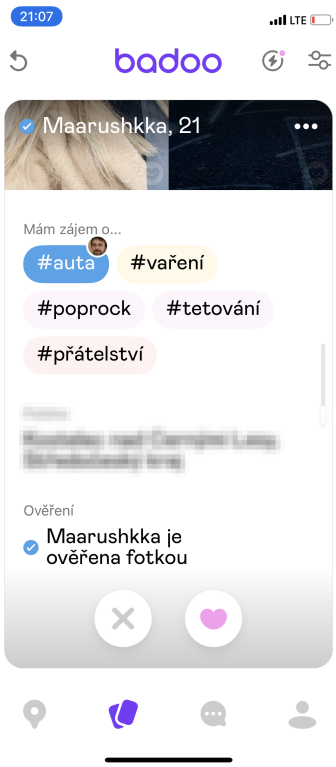
Počet profilů, které se uživateli zobrazují na hlavní stránce, lze navýšit zakoupením předplatného. Zpoplatněné jsou i funkce umožňující vidět, kteří uživatelé si zobrazili náš profil, nebo filtrování výsledků na hlavní stránce (obrázek 3.10).

3.4 Badoo

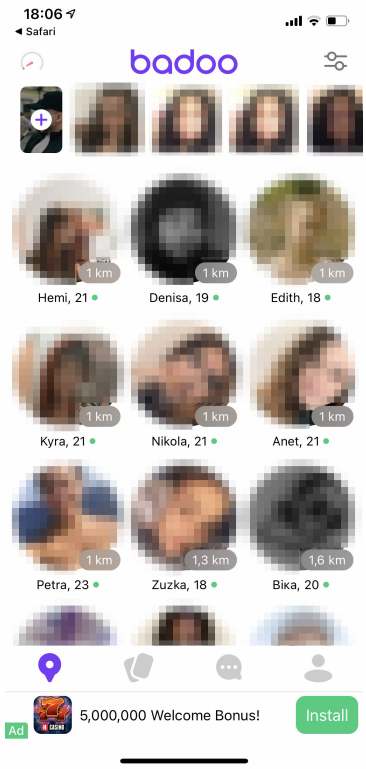
Přibližně stejně populární aplikací, jako je Tinder, je Badoo. Tato aplikace se rozšířila převážně v Evropě a kromě toho, že se můžete spojovat se svými protějšky, nabízí také možnost napsat uživatelům v okolí, kteří si chtějí povídat (jedná se o typ 1).

Pro registraci na této seznamce je potřeba zadat jméno, nahrát profilovou fotku a zvolit datum narození. V dalším kroce si uživatel vybere svou sexuální orientaci a záliby, čímž se znatelně odlišuje od ostatních výše zmíněných aplikací. Tyto záliby a zájmy poté uživatel vidí na profilu protějšku (obrázek 3.11) a může tak snadněji navázat konverzaci. Kromě zálib a zájmů může uživatel specifikovat i svou výšku, jakou školu navštěvoval, kde pracuje a nebo své názory na tři otázky, které Badoo předem nabídne.

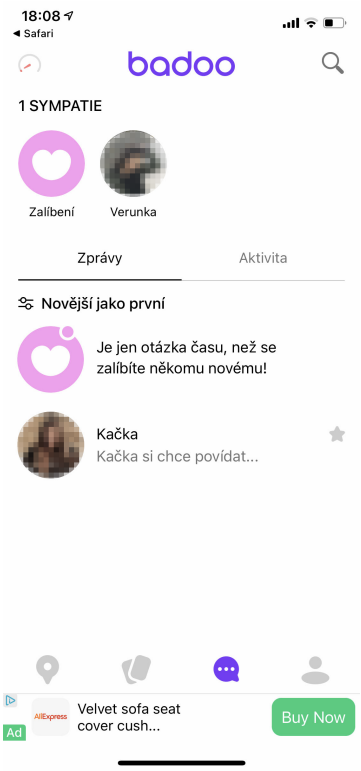
Jak jsem již zmínil, uživatel má kromě možnosti spojovat se s jinými uživateli tak, jak je to na jiných seznamkách, také možnost napsat přímo uživatelům v jeho okolí - aniž by se s někým musel spojovat. Tuto funkci může uživatel použít ovšem pouze třikrát, poté je po uživateli vyžadováno dobití kreditů.



Obrázek 3.11: Zájmy na profilu protějšku. Obrázek byl anonymizován.



Obrázek 3.12: Možnost psát uživatelům v okolí na Badoo. Obrázek byl anonymizován.



Obrázek 3.13: Vzhled výběrů chatu na Badoo. Obrázek byl anonymizován.

Kapitola 4

Vývoj multiplatformních aplikací pro mobilní zařízení

Tato kapitola pojednává o současných možnostech vývoje multiplatformní aplikací[1]. Postupně jsou zde rozebrány jednotlivé typy multiplatformních aplikací. U každého typu jsou vysvětleny výhody a nevýhody použití daného přístupu a motivace pro vývoj v dané architektuře.

Před samotným vývojem multiplatformní aplikace je nutné zvážit, zda architektura a způsob vývoje, který volíme je skutečně vhodný pro naše použití. Tyto aplikace se mohou tvořit mnoha způsoby. Můžeme například vyvíjet nativní aplikace, progresivní webové aplikace, nebo multiplatformní nativní aplikace. Každý z těchto přístupů má svá pro i proti. Mezi klíčové ukazatele, zda daný přístup vyhovuje našim požadavkům patří:

- doba vývoje,
- cena vývoje,
- liské zdroje,
- cena údržby,
- výkon aplikace,
- dostupnost aplikačních rozhraní,
- knihovny třetích stran.

V podsekcích této kapitoly je popsána základní charakteristika každého z přístupů, uvedeny jednotlivé příklady možných frameworků[2], či nástrojů pro vývoj a vztah vzhledem k požadavkům zmínených výše.

4.1 Nativní aplikace

První ze způsobů, jak můžeme vytvořit multiplatformní aplikaci, je vytvořit danou aplikaci vícekrát – pro každou platformu vytvoříme jednu aplikaci. Pro vývoj nativních aplikací

[1] Aplikace spustitelná na více platformách

[2] Softwarová struktura, která definuje pomocné nástroje, knihovny, nebo postupy pro vývoj software

jsou typicky používány nástroje vydané přímo vývojářem dané platformy. Pro platformu Android vyvíjí společnost Google IDE[3] Android SDK[4], kde programujeme v jazyce Kotlin, nebo Java. Vývoj pro veškeré platformy od Apple (tvOS, iOS, MacOS) je možný z prostředí xCode[5], kde programujeme v jazyce Objective C, nebo Swift. Takto vytvořené aplikace jsou jednoduše publikovatelné v obchodech Google Play a App store.

Zvolením tohoto přístupu máme k dispozici nativní prvky uživatelského rozhraní (tlačítko, klávesnice, komponenta pro výběr datumu a času) a veškeré prvky aplikačního rozhraní (notifikace, geolokace, stav baterie, vibrace apod.), které daná platforma zpřístupňuje. Výhodou je, že pokud jednotlivé prvky UI[6] použijeme ve své aplikaci a vývojář dané platformy aktualizuje vzhled uživatelského rozhraní (stalo se například při přechodu z iOS 13 na iOS 14 [10]), stačí náš stávající kód pouze překompilovat pro novější verzi platformy. Tímto se veškeré komponenty aktualizují a získají nový vzhled. Tento způsob aktualizace aplikace pro novější verzi je možný díky tomu, že vývojáři platforem s novou verzí typicky vydávají také aktualizace pro vývojářské nástroje. Další výhodou je výkon aplikace. Jelikož aplikace využívá přímo prvky a knihovny vydané vývojářem, má přímou podporu na zařízeních, tudíž je rychlejší. Rovněž nám vývoj pro jednu specifickou platformu umožňuje předcházet těžko identifikovatelným chybám aplikace.

Nevýhody, které s sebou vývoj nativních aplikací přináší, jsou velice často spojeny s financemi, které jsou potřeba pro jejich vývoj, rozvoj či údržbu. Vyvíjet jednu aplikaci pro více platforem nativně je velmi drahá záležitost vzhledem k lidským zdrojům. Zjednodušeně řečeno, pokud bychom chtěli vytvořit aplikaci pro iOS a Android zároveň a stačilo by najmout jednoho vývojáře pro každou platformu na jeden rok, náklady na vývojáře by byly přibližně 200 000 dolarů – podle dat ze serveru PayScale[7] pro rok 2021 je průměrná mzda vývojáře iOS aplikací v USA 104 624 dolarů ročně a mzda vývojáře Android aplikací 94 365 dolarů ročně. Vzhledem k tomu, že musíme mít jednu aplikaci pro každou platformu, se prodražuje také údržba. Pokud například chceme vyvinout novou vlastnost aplikace, musíme ji implementovat zvlášť pro každou z cílových platforem. Další nevýhodou je čas. Vývoj a údržba těchto aplikací trvá typicky delší dobu – právě proto, že je třeba vývoj a údržba zvlášť pro každou z cílových platforem.

4.2 Progresivní webové aplikace

Zjednodušeně se dá říct, že progresivní webové aplikace (zkráceně PWA) jsou webové aplikace, které jsou uzpůsobeny k tomu, aby byly spuštěny lokálně na zařízení uživatele ve webovém prohlížeči. PWA obsahují tzv. *service worker*, který pro aplikaci zajišťuje caching[8], synchronizaci na pozadí a naslouchání notifikacím. PWA se dají naprogramovat se stejnými nástroji jako klasické webové stránky – používáme tedy HTML, CSS a JavaScript. Publikovat je můžeme na Microsoft Store a Google Play. Nikoliv však na App store. Platforma iOS nezpřístupňuje tomuto typu aplikací své aplikační rozhraní, tudíž nedává smysl takové aplikace publikovat v obchodech (což samotný Apple v konečném důsledku nedovoluje) [31].

[3] Integrované vývojové prostředí
[4] https://developer.android.com/studio
[5] https://developer.apple.com/xcode/
[6] Zkráceně *User interface* – uživatelské rozhraní
[7] https://payscale.com
[8] Uchovávání dat na zařízení uživatele pro rychlejší a offline přístup

Výhodou je, že právě díky *service worker* jsou tyto aplikace spustitelné offline. Díky tomu, že využívají stejných technologií, jako weby, můžeme při vývoji ušetřit na lidských zdrojích. Nemusíme totiž programovat aplikaci pro jednotlivé platformy, ale stačí aplikaci napsat jednou. Další výhodou je čas potřebný pro vývoj aplikace. Takto vyvinutou aplikaci můžeme spustit na všech cílových platformách, které mají webový prohlížeč. Můžeme tedy říct, že pro samotný vývoj aplikace ze sekce 4.1 stačí jeden webový vývojář (podle dat ze serveru PayScale je mzda vývojáře webových aplikací v USA pro rok 2021 průměrně 83 425 dolarů ročně). Pro tvorbu PWA existuje několik frameworků. Mezi nejpoužívanějšími jsou React[9] a Angular[10]. Dále se používají Vue[11], Ionic[12] nebo Polymer[13]. Co se údržby takové aplikace týče, je snazší než u nativních aplikací, jelikož spravujeme pouze jednu kódovou základnu.

Nevýhodou je, že se musí naprogramovat veškeré komponenty, které jsou v aplikaci použity. V případě, že dojde ke změně uživatelského rozhraní na cílové platformě, musí se přepisovat také komponenty aplikace. Další nevýhodou je přístupnost aplikačních rozhraní cílových platforem. Jelikož se nejedná o nativní aplikaci, není dostupné aplikační rozhraní například pro ovládání vibrací, získání stavu baterie, rozšířenou realitu, nebo geofencing. Samotná odezva aplikace je potom pomalejší, než v případě nativního řešení.

4.3 Multiplatformní nativní aplikace

Jedná se o nativní aplikace, které mají stejnou kódovou základnu, ale jsou přeložitelné do vícero platforem. K tvorbě takových aplikací se používají zejména frameworky React Native[14] (od Facebooku), Xamarin[15] (od Microsoftu) a relativně nový framework Flutter[16] od Googlu. Pokud si vybereme vyvíjet aplikaci v React Native, budeme vyvíjet v jazyce Javascript. U Xamarinu je to C# a u Flutteru relativně nový jazyk Dart[17], který stejně jako Flutter vyvíjí Google. Tyto aplikace jsou publikovatelné v Google Play, App store i Microsoft store.

Jelikož můžeme vyvíjet jednu kódovu základnu a kompilovat ji pro všechny podporované platformy, je vývoj aplikace rychlý. To stejné platí i v případě údržby kódu, či vytváření nových funkcí aplikace. Aplikace mají k dispozici nativní komponenty cílových platforem, které lze použít. Zde je nutno dodat, že ne vždy jsou všechny komponenty plně podporované. Existují komponenty – například *DatePicker*, u kterých jsou potřeba malé úpravy, aby kopírovaly nativní vzhled. Výhodou je i rychlost takto vytvořené aplikace. Jelikož je aplikace přeložena jako nativní aplikace, je rychlost srovnatelná s aplikacemi ze sekce 4.1. Další výhodou je, že oproti PWA (ze sekce 4.2) nabízejí vývojáři plnou podporu aplikačních rozhraní. Vývojář má tedy dostupné API[18] pro stav baterie, vibrace nebo rozšířenou realitu.

Co se nevýhod týče, může se zdát, že se jedná o ideální řešení. Při vývoji multiplatformní aplikace tímto způsobem ovšem můžeme narazit na problémy spojené s odlišným zobrazením jednotlivých komponent napříč platformami. Takové chování není vždy žádané

[9] https://reactjs.org/
[10] https://angular.io/
[11] https://vuejs.org/
[12] https://ionicframework.com/
[13] https://www.polymer-project.org/
[14] https://reactnative.dev/
[15] https://dotnet.microsoft.com/apps/xamarin
[16] https://flutter.dev/
[17] https://dart.dev/
[18] Zkráceně „Application programming interface“ – tedy aplikační rozhraní

– například u tlačítek se může jednat o správné chování, u zobrazování vlastních definovaných komponent tomu tak již být nemusí.

4.4 Životní cyklus vývoje aplikace

4.4.1 Analýza

Prvním krokem při vývoji multiplatformních aplikací je analýza. Analýza má za cíl analyzovat trh, identifikovat cílovou skupinu, zjistit požadavky uživatelů a na základě těch definovat persóny. Rovněž by měla obsahovat, zda je daný problém řešitelný. V kontextu analýzy je persóna fiktivní osoba, která vznikla agregací výsledků dotazníkového šetření, či jiného typu průzkumu [5]. Díky persónám lze definovat požadavky na danou aplikaci z pohledu jednotlivých persón. Těmito požadavky mohou být:

- **Cílové platformy** – má potenciální uživatel telefon s iOS, nebo Android?
- **Definice potřeb** – jaké funkce by aplikace měla splňovat?
- **Požadavky na soukromí** – vyžaduje po mě aplikace data, které nepotřebuje ke své funkčnosti?
- **Požadavky na design** – je aplikace intuitivní, je práce s ní jednoduchá?
- **Požadavky na lokalizaci** – je aplikace lokalizována do jazyka, kterému rozumím?
- **Požadavky na náročnost aplikace** – je aplikace jednoduše spustitelná na mém zařízení bez jakýchkoliv znatelných problémů s výkonem?

4.4.2 Návrh

Na základě persón vytvořených v podsekci 4.4.1 můžeme navrhnout řešení, které splňuje požadavky jednotlivých persón. Součástí návrhu je definice samotné funkcionality aplikace, návrh grafického uživatelského rozhraní a zpětná validace navrženého řešení vůči požadavkům, které vznikly analýzou. Navržená aplikace by měla být pokud možno co nejvíce škálovatelná s ohledem na možné často měnící se potřeby koncových uživatelů.

Při návrhu se používají nástroje jako *entity relationship diagram*, *data flow diagram* nebo *use case diagram*. *Entity relationship diagram* (zkráceně ERD) je nástroj pro charakteristiku struktury dat (entit) a závislostí mezi nimi [14]. Používá se například pro znázornění struktury dat v databázi. *Data flow diagram* (diagram datových toků) je nástroj, který se používá pro popis funkce systému, který zachycuje transformace dat a vstupy, či výstupy jednotlivých částí systému [13]. Kromě popisu funkce systému se používá například i jako nástroj pro strategické plánování. *Use case diagram* (diagram případů užití) je nástroj, jenž zobrazuje funkce systému z aktérovy perspektivy [2]. Aktérem může být samotný uživatel, či jiný prvek systému. Pro návrh jednotlivých algoritmických problémů je typické použití vývojových diagramů. Ty znázorňují jakým způsobem se algoritmus v jednotlivých případech rozhoduje a jaké vykonává dílčí úlohy [20].

4.4.3 Implementace

Dalším krokem je samotná implementace návrhu z podsekce 4.4.2. Při implementaci se řeší konkrétní problémy jak z návrhu udělat funkční, spustitelnou aplikaci. V této fázi vybíráme

vhodné programovací jazyky a implementujeme navrženou architekturu. Při implementaci by se mělo kromě verifikace, zda aplikace splňuje všechny požadavky návrhu, dbát i na rozšířitelnost aplikace a její následnou údržbu.

V zásadě existují dva způsoby implementace. Aplikace lze implementovat zdola-nahoru, či shora-dolů [25]. Při implementaci zdola-nahoru jsou nejdříve implementovány dílčí části aplikace na nižší úrovni. Tyto části postupně napojujeme na sebe, čímž vzniká systém. U přístupu shora-dolů nejdříve implementujeme komponenty a moduly na vyšších vrstvách aplikace, přičemž postupně implementujeme dílčí části aplikace na nižších vrstvách. Mezi typické problémy, které je nutné při implementaci řešit, patří například integrita dat, bezpečnost, přenos dat či dostupnost zdrojů.

4.4.4 Testování

Závěrem celého cyklu vývoje aplikace je testování. Testování má za cíl odhalit chyby v programu, které nebylo možné identifikovat v průběhu vývoje. Zároveň má za cíl ověřit, že výsledný program funguje tak, jak byl navržen. Je vhodné výslednou aplikaci testovat na vícero zařízení s různými parametry. Zároveň je vhodné implementovat testy na úrovni aplikace tak, abychom případné chyby ve funkcionalitě byli schopni odhalit dříve, než se aplikace dostane do produkčního prostředí.

Na úrovni aplikace můžeme implementovat tři typy testů: *unit tests* (jednotkové testy), *integration tests* (integrační testy) a *end-to-end tests* [15]. Jednotkové testy se implementují na úrovni jednotlivých funkcí. Můžeme tak ověřit, že například funkce, která slouží pro výpočet obsahu obdelníku, generuje korektní výsledky. Pro implementaci jednotkových testů se používají například frameworky Mocha.js[19] nebo Jest[20]. Úlohou integračních testů je zjistit, zda kombinace jednotlivých částí kódu funguje korektně. Může se jednat například o dotazování na koncových bodech REST API. Integrační testy lze rovněž psát pomocí frameworků Jest a Mocha.js. *End-to-end tests* slouží pro testování funkcionality aplikace z pohledu uživatele. Pro takový typ testování se u webových aplikací používá například Cypress[21], nebo Puppeteer[22].

[19] https://mochajs.org/
[20] https://jestjs.io/
[21] https://www.cypress.io/
[22] https://pptr.dev/

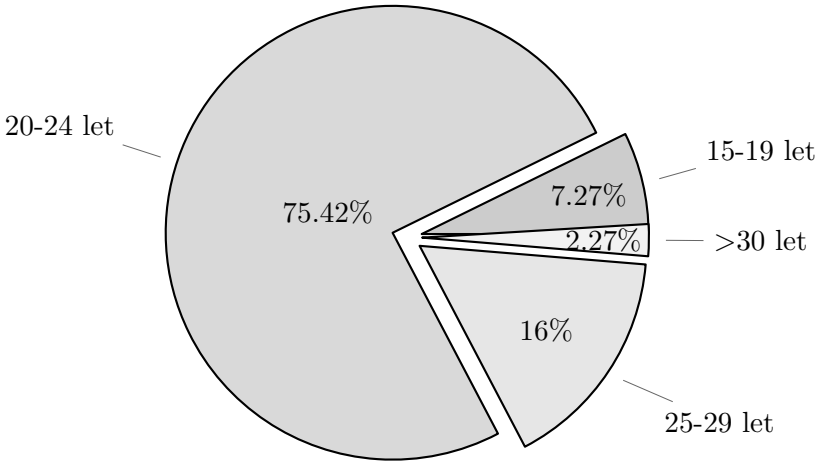
Kapitola 5

Analýza problému

V této kapitole je obsažena analýza cílové skupiny uživatelů, jejich potřeb a požadavků. Samotná analýza byla provedena na výsledcích dotazníkového šetření ze vzorku 484 respondentů. Výsledky anonymního dotazníkového šetření jsou k dispozici na přiloženém médiu. V kapitole je postupně detailně rozebrán vzorek respondentů, vytvoření persón na základě dotazníkového šetření, identifikace styčných bodů aplikace a definice problému.

5.1 Dotazníkové šetření

Z 484 respondentů bylo 482 sad odpovědí validních. Zbylé dvě byly vyloučeny z důvodu, že respondenti uvedli věk, který neodpovídal horní hranici lidského života. Analyzován je tedy vzorek 482 respondentů.

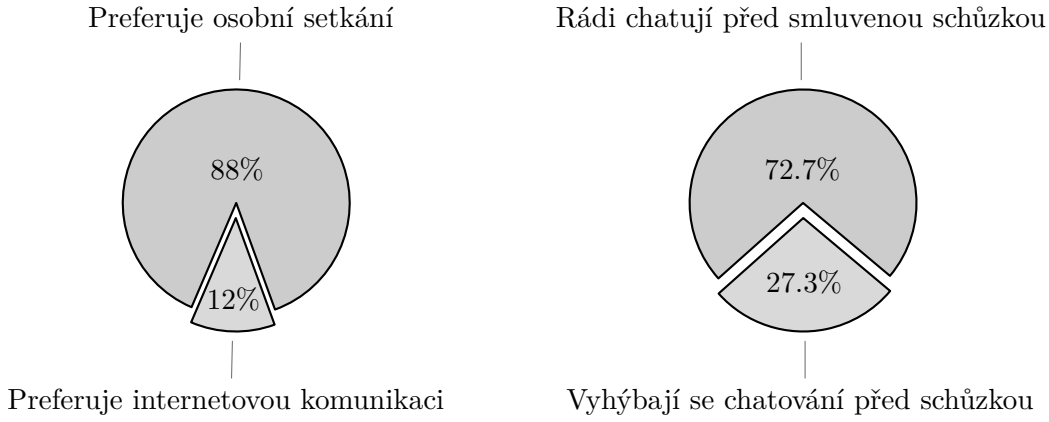


Obrázek 5.1: Koláčový graf znázorňující věk respondentů

V koláčovém grafu 5.1 je vyobrazeno věkové rozložení respondentů. Téměř 92% respondentů bylo ve věku 20-30 let. Dále můžeme z výsledků dotazníku vyčíst následující informace:

- Podíl respondentů s telefony obsahující operační systém **iOS:Android** je **33,1 % ku 66,9 %**,
- **53,9 %** respondentů jsou **muži**, **46,1 %** jsou **ženy**,
- **84,9 %** respondentů **nenavštěvuje** internetovou seznamku, ani nepoužívá aplikace jako Tinder 3.1, nebo Badoo 3.4,

Dotazník poté obsahoval otázky „Preferujete osobní setkání před komunikací na internetu?“ a „Baví vás před smluvenou schůzkou chatovat?“ (grafické znázornění odpovědí nalezneme na obrázku 5.2). Zde se v odpovědích dostáváme do sporu. Tento spor je vhodné detailně rozebrat.



Obrázek 5.2: Koláčové grafy znázorňující preferenci respondentů v osobním seznamování oproti seznamování na internetu

Z detailní analýzy dat vyplývá, že z celkového počtu 482 respondentů jich 308 preferuje osobní setkání, přičemž rádi chatují před smluvenou schůzkou. Pouze 44 (14.2 %) z nich však používá nějaký typ seznamovací aplikace. Z tohoto můžeme vyvodit, že zbylých 85.8 % používají příležitostně jiné rozšířené platformy, jako například Instagram[1], nebo Messenger[2].

Respondenti měli možnost vyjádřit se slovně a dodat komentář k jejich odpovědím. Zajímavé jsou komentáře respondentů, kteří odpověděli, že upřednostňují osobní kontakt a nevyužívají žádné seznamovací aplikace. Respondenti velice často jako důvod zmiňují, že uživatelé současných seznamek se velice často přetvařují. Rovněž svou volbu zdůvodňují tím, že seznamování přes seznamky je nepřirozené, či neosobní. „*Je to jako se vystavovat ve výloze.*“, „*Člověk je jak zboží v obchodě.*“ – Anonymní respondenti. Z komentářů přímo vyplývá, že lidem může vadit souzení na základě vnějšího vzhledu. Toto se týká všech aplikacích zmíněných v kapitole 3. Respondenti mužského pohlaví rovněž zmínili, že pokud je žena atraktivní, může mít i několik set žádostí o zprávy. V takovém případě je velice malá pravděpodobnost, že odpoví všem. Tento konkrétní problém je obvyklý na platformách 3.1 a 3.4. Jeden konkrétní uživatel se může ztratit v „moři“ ostatních žádostí o zprávy. To může mít zásadní vliv na konverzi chat $\Longrightarrow$ schůzka.

Dalším často zmíněným problémem je velké zmožství zpoplatněných funkcí. Respondentům velice často vadí, že zakoupením předplatného uživatelé nabydou podstatné výhody,

[1] https://instagram.com

která může být například v případě aplikace Tinder 3.1 rozhodující. Rovněž určité podmnožině respondentů vadí sexuální podtext a poměr uživatelů, kteří vyhledávají sexuální služby – toto se dá opět říct obecně o všech aplikacích z kapitoly 3. Část respondentů zmiňuje, že při používání těchto aplikací postrádají právě faktory zmíněné v podsekci 2.2.1 a to sice úsměvy, emoce nebo intonaci hlasu.

5.2 Persóny

Na základě agregace výsledků byly vytvořeny persóny. Tyto persóny reprezentují jednotlivce z cílové skupiny. Termínem cílová skupina uživatelů se rozumí podmnožina veřejnosti, na kterou bude produkt cílit [6]. V tomto případě jsou cílovou skupinou uživatelů primárně osoby ve věku 18-35 let, kterým vadí neosobní seznamovací aplikace. Může se jednat o studenty, pracující lidi, nebo rodiče. Cílový zákazník musí mít k dispoziti mobilní telefon s operačním systémem Android nebo iOS. Obrázky vytvořených persón byly vygenerovány pomocí ThisPersonDoesNotExist.com[3].

- věk: 26 let,
- pracující muž,
- nepoužívá seznamovací aplikace, protože raději někoho potká osobně,
- vadí mu povrchnost a přetvařující se uživatelé

Obrázek 5.3: Persóna – Tomáš

- věk: 24 let,
- studentka,
- rádá používá seznamovací aplikace, ale vadí ji sexuální podtext v konverzacích
- je stydlivá, ale uvítala by možnost potkat protějšek osobně

Obrázek 5.4: Persóna – Natálie

5.3 Závěr analýzy

Veškeré aplikace zmíněné v kapitole 3 nejsou dostatečně osobní. Bylo žjištěno, že mnoho respondentů dotazníkového šetření nevyužívá tyto služby právě kvůli neosobnímu kontaktu. Uživatelům seznamovacích aplikací rovněž vadí, že hodnocení protějšku v současně existujících aplikací je založeno na vnější kráse. Ačkoliv například Badoo 3.4 a Bumble 3.2 nabízí uživateli možnost vložit na profil detaily o svých zájmech a koníčcích, nejeví se to jako dostačující. Vzniká zde potřeba sofistikovaně vytvářet spojení s důrazem na vnitřní krásu. Nabízí se tedy možnost využití testů pro zjištění osobnostních typů.

Ukázalo se, že respondenti by se rádi seznamovali osobně, ale současné aplikace jim to nenabízí. Rovněž se ukázalo, že respondenti si rádi píšou před domluvenou schůzkou,

[3] https://thispersondoesnotexist.com/

přičemž ale neradi tráví psaním příliš mnoho času. Také by ocenili, kdyby jejich přítomnost v nabízených shodách byla hodnotnější. Zdůrazníme-li tento fakt, můžeme říct, že na současných platformách je přítomnost uživatele v nabízených shodách zanedbatelná, což uživatele mnohdy odrazuje od dalšího používání aplikace.

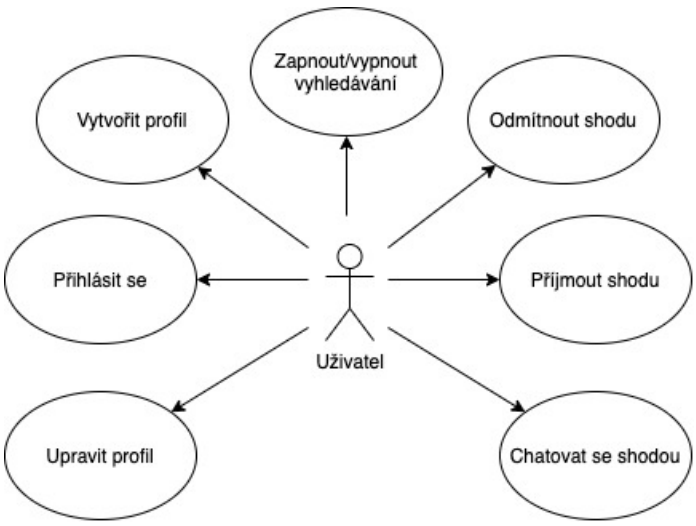
Kapitola 6

Návrh řešení

Obsahem této kapitoly je návrh řešení, které napraví nedostatky současně dostupných aplikací, které vyplynuly z kapitoly 5. Navrhovaným řešením je vytvořit novou mobilní multiplatformní aplikaci. Postupně je zde rozebrána navrhovaná funkcionalita, samotný technologický návrh a nastínění vhodné architektury. Obsažen je rovněž diagram případů užití, návrh grafického uživatelského rozhraní a návrh datového modelu.

6.1 Návrh funkcionality

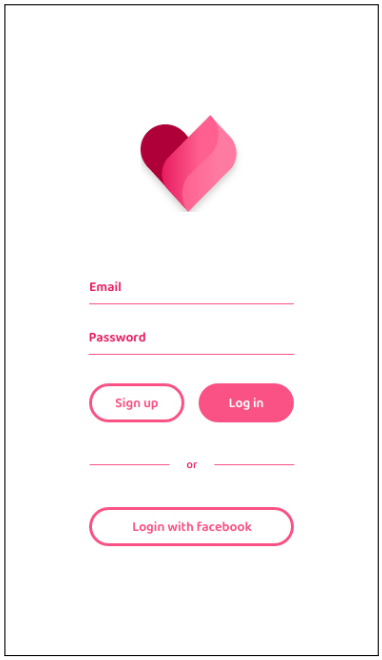
V diagramu případů užití (obrázek 6.1) je vyobrazen způsob, jakým bude uživatel s aplikací interagovat. Je potřeba, aby uživatel měl na platformě vlastní profil. Tudíž musí mít možnost se zaregistrovat a přihlásit. Rovněž je vhodné, aby uživatel měl možnost upravit profil v případě, že dojde k omylu při jeho vyplňování. Pokud je uživateli nabídnuta shoda, musí mít možnost ji odmítnout, či příjmout. Pokud se rozhodne shodu příjmout, bude mu umožněno chatovat s protějškem. Rovněž se můžou vyskytnout případy, kdy uživatel nebude mít zájem o vyhledávání nových spojení. Uživateli tedy musí být zpřístupněna možnost vypnout vyhledávání.



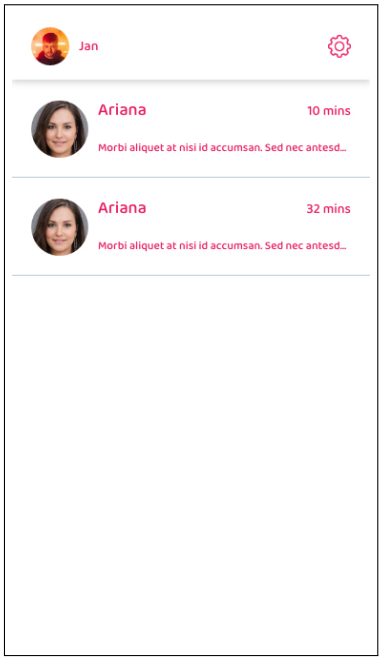
Obrázek 6.1: Diagram případů užití

Na úvodním pohledu aplikace (obrázek 6.2) se uživatel má možnost registrovat a přihlásit. Přihlášení je možné prostřednictvím e-mailu, nebo facebooku. Při registraci je uživatel vyzván k zadání svého pohlaví, jména a data narození. Následuje výběr pohlaví, o které má zájem a nahrání fotografie. Poté se uživateli zobrazí hlavní pohled, na kterém uvidí v případě shody chat (obrázek 6.3).

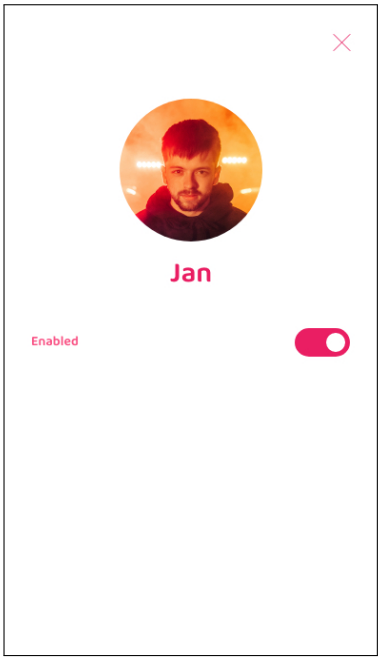
Po úspěšné registraci je tedy aplikace okamžitě použitelná. Pokud ovšem uživatel chce zpřesnit výsledky, které mu algoritmus nabídne a dostávat tak relevantnější nabídku potenciálních partnerů, může zadat typ své osobnosti podle *Myers-Briggs type indicator*, který byl popsán v sekci 2.5. Ten je následně brán v potaz při vyhledávání. Tímto aplikace docílí naplnění požadavku o sofistikovanějším vyhledávání spojení ze sekce 5.3. K určení typu osobnosti může uživatel využít například webovou aplikaci 16personalities[1]. Samotná aplikace zadání osobnostního typu nevyžaduje a funguje i bez něj.



Obrázek 6.2: Přihlašovací stránka aplikace



Obrázek 6.3: Hlavní stránka aplikace s chaty

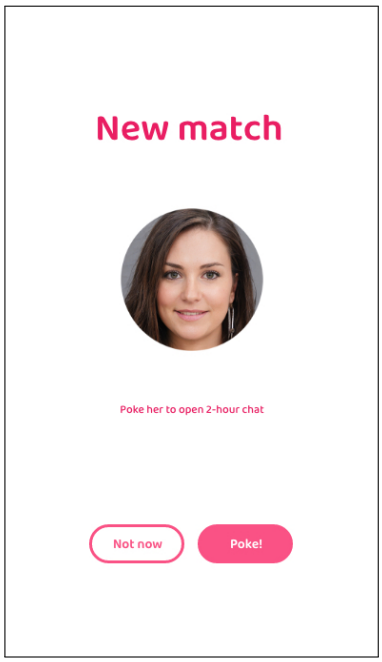


Obrázek 6.4: Nastavení profilu v aplikaci

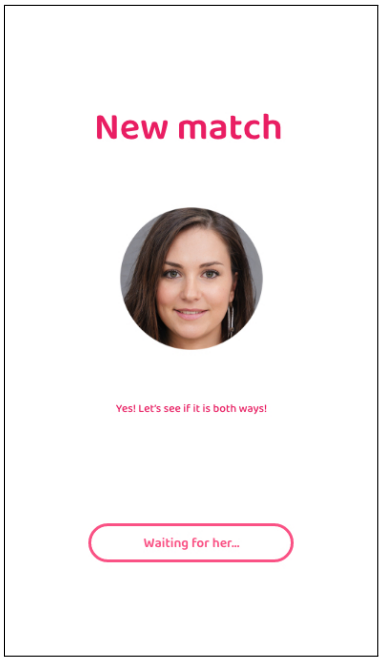
Jelikož by se respondenti z kapitoly 5 rádi seznamovali osobně, bylo vhodné navrhnout aplikaci tak, aby na ní uživatelé netrávili příliš mnoho času chatováním a co nejdříve se sešli. Navržená aplikace tedy funguje tak, že po vyplnění vstupních informací využívá technologií určení polohy zařízení k vyhledání potenciálních partnerů v okolí. Partneři jsou vyhledáváni v okruhu 1000 metrů. V případě, že je partner v okolí nalezen (tedy vyhovuje kritériím), zobrazí se uživateli notifikace, která odkáže oba účastníky na pohled s novou shodou (obrázek 6.5). Na tomto pohledu musí oba z účastníků svolit k tomu, že spolu chtějí chatovat. V tomto bodě je oběma účastníkům ukázána pouze fotografie protějšku, se kterým mají možnost se setkat. Díky tomu se budou muset uživatelé v případě shody sami představit a předpokládá se, že diskuze bude přirozenější.

Aby se zamezilo nadměrnému počtu možných propojení a uživatel se tak neztratil ve velkém množství uživatelů, aplikace generuje propojení s minimálním odstupem 4 hodin.

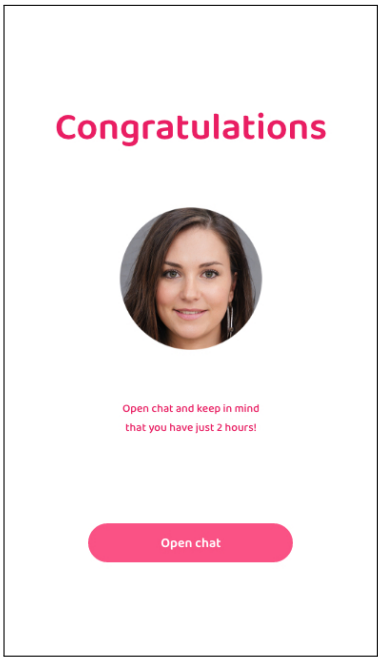
[1] https://www.16personalities.com/



Obrázek 6.5: Pohled s novou shodou



Obrázek 6.6: Čekání na potvrzení shody protějškem



Obrázek 6.7: Nabídka chatování po vzájemné shodě

Po vstupu do chatu mají uživatele dvě hodiny na to, aby si spolu psali. Samotný chat by měl co nejrychleji vyeskalovat k pozvání na schůzku. Jelikož by v momentě shody měli být oba uživatelé relativně blízko u sebe, může ke schůzce dojít okamžitě. Zde aplikace řeší problém neosobního seznamování, který byl zdůrazněn v kapitole 5. Po uplynutí dvou hodin se chat nenávratně vymaže z uživatelovy hlavní obrazovky 6.3 a nelze již nikdy vytvořit stejnou shodu. Pokud uživatel nechce vyhledávat spojení a nechce být zahrnut do vyhledávání spojení jiných uživatelů, může vypnout vyhledávání v menu aplikace (obrázek 6.4).

Takto navržená aplikace vyhovuje persónám ze sekce 5.2. Aplikace navádí uživatele k tomu, aby se seznamovali osobně, čímž zároveň částečně eliminuje povrchnost a přetvařování se.

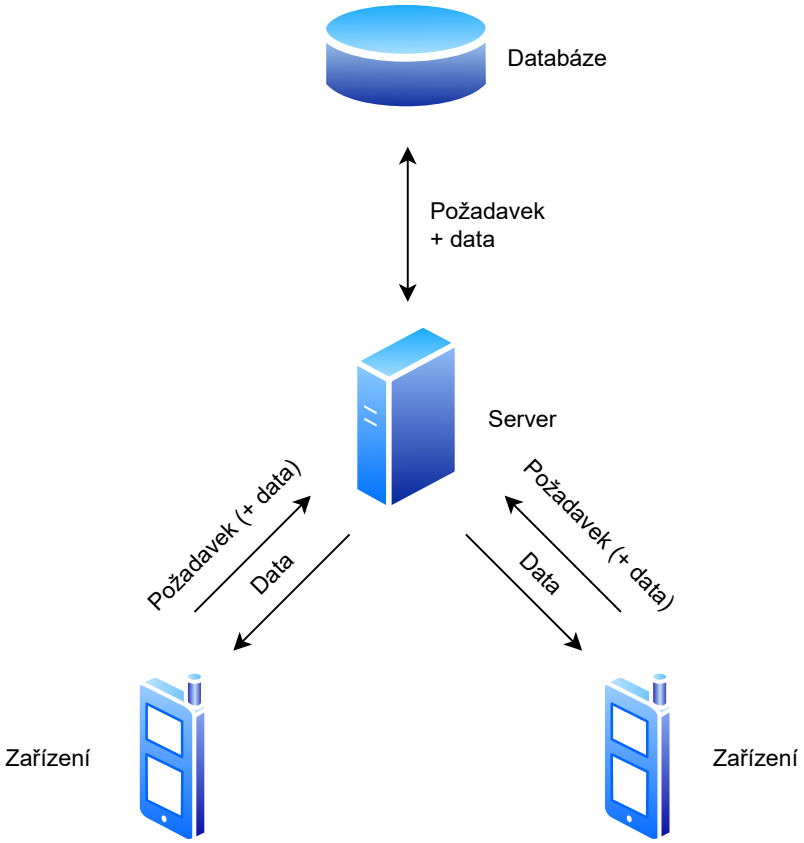
6.2 Technologický návrh

Protože má dnes u sebe téměř každý svůj chytrý mobilní telefon, je vhodné, aby byla vytvořena mobilní aplikace, která bude dostupná pro nejpoužívanější platformy – v současnosti iOS a Android. Po zvážení všech klíčových kritérií z kapitoly 4 byl jako nejvhodnější způsob tvorby multiplatformní aplikace pro tento účel zvolen vývoj multiplatformní nativní aplikace 4.3.

Jelikož aplikace bude multiplatformní a je pravděpodobné, že by v budoucnu mohlo vzniknout webové rozhraní, dává smysl co nejvíce separovat samotnou logiku aplikace od zobrazovací vrstvy. Nejvhodnější architekturou pro takový typ aplikace je třívrstvá architektura[2]. Díky ní bude možné vytvořit více implementací prezentační vrstvy, které budou

[2]Architektura oddělující 3 základní vrstvy aplikace – prezentační vrstvu, aplikační vrstvu a datovou vrstvu. Zodpovědností prezentační vrstvy je zobrazovat data z aplikační vrstvy. Aplikační vrstva, kde probí-

používat stejnou aplikační vrstvu (obrázek 6.8). Tedy bude možné vytvořit jak samotnou aplikaci, tak i později webové rozhraní.



Obrázek 6.8: Návrh architektury aplikace

Jak již bylo zmíněno v sekci 6.1, spojení, které v aplikaci vzniknou, budou po určitém časovém intervalu zanikat. Toho lze docílit více způsoby. Lze vytvořit skript, jehož účelem by bylo mazat expirované propojení. Tento způsob ovšem může být výpočetně náročný. Skript by musel každou minutu iterovat skrz všechny uživatele a ověřovat, zda jsou propojení stále validní. Při větším počtu uživatelů by tedy mohlo dojít k výrazně delší době výpočtu, která by ve výsledku mohla být větší, než samotný interval mezi jednotlivými spuštěními skriptu. Jako druhé řešení takového problému se nabízí volání metod aplikačního rozhraní v momentě, kdy uživatel chce interagovat s propojením. Tímto způsobem nedojde k zbytečnému vytížení serveru a k ověření, zda je propojení platné, dojde vždy pouze když to bude potřeba.

6.2.1 Model dat

Uživatel

Pro správnou funkčnost aplikace musí aplikace uchovávat o uživateli dostatečné množství dat. Mezi tyto data patří kromě základních informaci, jako je jméno uživatele, e-mail, heslo

hají samotné výpočty a hlavní logika aplikace, poté komunikuje s datovou vrstvou, která uchovává samotná data a může implementovat určité operace nad nimi [4]

a datum narození také informace o poslední známé poloze, token, který bude použit pro zasílání notifikací a záznam již uskutečněných propojení.

Jednotlivé atributy modelu a jejich význam:

- **accepted matches** – spojení, které uživatel přijal,
- **avatar URL** – adresa vedoucí k profilovému obrázku uživatele,
- **birthday** – datum narození uživatele,
- **conversations** – konverzace, ve kterých uživatel je (jedná se o konverzace s aktivními spojeními),
- **coordinates** – aktuální poloha uživatele,
- **deleted** – informace, zda byl uživatelům profil smazán,
- **id** – unikátní identifikátor uživatele,
- **interested in** – udržuje informaci o pohlaví protějšku, o které má uživatel zájem,
- **legal agreement** – informace, zda uživatel přijal podmínky užití aplikace,
- **matches** – seznam identifikátorů protějšků, se kterými již byla vytvořena shoda nezávisle na tom, zda byla přijata,
- **name** – jméno uživatele,
- **sex** – pohlaví uživatele,
- **token** – token pro zasílání notifikací,
- **e-mail** – e-mail uživatele,
- **password** – heslo k uživatelskému účtu

6.2.2 Zpráva

Data, které bude třeba uchovávat v případě odeslané zprávy jsou o poznání jednodušší. Atributy modelu pro zprávu a jejich význam:

- **date** – Datum odeslání zprávy,
- **message** – Samotný obsah zprávy,
- **from** – ID uživatele, jež zprávu odeslal,
- **to** – ID uživatele, jemuž byla zpráva odeslána

Kapitola 7

Implementace

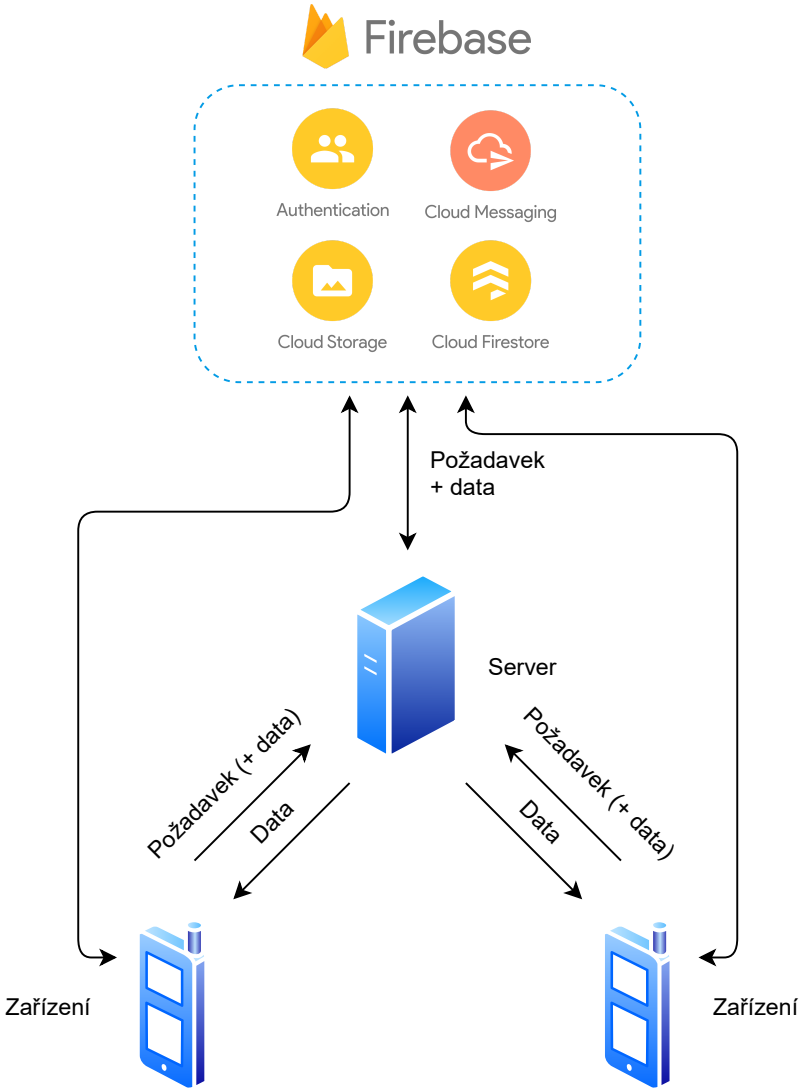
Tato kapitola pojednává o způsobu, jakým byla docílena realizace samotného návrhu z kapitoly 6. Postupně do detailu rozebírá naimplementovanou architekturu aplikace, implementaci backendu a implementaci frontendu. Obsažen je i popis algoritmu, jenž vybírá vhodné protějšky. Backend i aplikace byly implementovány cestou shora-dolů, která byla zmíněna v podsekci 4.4.3.

7.1 Architektura aplikace

Architekturu aplikace, která byla navržena v sekci 6.2 bylo třeba upravit tak, aby podporovala zasílání notifikací. Při průzkumu, kolik úsilí je třeba vynaložit pro vytvoření vlastního serveru, jehož odpovědností by bylo zasílat notifikace, jsem usoudil, že bude mnohem lepší použít již existující řešení. V současnosti jsou nejpoužívanější tyto tři řešení: AWS[1], Parse[2] a Firebase[3]. Z důvodu nižších výchozích zkušeností, a tudíž strmější učící křivky, jsem se rozhodl, že AWS nebude vhodné řešení. Parse je *self-hosted*, tudíž by bylo třeba vlastních serverů. Zvolil jsem tedy, že použiji řešení od Google – Firebase. Jedná se o BaaS[4], který poskytuje vývojáři nástroje například pro autentizaci uživatelů (kromě e-mailové autentizace podporuje také autentizaci pomocí sociálních sítí), Cloud messaging[5] a nerelační databázi. Rovněž pro něj existuje velmi dobře udržovaná a zdokumentová knihovna RNFirebase[6], která může být použita pro implementaci aplikace. Aplikační vrstva z obrázku 6.8 byla tedy rozšířena o nástroje, které poskytuje Firebase. Koncové zařízení ovšem stále komunikuje primárně se serverem, na kterém běží aplikační rozhraní. Výjimkou je, když uživatel využívá autentizaci, naslouchá notifikacím, nebo nahrává profilovou fotku.

Namísto samostatné databázové vrstvy (obrázek 7.1) byla použita služba Cloud Firestore. Použití DaaS[7] má řadu výhod – nejpodstatnější z nich je ovšem škálovatelnost řešení. Implementace databáze je detailně popsána v podsekci 7.2.2.

[1] https://aws.amazon.com/
[2] https://parseplatform.org/
[3] https://firebase.google.com/
[4] Backend as a service
[5] Zasílání zpráv a notifikací v aplikaci
[6] https://rnfirebase.io/
[7] *Database as a service*



Obrázek 7.1: Architektura implementované aplikace

7.2 Backend

Pro implementaci samotného backendu jsem zvolil Node.js[8], což je runtime prostředí pro JavaScript. Využívá asynchronní I/O operace, díky čemuž jsou řešení vytvořena v něm škálovatelná. Samotný Node.js obsahuje knihovny, které umožňují aplikaci v něm napsané, vytvořit HTTP[9] server. Takový HTTP server poté může zpracovávat HTTP požadavky a odpovídat na ně.

JavaScript je dynamicky typovaný jazyk. Pro větší projekty (které nejsou vyvíjeny jako prototyp) je tedy vhodné zvážit, zda není vhodnější využít statického typování. Díky statickému typování lze předejít mnohým *runtime* chybám a vývoj aplikace je pohodlnější. V takových případech se nabízí možnost využít nadstavbu pro JavaScript – TypeScript[10].

[8] https://nodejs.org/en/

[9] *Hypertext transfer protocol* – je protokol na aplikační vrstvě, jenž je navržen pro komunikaci mezi webovým prohlížečem a serverem.

[10] https://www.typescriptlang.org/

Kód napsaný v jazyce TypeScript se poté kompiluje do JavaScriptu a následně spouští v Node.js. Typová kontrola probíhá při kompilaci do JavaScriptu. V případě této práce bylo vhodné použít nádstavbu TypeScript, jelikož se jedná o poměrně komplexní aplikaci, ve které je zvýšené riziko možných runtime chyb.

Současný trend je implementovat aplikační vrstvu jako REST API, nebo GraphQL. Výhodou implementace pomocí GraphQL je automaticky generovaná dokumentace, nebo možnost žádat pouze o data, která skutečně potřebujeme. GraphQL zpřístuňuje pouze jeden koncový bod rozhraní, ve kterém klient specifikuje dotaz (*Query*). Naproti tomu REST API zpřístupňuje sadu koncových bodů, které implementují CRUD[11]. Můžeme říct, že zásadní rozdíl mezi GraphQL a REST API je, že u GraphQL specifikuje podobu odpovědi ten, kdo žádá o data. Naproti tomu u REST API nemá klient možnost ovlivnit strukturu odpovědi serveru. Nevýhodou GraphQL ale je, že takto navržené API vrací pokaždé HTTP stavový kód HTTP 200 a některé implementace dělá až příliš složité.

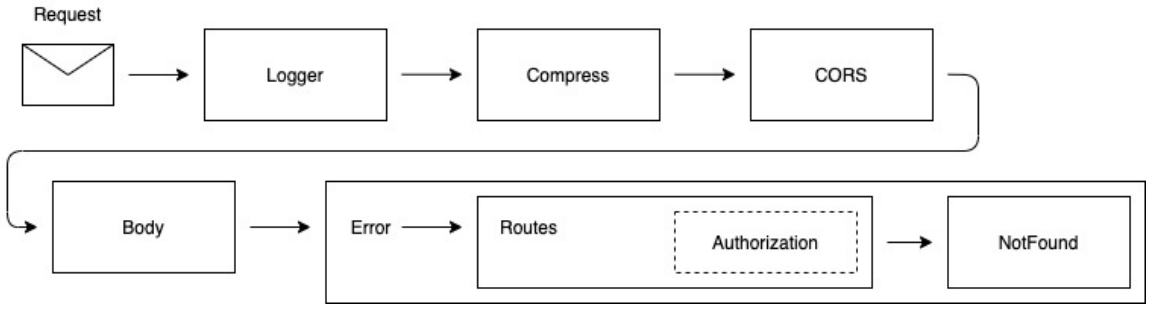
Po zvážení všech kladných a záporných faktorů jsem se rozhodl, že pro potřeby této aplikace bude vhodnější implementovat REST API. Implementace REST API pouze s HTTP knihovnou, která je obsažena v Node.js, by ale byla zdlouhavá. Bylo tedy vhodné použít komplexnější řešení. Pokud uvážíme frameworky, které se dnes běžně využívají pro tvorbu takových API, nabízí se 2 možnosti. Express.js[12] a Koa.js[13]. Oba frameworky byly vytvořeny stejným týmem, ovšem Koa.js je lehčí než Express.js a klade důraz na nahrazování *callbacků* za *async-await*, což dělá kód znatelně čitelnějším. Pro implementaci jsem tedy zvolil Koa.js. Samotný framework funguje na bázi *middleware*. Jednotlivé *middlewares* jsou volány postupně za sebou, přičemž aktuální *middleware* vždy spouští následující *middleware* a může mu předávat data.

```
1    /* ... */
2    app.use(logMiddleware)
3    app.use(koaCompress())
4    app.use(koaCors())
5    app.use(koaBody())
6    app.use(errorMiddleware)
7    app.use(routes)
8    app.use(notFoundMiddleware)
9    /* ... */
```

Výpis 7.1: Ukázka přidávání jednotlivých *middlewares* do aplikace – *backend/src/app.ts*

Příchozí požadavek tedy postupně prochází rouru *middlewares* z výpisu 7.1. Nejdříve dojde k *logování* požadavku a poté ke komprimaci dat, pokud je potřeba. Následně se musí ověřit, zda požadavek vyhovuje *cross-origin resource sharing*[14] restrikcím. Poté se zanalyzuje tělo požadavku. Další v rouře *middlewares* je *errorMiddleware*. Jedná se o *try-catch* blok, který obaluje celý zbytek aplikace. Tento *middleware* zjišťuje, zda při vyhodnocování požadavku nenastala chyba. Pokud ano, odpoví klientovi se zpracovanou chybovou hláškou. Pak už zbývá pouze *routes* a *notFoundMiddleware*. *Routes* zajišťuje routování požadavku v rámci aplikace. Pokud by nebyla nalezena žádná cesta, která by vyhovovala požadavku, tak *notFoundMiddleware* vytvoří odpověd ve formátu chybové hlášky. Graficky znázorněný postup zprávy mezi *middleware* je znázorněný na obrázku 7.2

[11] Základní operace nad daty – *create, read, update, delete*
[12] https://expressjs.com/
[13] https://koajs.com/
[14] Mechanizmus, díky kterému je možné povolit přístup ke zdrojům pouze specifickým žadatelům.



Obrázek 7.2: Postup zprávy skrz *middleware*

Jakmile je nalezena cesta v routeru, může dojít k autentizaci požadavku, pokud je k dané cestě přiřazen autentizační *middleware*. Ta spočívá v ověření, zda požadavek přichází od přihlášeného uživatele. Následně se volají metody příslušného *controller*. *Controller* je poté zodpovědný za získání dat z požadavku a jejich validaci. Je tedy třeba validovat, zda data obsažená v požadavku odpovídají struktuře, kterou API na daném koncovém bodě očekává. Toho lze docílit následujícími způsoby:

1. validace jednotlivých parametrů,
2. validace schématem,
3. validace příkladem

Validovat jednotlivé parametry jeden po druhém by nebylo tolik efektivní a při změně byť jen jednoho parametru by to mohlo znamenat změny na mnoha místech v kódu. Efektivnějšími variantami jsou tedy 2 a 3. Při validaci příkladem je potřeba vytvořit příklad dat, který vyhovuje požadovanému schématu. Z takového exempláře se poté vyvodí schéma, které se použije při validaci dat. Takový přístup mi přišel ovšem poněkud neformální a vágní. Navíc by mohly vzniknout problémy s vlastními datovými typy, které by validátor nemusel podporovat. Rozhodl jsem tedy, že nejvhodnější způsob validace příchozích dat je 2. Nejznámější definovanou specifikací, pomocí které je možné určit, jakou strukturu má mít objekt, je JSON Schema[15]. Existuje široká škála knihoven, které tuto specifikaci implementují[16]. Pro prostředí Node.js je implementován balíček *jsonschema*[17]. Z dat požadavku se tedy vytvoří JSON, který se následně validuje za pomocí definovaného schématu. Ukázku kontroleru včetně validace vstupních dat můžeme vidět ve výpisu 7.2.

```
1  export const acceptMatch = async (ctx: Context) => {
2    const data: TRequestAcceptMatch = {
3      userId: ctx.params.id,
4      ...ctx.request.body
5    }
6
7    validate(data, JSONRequestAcceptMatch)
8    ctx.body = await usersOperations.acceptMatch(data)
9  }
```

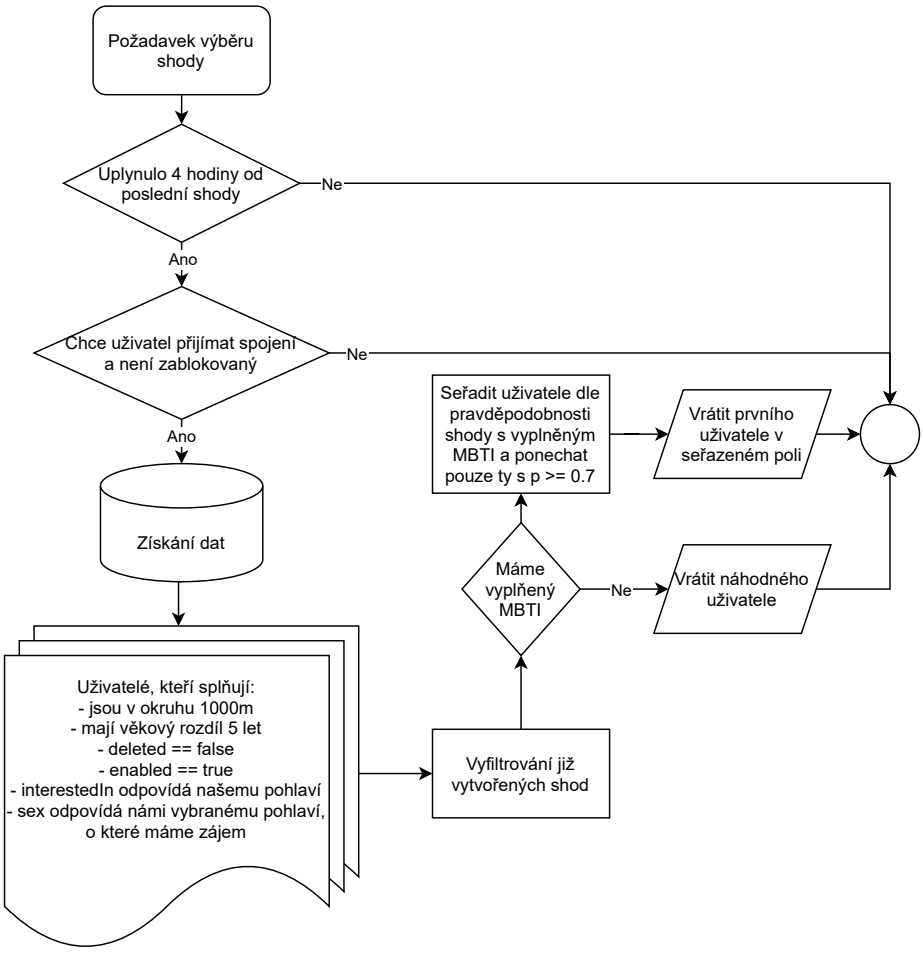
Výpis 7.2: Kontrolér požadavku o přijetí shody – *backend/src/controllers/users.ts*

[15] https://json-schema.org/
[16] https://json-schema.org/implementations.html
[17] https://www.npmjs.com/package/jsonschema

Jelikož je použit TypeScript, je vhodné validovaná data otypovat. Validace dat probíhá v momentě, kdy je aplikace spuštěna a zpracovává požadavek. Určení datového typu je ovšem potřeba před samotnou kompilací. Je tedy nutné vytvořit datové typy z definovaných schémat. Po vytvoření takových typů a validaci vstupních dat lze říct, že validovaná data vyhovují danému typu a lze jej tedy vstupním datům přiřadit.

S validovanými daty se poté volají metody příslušného *operation*. V těchto metodách se nachází business logika. *Operation* jsou například zodpovědné za zjišťení, zda uživatel, o jehož data žádáme, existuje. Pokud je potřeba, manipulují s daty. Také jsou zodpovědné za vyvolání chyb v případě, že je žádáno o data, které neexistují. Dílčí metody *operation* získávají data z *repository*, které implementují metody pro přístup do perzistentního úložiště. V našem případě tedy do Firestore, jak již bylo zmíněno v sekci 7.1. *Repository* vytvářejí z pohledu *operation* rozhraní databáze. Takový přístup je vhodný, jelikož umožňuje jednoduchou migraci na jinou databázi. V případě, že by vznikla potřeba nahradit stávající databázi, stačilo by pouze znovu implementovat metody v *repository*.

7.2.1 Algoritmus výpočtu shody



Obrázek 7.3: Vývojový diagram algoritmu výpočtu shody

Při vytváření nové shody je potřeba zohlednit některé parametry, které jsou klíčové pro vyhledání odpovídajícího spojení. Na obrázku 6.1 je zobrazen vývojový diagram použitého algoritmu.

Při příchozím požadavku o nalezení nové shody je potřeba zkontrolovat, zda od posledního propojení uplynuly alespoň 4 hodiny, jak bylo navrženo v sekci 6.1. Pokud ano, tak se pokračuje kontrolou, zda uživatel nemá zablokovaný profil a zda chce přijímat nové shody. Poté je potřeba získat data z databáze. Data, která vrátí databáze, by měla obsahovat pouze relevantní uživatele v našem okolí. Režie zpracování dat by na straně API měla být co nejměnší, tudíž databáze již musí vracet relevantní výsledky co se týče polohy a základních vlastností uživatelů. Nad takovými daty se provede vyfiltrování již vytvořených spojení (tudíž nebude možno vytvořit jednu shodu dvakrát) a následně se bere v potaz uživatelův MBTI typ, pokud je vyplněn. Posledním krokem je vrácení ideálního spojení zpět programu.

7.2.2 Databáze

V návrhu architektury (obrázek 6.8) byla znázorněna databázová vrstva. Po analýze modelu dat (podsekce 6.2.1) a průzkumu možných řešení, jsem došel k závěru, že je vhodné implementovat NoSQL databázi. K takovému rozhodnutí vedl fakt, že data, která je třeba uchovávat v databázi, nemají složitou strukturu a zároveň bylo potřeba vyhledávat v databázi uživatele na základě polohy. Bylo tedy zapotřebí zvolit prostorovou databázi (*spatial database*). Jelikož jsem se již rozhodl, že použiji některé služby, které nabízí Firebase, zvažoval jsem využít také Firestore. V úvahu přicházela i možnost implementovat databázi jako MongoDB. Ta by se ovšem musela hostovat a vzhledem k ostatním využitým službám by se obtížněji škálovala. Po zvážení všech možností byl nakonec vybrán Firestore.

Samotný Firestore neimplementuje funkce nad datovým typem *Geopoint*. Pro správnou funkcionalitu aplikace je ale nutné vyhledávat nejbližší uživatele. Této funkcionality lze docílit použitím nádstavby Geofirestore[18]. Ta využívá geohashů pro výpočet vzdálenosti mezi dvěmi body a v databázi je nutné zaznamenávat strukturu „g“, která je obálkou pro *geohash* a *geopoint*.

[18] https://www.npmjs.com/package/geofirestore

Výsledné schéma databáze vypadá následovně:

```
/
├── users
│   └── {userId}
│       ├── Map[] acceptedMatches
│       │   ├── Timestamp date
│       │   └── String userId
│       ├── String avatarUrl
│       ├── Timestamp birthday
│       ├── String[] conversations
│       ├── Geopoint coordinates
│       ├── Boolean deleted
│       ├── Map g
│       │   ├── String geohash
│       │   └── Geopoint geopoint
│       ├── String id
│       ├── String interestedIn
│       ├── Boolean legalAgreement
│       ├── Map[] matches
│       │   ├── Timestamp date
│       │   └── String userId
│       ├── String name
│       ├── String sex
│       └── String token
└── chat
    └── {chatId}
        ├── Timestamp started
        └── messages
            └── {messageId}
                ├── Timestamp date
                ├── String from
                ├── String message
                └── String to
```

7.3 Mobilní aplikace

Jak již bylo zmíněno v sekci 6.2, nejvhodnější způsob implementace navržené aplikace je multiplatformní nativní aplikace 4.3. Ze všech možností řešení uvedených v podsekci 4.3 se jako nejlepší volba jevil s ohledem na podporu komunity React Native. Jelikož je komunita okolo React Native velká, vzniká široká škála užitečných balíčků, které lze použít při tvorbě aplikace. Existují balíčky pro směrování v aplikaci, různé pomocné knihovny, nebo celé předdefinované komponenty – potažmo *headless* komponenty[19].

Při vývoji aplikace pomocí React Native existují 2 způsoby, jak aplikace vyvíjet. Lze použít buď prostředí Expo CLI[20], nebo React Native CLI. Prostředí Expo obsahuje předdefinované funkce a knihovny pro práci s API zařízení a usnadňuje sestavovací proces.

[19] Komponenty, jenž definují chování, ovšem nedefinují vzhled.
[20] https://expo.io/

Vývojář má možnost využít aplikaci Expo App, díky které může testovat aplikaci nezávisle na platformě. Nevýhodou ovšem je, že nelze použít knihovny, které jsou napsané v nativním kódu platformy a nelze přidávat nativní moduly. Naproti tomu React Native CLI obsahuje pouze základní sadu knihoven (*„Hello world“* v prostředí Expo má přibližně 25MB, jelikož obsahuje celou sadu předdefinovaných knihoven a funkcí) a lze přidávat nativní moduly. Nevýhodou je, že je třeba aplikaci sestavovat pomocí XCode[21] a Android Studio[22], což s sebou nese problémy s podepisováním aplikace klíči. Zároveň nelze vyvíjet aplikaci pro iOS bez přístupu k počítači se systémem macOS. Vzhledem k charakteristice aplikace jsem zvolil použití React Native CLI. K tomuto rozhodnutí vedla především potřeba využívat polohu zařízení na pozadí.

Samotnou aplikaci tvoří sada vytvořených pohledů, komponent, kontextů, funkcí pro práci s backendem popsaným v podsekci 7.2, funkcí pro práci s API zařízení a knihoven. Jak již bylo zmíněno v podsekci 7.1, rozhodl jsem se pro použití Firebase. K samotné komunikaci s Firebase jsem použil knihovnu RNFirebase, která byla rovněž zmíněna v podsekci 7.1. Knihovna byla použita pro autentizaci uživatele prostřednictvím sociálních sítí (Facebook) a e-mailu, přijímání zpráv z Cloud messaging a nahrávání obrázků do Cloud storage.

Pro směrování mezi jednotlivými pohledy v aplikaci existují dva typy balíčků. Nativní a balíčky založené na jazyce JavaScript, které emulují navigaci. Nejpoužívanějším balíčkem, který zprostředkovává nativní navigaci je „react-native-navigation“[23] (dále RNN). U řešeních založených na Javascriptu to je „react-navigation“[24] (dále React Navigation). Zde jsem se rozhodoval na základě tří klíčových faktorů – výkon, integrace s jinými balíčky a míra použití v komunitě. Co se výkonu týče, nativní řešení dosahuje lepších výsledků. Ovšem integrace s jinými balíčky může být obtížnější. Částečně je to způsobeno tím, že RNN se chová ke každému pohledu, jako by to byla samostatná React aplikace. Obalení aplikace do kontextu, či stavového manageru, může v tomto ohledu působit problémy s navigací. Co se použití při tvorbě aplikací týče, React Navigation je podstatně používanější, než RNN (pro porovnání – počet stažení React Navigation za uplynulý týden je k datu 21.4 roven 158 364, naproti tomu RNN má stažení pouze 56 781). Po zvážení všech klíčových faktorů jsem se rozhodl použít řešení založené na JavaScript – React Navigation.

Se směrováním také úzce souvisí problém *deep linking*. Jedná se o technologii, která umožňuje otevřením odkazu v zařízení spustit aplikaci a přesměrovat uživatele na konkrétní místo v aplikaci. Pokud je již spuštěna, dojde k přesunutí aplikace do popředí a přesměrování. *Deep linking* má podporu v balíčku React Native a je velmi dobře zdokumentován. Využití *deep linking* v aplikaci je především při otevírání notifikací, které oznamují novou příchozí zprávu nebo novou shodu.

Veškeré komponenty v aplikaci jsem implementoval tak, aby bylo vždy možné jednoduše komponentu přestylovat, rozšířit, či jinak modifikovat. Základ všech komponent je v nativních komponentách. Výjimečně bylo potřeba implementovat chování komponenty rozdílně pro iOS a Android. Pro tyto případy existuje v React Native možnost vytvořit komponenty se stejným jménem, ale jinou příponou názvu souboru. Například komponenta *Loader.tsx* tedy může mít rozdílné implementace pro každou z platforem – *Loader.ios.tsx* a *Loader.android.tsx*.

[21] https://developer.apple.com/xcode/
[22] https://developer.android.com/studio
[23] https://www.npmjs.com/package/react-native-navigation
[24] https://www.npmjs.com/package/react-navigation

Každá React aplikace potřebuje uchovávat aktuální stav. Při implementaci webové aplikace je typické použití balíčků Redux[25] či Flux[26]. Od verze 16.3.0 React podporuje novou Context API. Tuto API lze využít jako náhradu za zmíněné balíčky. Redux je komplexní nástroj, který vývojáři umožňuje centrálně spravovat stav aplikace. Navíc ale umožňuje vytváření akcí, na základě kterých může dojít k úpravě dat na vícero místech. Naproti tomu Context API pouze obalí vybranou část aplikace a metody či data, které jsou v rámci takového Contextu dostupné, definuje vývojář. Po zvážení složitosti aplikace jsem usoudil, že nebude potřeba využít komplexních řešení. Aplikace bude sloužit co se dat týče pouze jako zobrazovací vrstva. Nepředpokládá se tedy, že by bylo potřeba komplexního stavového manageru. Využil jsem tedy jednoduchého Context API, kterým jsem obalil celou aplikaci.

Komunikace s backendem, jehož implementace byla popsána v sekci 7.2, je zajištěna prostřednictvím jednoduché Fetch API[27]. Pomocí Fetch API je implementována sada funkcí, které se dotazují na data na příslušných koncových bodech, které backend zpřístupňuje. Návratová hodnota samotných metod je vždy otypována, tudíž při vývoji je práce s backendem pohodlnější.

7.3.1 Práce s API zařízení

Pro korektní funkcionalitu je třeba využívat některé API zařízení. Jedná se o geolokaci a notifikace.

Aby tyto API fungovaly korektně na všech cílových platformách, je potřeba se nejdříve dotázat uživatele, zda poskytne aplikaci příslušná oprávnění pro přístup k API. Většina balíčků, které implementují komunikaci se samotným rozhraním obvykle nabízí možnost dotázat se na příslušná oprávnění. To ovšem znamená, že pokud používáme více rozhraní zařízení a o udělení oprávnění žádáme prostřednictvím dílčích balíčků, můžeme narazit na nekonzistenci chování. Pro účely eliminace inkonzistentního chování existuje balíček „react-native-permissions“[28]. Ten nabízí sjednocené funkce pro dotazování na oprávnění.

```
1   ReactNativeForegroundService.add_task(
2       () => {
3           ensureLocationPermission(() => {
4               RNLocation.subscribeToLocationUpdates(
5                   ([ locations ]) => {
6                       userUpdatePosition({
7                           id: auth().currentUser!.uid,
8                           latitude: locations.latitude,
9                           longitude: locations.longitude
10                      })
11                  }
12              )
13          })
14      }, { delay: 1000, onLoop: false, taskId: 'background_location_sniff',
15      onError: (e: any) => console.log('Error logging:', e),
16      },
17  )
```

Výpis 7.3: Registrace naslouchače geolokace pro Android – *frontend/src/helpers/location.ts*

[25] https://www.npmjs.com/package/redux
[26] https://www.npmjs.com/package/flux
[27] https://reactnative.dev/docs/network
[28] https://www.npmjs.com/package/react-native-permissions

Jak již bylo navrženo v sekci 6.1, aplikace má využívat sledování telefonu pro určení vhodných shod. Jelikož se ale nepředpokládá, že uživatel bude mít vždy aplikaci zapnutou a na popředí, je nutné, aby aplikace odesílala informace o poloze uživatele také na pozadí, nebo když je vyplá. Takového chování lze docílit, když budeme získávat pouze významné změny v poloze. Aplikace tak nebude spotřebovávat značné množství baterie a operační systém cílové platformy aplikaci neukončí. U zařízení se systémem Android je navíc potřeba spustit *foreground service*. Ta zobrazuje ve stavovém řádku permanentní notifikaci, která informuje uživatele o využití polohy. V případě, že dojde ke značné změně polohy, odešle se požadavek na backend, který vyhodnotí, zda existuje možné spojení. Pokud spojení bylo nalezeno, odešle se notifikace oběma účastníkům. Samotné naslouchání změnám polohy je možné prostřednictvím balíčků. Vhodné rozhraní poskytují „react-native-location“[29], „@react-native-community/geolocation“[30] nebo „react-native-geolocation-service“[31]. Integrace s *foreground service* byla ovšem úspěšná pouze s balíčkem „react-native-location“, který jsem nakonec použil. Registrace naslouchače pro systém Android je popsána ve výpisu 7.3.

Zpracování notifikací je v aplikaci implementováno pomocí již zmíněné knihovny RNFirebase. Aby bylo možné zaslat zařízení notifikaci, je třeba vygenerovat token. Tento token je vygenerován zařízením a následně odeslán do backendu. Příklad vygenerování tokenu je ve výpisu 7.4. Token je následně využit v backendu pro zasílání notifikací.

```
1    /* ... */
2    console.log('NOTIFICATION: Authorized...')
3    const token = await messaging().getToken()
4    await userSetNotificationToken({ id: user.uid, token })
5    console.log('NOTIFICATION: Notification token request send to API')
6    return messaging().onTokenRefresh(async refreshedToken => {
7      await userSetNotificationToken({
8        id: user.uid,
9        token: refreshedToken,
10     })
11   })
12   /* ... */
```

Výpis 7.4: Získání tokenu pro zařízení, odeslání tokenu na API a zaregistrování naslouchače v případě, že by došlo k expiraci tokenu – *frontend/src/routes/index.tsx*

U samotného zpracovávání tokenu je nutné ošetřit tři stavy:

1. Stav, kdy je aplikace v popředí
2. Stav, kdy je aplikace v pozadí
3. Stav, kdy je aplikace vypnutá

Ke každému jednotlivému případu je nutné implementovat vlastní metodu, která bude takový typ zprávy zpracovávat. Například ve výpisu 7.5 lze vidět zpracování zprávy o novém propojení. Samotná notifikace s sebou nemusí nést pouze objekt notifikace, ale může být

[29] https://www.npmjs.com/package/react-native-location
[30] https://github.com/react-native-geolocation/react-native-geolocation
[31] https://github.com/Agontuk/react-native-geolocation-service

rovněž nositelem dat. Jedná se o volné 4KB, které mohou obsahovat informace pro *deep-linking*, nebo například informace o příchozí zprávě od jiného uživatele.

```
  if (!!user && !!userData) {
    messaging().onMessage(async remoteMessage => {
      /* ... */
      if (remoteMessage?.data?.type === 'acceptedMatch') {
        setNotification!('You have a new match!')
        userGetConversations({ id: user.uid }).then(data => {
          setConversations!(data)
        })
      }
      /* ... */
    })
  }
```

Výpis 7.5: Zpracování příchozí zprávy o novém spojení – *frontend/src/routes/index.tsx*

Kapitola 8

Testování

Aplikaci, která vznikla v kapitole 7 bylo potřeba řádně otestovat. Testování této aplikace probíhalo v průběhu vývoje, poté na jednotlivých platformách a nakonec samotným použitím aplikace dvěma dobrovolníky.

8.1 Testování API

Při vývoji byla aplikace kontinuálně testována. Bylo potřeba zajistit, aby jednotlivé změny v API nezapřičinily její nefunkčnost, tudíž bylo vhodné implementovat alespoň základní integrační testy. Tyto testy jsou v aplikaci implementovány především pro operace nad uživatelem, kde je potřeba zvlášť dbát na funkčnost systému. Jednotlivé testy byly implementovány pomocí frameworku Mocha.js, který byl zmíněn v sekci 4.4.4.

Navíc k integračním testům byla API neustále testována za pomocí programu Postman[1], který umožňuje zasílat požadavky na API se specifikací HTTP metody. Tyto testy byly prováděny především za účelem zjištění, zda data získané z API vhodně reflektují provedené změny.

8.2 Testování na cílových platformách

Aplikace byla testována na zařízení s iOS 14.4.2 (iPhone X) a Android 10 (Samsung Galaxy A12). Na obou platformách byla otestována řada úkonů, které jsou popsány společně s výsledky v tabulce 8.1. První oblasí testování byla samotná tvorba uživatelského profilu. Následně bylo sledováno, zda aplikace správně odesílá polohu zařízení při zaznamenání významných změn polohy. Rovněž bylo otestováno, zda aplikace korektně přijímá notifikace jak na popředí aplikace, tak na pozadí. Notifikace s sebou mohou nést informaci o *deep linking*. Dalším bodem testování tedy bylo, zda po otevření notifikace dojde k otevření aplikace na správném pohledu se správnými daty. Poslední oblastí testování na cílových platformách bylo testování chatu.

Při testování na jiných Android zařízení se ukázalo, že mohou nastat problémy s *Foreground service*, která odesílá polohu zařízení na API. Tu některá zařízení ukončují po nespecifikované době. Příčinou může být optimalizace využití baterie ze strany výrobce zařízení, nebo jiné omezení vůči aplikacím. Na testovacím zařízení Samsung Galaxy A12 byla *Foreground service* spuštěna bez komplikací po celou dobu pozorování (3 dny).

[1] https://www.postman.com/

	Popis testu	Android	iOS
1.	**Instalace aplikace**	ANO	ANO
2.	**Vytvoření účtu pomocí e-mailu**	ANO	ANO
3.	**Vytvoření účtu pomocí Facebooku** Poznámka: Uživatelův Facebook profil musí být přiřazen k aplikaci jako „tester“.	ANO	ANO
4.	**Přihlášení do účtu pomocí e-mailu**	ANO	ANO
5.	**Vyplnění profilu**	ANO	ANO
6.	**Vytvoření spojení při přiblížení ke shodě**	ANO	ANO
7.	**Notifikace je odeslána v případě nové shody**	ANO	ANO
8.	**Otevření notifikace odkáže uživatele na pohled se shodou**	ANO	ANO
9.	**Uživatel může přijímout shodu**	ANO	ANO
10.	**Vytvoření spojení odešle účastníkům notifikaci**	ANO	ANO
11.	**Uživatelé mohou chatovat po dobu 2 hodin od vytvoření spojení**	ANO	ANO

Tabulka 8.1: Testování na cílových platformách

8.3 Uživatelské testování

Aplikace byla rovněž otestována simulací reálné situace. Subjekty pro testování byli muž a žena. Oba z dvojice jsou zástupci cílové skupiny, která byla definována v sekci 5.2. Muž disponoval zařízením se systémem Android 10, žena zařízením se systémem iOS 14.4. Níže jsou popsány dílčí kroky průběhu testování.

1. Žena se nacházela na zastávce „Moravské náměstí“ integrovaného dopravního systému v Brně. Muž se nacházel na zastávce „Semilasso“ – rovněž v Brně.
2. Oba z účastníků se v aplikaci zaregistrovali a vytvořili své uživatelské profily.
3. Muž nastoupil do tramvaje, která jela směrem k zastávce „Moravské náměstí“.
4. Přibližně u zastávky „Antonínská“ přišla oběma účastníkům notifikace o vytvoření.
5. Oba účastníci spojení přijali.
6. Účastníkům bylo umožněno chatovat po dobu 2 hodin.
7. Po uplynutí intervalu 2 hodin již nenalezli svou shodu na hlavním pohledu aplikace.

Současně s testováním samotné funkcionality aplikace bylo provedeno pozorování, zda některý z účastníků nemá problémy s používáním aplikace z pohledu UX. Zde nebyly vypozorovány žádné nepřesnosti a oba z účastníků se v aplikaci orientovali s přehledem.

Kapitola 9

Závěr

Cílem mé práce bylo vytvořit multiplatformní aplikaci pro netradiční seznamování lidí, která opravuje problémy současných aplikací. Aby bylo možné takovou aplikaci vytvořit, nastudoval jsem potřebné materiály týkající se seznamování lidí a analyzoval jsem současné řešení.

Na základě vědomostí nabytých ze studování problematiky jsem udělal průzkum trhu metodou dotazníkového šetření. Výsledky dotazníkového šetření jsem zanalyzoval a na základě této analýzy jsem vytvořil design aplikace z funkcionální a vzhledové stránky. Dotazníkové šetření ukázalo na problém neosobního seznamování, které současné aplikace podporují. Návrh jsem poté implementoval za použití vybraných technologií. Výslednou aplikaci jsem úspěšně otestoval na cílových platformách. Součástí testování byla rovněž simulace reálného užití aplikace.

Výsledná aplikace využívá geolokačních nástrojů pro vyhledání možných spojení v okolí uživatele. Pro určení, zda nalezené spojení je vhodné, je použit psychologický model MBTI. Po vytvoření spojení a následném akceptování oběma účastníky, je uživatelům umožněno po dobu 2 hodin chatovat. Cílem je, aby se uživatelé co nejdříve domluvili na osobní schůzce a netrávili chatováním nadměrné množství času. Po dvou hodinách již chat nelze dohledat. Jednotlivé shody je možné vytvořit v časových rozestupech 4 hodin.

Aplikace je škálovatelná a do budoucna budu v jejím vývoji určitě pokračovat. Nabízí se zde možnost vytvořit neuronovou síť, která by uživatelům nabízela shody. Taková neuronová síť by mohla být založena na pozorování, zda se uživatelé skutečně setkají. Dále je zde prostor pro zlepšení bezpečnosti aplikace. Lze například šifrovat zprávy, které jsou uloženy v databázi pomocí asymetrického šifrování (implementovalo by se tedy end-to-end šifrování). Také by se dalo omezit množství informací, které je obsaženo v odpovědích z API.

Aby aplikace korektně fungovala, je potřeba, aby ji používalo dostatečné množství lidí, kteří se pohybují především v rámci jednoho města. Aplikaci tedy nelze bez dostačující marketingové kampaně spustit. V případě, že by aplikace byla spuštěna na místě, kde ji používá málo lidí, nemuselo by nikdy dojít k vytvoření propojení. Tudíž by uživatelé aplikaci postupně odinstalovali. Plánuji tedy aplikaci nasadit ve větším městě, kterým je například Praha.

Literatura

[1] Arikha, N. *Swiping Right in the 1700s: The Evolution of Personal Ads.* únor 2014. Dostupné z: https://longreads.com/2014/02/20/swiping-right-in-the-1700s-the-evolution-of-personal-ads/.

[2] Bittner, K. a Spence, I. *Use case modeling.* Addison-Wesley Professional, 2003.

[3] Chazan, M. Toward a long prehistory of fire. *Current Anthropology.* University of Chicago Press Chicago, IL. 2017, sv. 58, S16, s. S351–S359.

[4] Chen, S.-C., Gulatit, S., Hamid, S., Huang, X., Luo, L. et al. A three-tier system architecture design and development for hurricane occurrence simulation. In: IEEE. *International Conference on Information Technology: Research and Education, 2003. Proceedings. ITRE2003.* 2003, s. 113–117.

[5] Dam, R. F. a Siang, T. Y. *Personas – A Simple Introduction.* Dostupné z: https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them.

[6] Engl, T. Analysis and proposal of public relations activities for OEM Automatic s.r.o. in segment B2B. Vysoká škola ekonomie a managementu. 2012.

[7] Fast, J. *Body language.* Simon and Schuster, 1970.

[8] Finkel, E. J., Eastwick, P. W. a Matthews, J. Speed-dating as an invaluable tool for studying romantic attraction: A methodological primer. *Personal Relationships.* Wiley Online Library. 2007, sv. 14, č. 1, s. 149–166.

[9] Flannery, K. V. Prehistoric social evolution. *Research frontiers in anthropology.* Prentice-Hall Englewood CIiffs, NJ. 1995, s. 1–26.

[10] Ibanez, A. *New UIDatePicker in iOS 14.* červenec 2020. Dostupné z: https://www.andyibanez.com/posts/new-uidatepicker-ios14/.

[11] Jung, C. G. *Psychological types.* Routledge, 2014.

[12] Kuboková, K. Kvalitativní studie užití online seznamovacích serverů v České Republice. Univerzita Karlova, Fakulta sociálních věd. 2016.

[13] Li, Q. a Chen, Y.-L. Data flow diagram. In: *Modeling and Analysis of Enterprise and Information Systems.* Springer, 2009, s. 85–97.

[14] Li, Q. a Chen, Y.-L. Entity-relationship diagram. In: *Modeling and Analysis of Enterprise and Information Systems.* Springer, 2009, s. 125–139.

[15] Luo, L. Software testing techniques. *Institute for software research international Carnegie mellon university Pittsburgh, PA*. 2001, sv. 15232, 1-19, s. 19.

[16] Mallen, M. J., Day, S. X. a Green, M. A. Online versus face-to-face conversation: An examination of relational and discourse variables. *Psychotherapy: Theory, Research, Practice, Training*. Educational Publishing Foundation. 2003, sv. 40, 1-2, s. 155.

[17] McLeod, S. Maslow's hierarchy of needs. *Simply psychology*. 2007, sv. 1, s. 1–8.

[18] MentalHelp. *Psychological Testing: Myers-Briggs Type Indicator*. Dostupné z: https://www.mentalhelp.net/psychological-testing/myers-briggs-type-indicator/.

[19] Michael, J. Using the Myers-Briggs type indicator as a tool for leadership development? Apply with caution. *Journal of Leadership & Organizational Studies*. Sage Publications Sage CA: Thousand Oaks, CA. 2003, sv. 10, č. 1, s. 68–81.

[20] Nassi, I. a Shneiderman, B. Flowchart techniques for structured programming. *ACM Sigplan Notices*. ACM New York, NY, USA. 1973, sv. 8, č. 8, s. 12–26.

[21] O'Neil, D. A. a Petty, M. D. Heuristic methods for synthesizing realistic social networks based on personality compatibility. *Applied Network Science*. SpringerOpen. 2019, sv. 4, č. 1, s. 1–49.

[22] Pittenger, D. J. Measuring the MBTI... and coming up short. *Journal of Career Planning and Employment*. 1993, sv. 54, č. 1, s. 48–52.

[23] Pototska, I. *How Much Does It Cost to Develop a Dating App Like Tinder?* Dostupné z: https://old.yalantis.com/blog/how-much-tinder-cost/.

[24] Reports, V. *Online Dating Market Size is Projected to Reach USD 3.592 Billion by 2025 - Valuates Reports*. Oct 2020. Dostupné z: https://www.prnewswire.com/news-releases/online-dating-market-size-is-projected-to-reach-usd-3-592-billion-by-2025---valuates-reports-301146610.html.

[25] Sabatier, P. A. Top-down and bottom-up approaches to implementation research: a critical analysis and suggested synthesis. *Journal of public policy*. JSTOR. 1986, s. 21–48.

[26] Schroeder, J., Kardas, M. a Epley, N. The humanizing voice: Speech reveals, and text conceals, a more thoughtful mind in the midst of disagreement. *Psychological science*. Sage Publications Sage CA: Los Angeles, CA. 2017, sv. 28, č. 12, s. 1745–1762.

[27] Sefcek, J. A., Brumbach, B. H., Vasquez, G. a Miller, G. F. The evolutionary psychology of human mate choice: How ecology, genes, fertility, and fashion influence mating strategies. *Journal of Psychology & Human Sexuality*. Taylor & Francis. 2007, sv. 18, 2-3, s. 125–182.

[28] Suttie, J. *Should You Call or Text? Science Weighs In*. 2020. Dostupné z: https://greatergood.berkeley.edu/article/item/should_you_call_or_text_science_weighs_in.

[29] Tankovska, H. *U.S. dating apps by audience size 2019.* 2021. Dostupné z: https://www.statista.com/statistics/826778/most-popular-dating-apps-by-audience-size-usa/.

[30] Tripathi, S. a Rani, C. The impact of agricultural activities on urbanization: Evidence and implications for India. *International Journal of Urban Sciences.* Taylor & Francis. 2018, sv. 22, č. 1, s. 123–144.

[31] Vu, L. *Publishing PWAs to Major App Stores: The Whys and Hows* -. Leden 2021. Dostupné z: https://www.simicart.com/blog/pwa-app-stores/.

Příloha A

Obsah přiloženého paměťového média

- `/`
  - `sources`
    - `backend` ..... Zdrojové kódy backendu
    - `frontend` ..... Zdrojové kódy mobilní aplikace
    - `README.md` ..... Popis zprovoznění projektu ve vývojovém prostředí
  - `resources`
    - `thesis` ..... Zdrojové soubory textové práce
    - `xdemel01.pdf` ..... Textová práce ve formátu PDF
    - `questionnaire` ..... Výsledky dotazníkového šetření
    - `app.apk` ..... Balíček aplikace pro instalaci na zařízení Android