
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**
**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**


# MOBILNÍ APLIKACE PRO VYHLEDÁNÍ FOTEK ÚČASTNÍKŮ

**MOBILE APPLICATION FOR FINDING MEETING PARTICIPANT PHOTOS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** — **MARTIN ČERVENÝ**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **Ing. VÍTĚZSLAV BERAN, Ph.D.**
**SUPERVISOR**



**BRNO 2018**

*Zadání bakalářské práce/19505/2017/xcerve19*

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

# Zadání bakalářské práce

Řešitel: **Červený Martin**
Obor: Informační technologie
Téma: **Mobilní aplikace pro vyhledání fotek účastníků**
**Mobile Application for Finding Meeting Participant Photos**
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s tvorbou aplikací pro vybranou mobilní platformu. Prostudujte možnosti využití Google API pro vyhledávání a využití obrázků.
2. Navrhněte mobilní aplikaci včetně GUI, která ze vstupního textového seznamu jmen osob automaticky nalezne jejich fotografie a umožní uživateli z několika navržených fotografií vybrat jednu výslednou pro každou osobu.
3. Vyberte vhodné existující nástroje a navrženou aplikaci implementujte.
4. Proveďte experimenty na efektivitu a použitelnost aplikace. Klíčové nedostatky opravte.
5. Vytvořte plakát a krátké video prezentující klíčové výsledky vašeho řešení.

Literatura:

- R. Hartson, P. Pyla. *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*, Morgan Kaufmann, ISBN-10: 0123852412, 2012.
- Dále dle pokynu vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a částečně body 3 a 4.

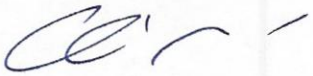
Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Beran Vítězslav, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
L.S. 612 66 Brno, Božetěchova 2



doc. Dr. Ing. Jan Černocký
*vedoucí ústavu*

# Abstrakt

Cieľom práce je navrhnúť a implementovať aplikáciu pre vyhľadanie účastnikov stretnutí, ich fotiek a menšiu správu týchto stretnutí s možnosťou písania poznámok. Aplikácia by mala zefektívniť prípravu na tieto stretnutia. Pre úspešný vývoj bol najskôr vykonaný prieskum cieľovej skupiny, analýza požiadavkov a následne návrh a implementácia. Výsledkom je funkčná mobilná aplikácia, ktorá je otestovaná s ohľadom na použiteľnosť užívateľského rozhrania.



# Abstract

The goal of this work is to design and implement an application to find meeting participants, their photos and to manage these meetings with the ability to write notes.The application should make preparations for these meetings more efficient. For the successful development, the target group survey, the requirements analysis and subsequently the design and implementation were performed. The result is a functional mobile application that is tested with aspect to usability of the user interface.



# Klíčová slova

mobilná aplikácia, iOS, užívateľské rozhranie, design, API, vyhľadanie fotiek, React Native, správa schôdzok a účastníkov

# Keywords

mobile application, iOS, user interface, design, API, finding photos, React Native, managing meetings and attendees



# Citace

ČERVENÝ, Martin. *Mobilní aplikace pro vyhledání fotek účastníků*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vítězslav Beran, Ph.D.

# Mobilní aplikace pro vyhledání fotek účastníků

## Prohlášení

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

.....
Martin Červený
15. května 2018


## Poděkování

Touto cestou by som sa chcel poďakovať vedúcemu práce Ing. Vítězslavovi Beranovi, Ph.D za odborné vedenie, pomoc a cenné rady pri návrhu a implementácii. Tiež ďakujem každému, kto sa podieľal na dokončovaní a testovaní aplikácie. Nakoniec by som sa chcel poďakovať rodine a priateľke za trpezlivosť a morálnu podporu.

# Obsah

# Kapitola 1

# Úvod

Mobilné zariadenia sú v dnešnej dobe bežnou súčasťou života väčšiny ľudí. Slúžia predovšetkým k jednoduchšiemu spájaniu ľudí, ale uľahčujú nám aj veľké množstvo rôznych aktivít. Každý človek sa snaží vyžívať také mobilné aplikácie, aby si čo najviac zjednodušil rôzne činnosti, či už ide o tie z bežného života, alebo naopak, tie z pracovného.

Cieľom tejto bakalárskej práce je navrhnúť a vytvoriť mobilnú aplikáciu, ktorá umožňuje vyhľadávanie osôb na schôdzkach a správu týchto schôdzok. Aplikácia poskytuje efektívnu prácu s poznámkami, schôdzkami a ich účastníkmi, ktorých je možné vyhľadávať prostredníctvom internetu. Zo zoznamu vyhľadaných fotografií je následne možné vybrať jednu výslednú pre každého účastníka.

Práca je rozčlenená do šiestich kapitol. Kapitola 2 sa venuje položením teoretického základu pre správny návrh UX dizajnu pomocou metodológie Design thinking. Dočítate sa tu taktiež o rôznych možnostiach a spôsoboch vyhľadania a prípadného spracovania fotiek ľudí. Na konci tejto kapitoly sú popísané už existujúce podobné riešenia. V ďalšej kapitole sa zameriavam na jednotlivé technológie umožňujúce vývoj mobilných aplikácií pre platformu iOS a výber najvhodnejšej technológie pre túto aplikáciu. Táto kapitola sa teda venuje hlavne frameworku React Native a rôznym knižniciam a architektúram, ktoré boli vybrané pre implementáciu tejto aplikácie. V štvrtej kapitole je popísaná analýza riešenia zahrňujúca definíciu problému a prieskum cieľovej skupiny s výstupom definície užívateľských potrieb. Ďalej je tu popísaný návrh dátového modelu, užívateľského rozhrania na základe týchto požiadavkov a návrh spôsobu vyhľadávania fotiek ľudí. Posledná kapitola je venovaná implementácii kľúčových prvkov aplikácie a popisom použitých nástrojov a knižníc. Záver tejto kapitoly tvoria rôzne užívateľské testy spolu s ich vyhodnotením.

# Kapitola 2

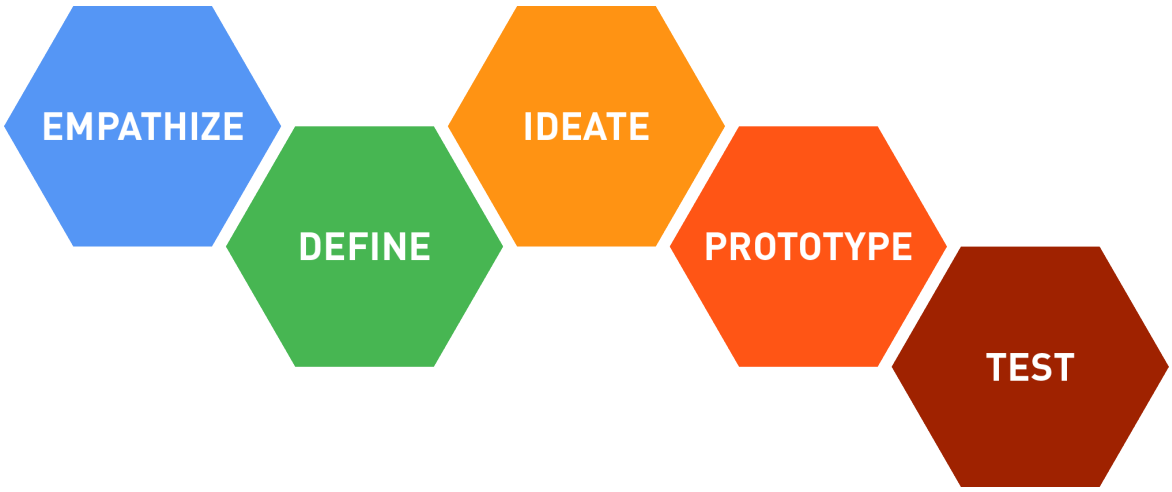
# Teoretický úvod

V tejto kapitole je popísaná základna teória potrebná k návrhu a implementácií výslednej aplikácie. Pre správny *user experience* (UX[1]) návrh je veľmi dôležité poznať koncového užívateľa a zanalyzovať požiadavky na aplikáciu. Na základe tejto analýzy sa následne vytvára návrh jednotlivých prvkov grafického rozhrania. Jedna z metodológií takéhoto vývoja softvéru, ktorá je využitá v tejto práci sa nazýva *Design thinking*.

Ďalej sa v tejto kapitole nachádzajú rozobraté rôzne možnosti pre vyhľadanie fotiek ľudí na internete a podobné existujúce aplikácie.

## 2.1 Design thinking

Design thinking je metodológia, ktorej cieľom je nájsť riešenie problému. Je využívaná za účelom vytvárania nových nápadov, inovácií, nečakaných pohľadov na vec a zlepšenia produktov či služieb. Táto metóda pozostáva z piatich rôznych fáz (obrázok 2.1) a tými sú Porozumenie (*Empathize*), Definovanie(*Define*) Vytváranie nápadov (*Ideate*), Prototypovanie (*Prototype*) a Testovanie (*Test*) Poznatky v tejto podkapitole sa opierajú o zdroj [13].




Obrázek 2.1: Design thinking - fázy [8]

[1]UX - https://en.wikipedia.org/wiki/User_experience_design

## Porozumenie

Cieľom tejto fázy je získať empatické chápanie problému, ktorý je potreba vyriešiť. Na začiatku je nutné vykonať výskum prostredníctvom pozorovania a konverzovania s budúcimi užívateľmi, kvôli porozumeniu ich skúsenostiam a hlbšiemu pochopeniu problémov. Porozumenie je základom pre proces návrhu a dovoľuje získať prehľad o používateľoch. Vcítenie sa do ich myslenia pomôže dosiahnuť čo najlepšie pochopenie daných problémov a užívateľských potrieb, ktoré sú základom vývoja konkrétnej aplikácie. Pre zosobnenie reálnych užívateľov sa využívajú tzv. persony.

### Persony

Persony sú fiktívni typickí reprezentanti cieľovej skupiny vyvíjanej aplikácie. Pre čo najpresnejšiu definíciu užívateľských potrieb je potreba persony do detailov prepracovať. Každá persona by mala obsahovať nasledujúce údaje:

- Meno
- Vek
- Rodinný stav
- Zamestnanie
- Bydlisko
- Záľuby a obľúbené veci

### Scenáre použitia

Sú to popisy bežných situácií, v ktorých sa môžu užívatelia ocitnúť a v ktorých budú aplikáciu používať. V každom scenári použitia je teda potreba popísať akým spôsobom by mohla aplikácia užívateľovi pomôcť a ako by ju mohol čo najefektívnejšie využiť pre vyriešenie jeho problému.

### Užívateľské potreby

Užívateľské potreby sú požiadavky, ktoré užívateľ od aplikácie očakáva a ktoré potrebuje. Definujú sa na základe vytvorených person a scenárov, ktoré tieto potreby dokážu jednoducho odhaliť. Ich definícia je najdôležitejší krok pre úspešný a použiteľný návrh užívateľského rozhrania. Na základe užívateľských potrieb je možné potom navrhnúť prvky rozhrania aplikácie tak, aby riešili všetky vopred definované problémy.

## Definovanie

Počas tejto fázy sa zhromažďujú informácie, ktoré boli získané vo fáze *Porozumenia*. Ide hlavne o správne pochopenie existujúcich problémov používateľov. Všetko, čo bolo pozorované a objavované je teraz potrebné sformulovať do daného problému. Výstup tejto fázy by mal byť odpoveďou na otázku: *Aký je konkrétny problém, ktorý je potreba riešiť novým produktom?* Zodpovedanie tejto otázky pomôže dať jasnú definíciu a štruktúru problému, ktorá sa prejaví v návrhu. Vďaka nej bude možné navrhnúť funkcie a prvky aplikácie tak, aby vedeli vyriešiť užívateľské problémy, alebo aspoň umožniť riešenie týchto problémov s minimálnym úsilím.

### Vytváranie nápadov

V tejto etape je treba začať vytvárať nové nápady. Zlepšilo sa pochopenie potrieb užívateľov vo fáze *Porozumenia* a prebehla analýza pozorovania v druhej fáze. Teraz je možné rozmýšľať trochu inak, ako bol definovaný problém. Ide o to identifikovať nové, alternatívne spôsoby riešenia. Je veľmi dôležité získať čo najviac nápadov a rozšíriť problémový priestor. V tejto fáze nie je žiadny nápad zlá myšlienka a množstvo nápadov nahradzuje ich kvalitu. Či sú nápady vôbec realizovateľné je nepodstatné, to sa posúdi neskôr.

Výstupom tejto fázy je získať čo najviac nápadov, aby pomohli vyriešiť definovaný problém, alebo poskytnúť prvky potrebné na obchádzanie problému.

### Prototypovanie

Vo fáze prototypovania sa vytvorí niekoľko lacných zmenšených verzií produktu alebo len špecifických funkcií, ktoré sa nachádzajú v produkte, aby sa dali preskúmať riešenia problémov vytvorené v predchádzajúcich fázach. Cieľom prototypovania je teda dať všetky myšlienky do skutočnej aplikácie a pochopiť, ktoré komponenty myšlienok fungujú a ktoré zas nie.

V tejto fáze sa začne uvažovať nad realizovateľnosťou nápadov. Prvé prototypy môžu byť len veľmi hrubé zobrazenia výsledného produktu. Sú vytvorené na to, aby poskytovali podnety na diskusiu s cieľom zdokonaľovania určitých funkcií. Prototypy je potreba testovať, aby sa dokázalo nájsť najlepšie možné riešenie pre každý z definovaných problémov v predchádzajúcich fázach. Realizované riešenia problémov sa buď ponechajú alebo vylepšia a znova preskúmajú. Ako výstup tejto fázy je potreba mať lepšiu predstavu o problémoch, ktoré sú momentálne prítomné a tiež informovanosť o tom, ako by sa skutoční používatelia správali a premýšľali pri interakcii s finálnou aplikáciou.

### Testovanie

V tejto fáze sa požaduje spätná väzba o prototypoch od používateľov a prinášajú sa reálnejšie prototypy, vďaka ktorým sa získa od nich ich hodnotenie. Cieľom je vyhodnotiť nápady na produkt, pochopiť viac o potrebách koncových užívateľov a zlepšiť produkt pred finálnym vydaním. Pri testovaní prototypoch si je vhodné položiť otázku, či spĺňa toto riešenie potreby používateľov a či sa zlepšilo plnenie úloh. Výsledky testovania sa ďalej používajú na predefinovanie jedného, prípadne viacerých problémov a informujú o porozumení používateľov, čo si myslia, ako sa správajú, alebo ako sa cítia. Počas tejto fázy sa teda tiež uskutočňujú zmeny a vylepšenia.

## 2.2 Vyhľadanie fotiek ľudí

V nasledujúcej podkapitole sú popísané rôzne možnosti vyhľadania a spracovania užívateľských dát a fotiek. Najviac možností ponúkajú rôzne sociálne siete, pretože obsahujú veľké databázy užívateľov a ich REST[2] API[3] umožňujú medzi nimi vyhľadávať.

[2]REST - Representational State Transfer, dostupné z: https://en.wikipedia.org/wiki/Representational_state_transfer

[3]API - Application programming interface, je rozhranie programovanie aplikácií

### Google Custom Search

Je platforma [11] poskytovaná firmou Google, založená na službe Google Search[4], teda vyhľadávači Google. Dovoľuje vývojárom vytvoriť si vlastný špecifický vyhľadávač pre svoju aplikáciu či web. Pre vytvorenie vyhľadávača je potrebný API kľúč, ktorý sa dá získať na vývojárskych stránkach spoločnosti Google. Možnosti na úpravu a konfiguráciu vyhľadávača sú prakticky neobmedzené, od úpravy front-endu, teda vzhľadu až po back-end. Taktiež je možné si nadefinovať, či sa má vyhľadávať v rámci vlastného webu alebo celého internetu. V takom prípade sa dá vyhľadávač nakonfigurovať, aby vyhľadával len to, čo je potrebné – napríklad len obrázky. Pre zobrazenie výsledkov je možné využiť štandardnú stránku od Google alebo vlastnú v rámci aplikácie. Pre väčšiu kontrolu nad vyhľadanými dátami Google vytvoril Custom Search JSON/Atom API[5], ktoré umožňuje získať vyhľadávané dáta vo formáte JSON[6] alebo Atom[7] a následne ich bez problémov programovo ďalej spracovať. Toto API však poskytuje len 100 hľadaní denne bezplatne a následne je spoplatnené.

Využitie Google Custom Search však pre vyhľadanie ľudí a ich fotiek na základe mena nie je príliš vhodné, keďže nemá vlastnú databázu užívateľov a vyhľadáva podľa kľúčových slov. V takom prípade sa medzi hľadané výsledky dostanú nerelevantné odkazy a fotky, ktoré je prakticky nemožné odfiltrovať.

### LinkedIn API

LinkedIn API [14], ktoré beží na platforme REST, poskytuje jednoduchú konzistentnú reprezentáciu ľudí, spoločností, pracovných ponúk a rôznych vzťahov medzi nimi. Dáta, ktoré poskytuje je možné čítať v XML[8] alebo JSON formáte. Získať je možné všetky verejné informácie z profilu vrátane profilovej fotky. Tieto informácie môžu byť rôzne zoskupené, závisí podľa jednotlivých parametrov volaní. Z dôvodu bezpečnosti dát, je pre získanie prístupu k API nutná autentifikácia pomocou OAuth 2.0[9] protokolu. Tento protokol využíva prihlásenie pod užívateľským LinkedIn účtom, na základe ktorého vygeneruje unikátny token so 60 dňovou platnosťou. Pomocou platného tokenu je následne možné volať jednotlivé metódy REST API.

Od roku 2015 je prístup k API obmedzený len pre schválených partnerov spoločnosti LinkedIn a teda ho nie je možné verejne používať.

### Google+ API

REST API sociálnej siete Google+ [12]ponúka metódu priamo pre vyhľadávanie v databáze všetkých verejných profilov užívateľov podla zadaného reťazca, ktorý sa väčšinou skladá z mena a priezviska, príp. užívateľského mena. Toto volanie vyzerá nasledovne:

*GET https://www.googleapis.com/plus/v1/people&query={string}*

Za "string" sa dosadí hľadaný reťazec, ktorý obsahuje meno a priezvisko. Následne príde odpoveď vo formáte JSON, s ktorou je možné ďalej pracovať.

[4] https://developers.google.com/search/

[5] https://developers.google.com/custom-search/json-api/v1/overview

[6] JSON - JavaScript Object Notation, je formát zápisu pre ukladanie a prenos dát

[7] ATOM - Atom Syndication Format, je webový štandard pre publikovanie syndikovaného obsahu, dostupné z: https://cs.wikipedia.org/wiki/Atom_(standard)

[8] XML - eXtensible Markup Language, je rozšíriteľný značkovací jazyk

[9] https://oauth.net/2/

Pre prístup k tejto API metóde je taktiež nutná autentifikácia, buď pomocou API kľúča alebo podobne ako pri LinkedIn API - cez OAuth 2.0 protokol. API kľúč je možné vygenerovať v rámci užívateľského Google účtu a ostáva nemenný. Nie je ho potreba znova generovať. Výhodou oproti OAuth 2.0 je to, že nie je potrebné prihlásenie pomocou vlastného užívateľského účtu.

V súčasnej dobe však toto API nie je funkčné, keďže nevracia žiadne výsledky a teda ho nie je možné využiť.

### Facebook Graph API

Facebook Graph API [6] ponúka podobne ako aj API od Google+ vyhľadávanie priamo medzi verejnými užívateľskými účtami. Pre využívanie je najskôr nutné založiť aplikáciu na vývojárskej stránke Facebooku[10] a následne ju integrovať s mobilnou aplikáciou pomocou *Bundle ID*[11] pre iOS alebo *Key Hashes*[12] pre Android. V nasledujúcom volaní je potreba len doplniť "string" a typ hľadania, ktoré je v tomto prípade "user", keďže je potreba vyhľadávať medzi užívateľmi.

*GET graph.facebook.com/search?q={string}&type={user}*

Pre zavolanie API je taktiež potrebný *access token* získaný prostredníctvom OAuth 2.0 a teda je nutné prihlásenie. Získané dáta sú vo formáte JSON, avšak základné povolenia API sú značne obmedzené len pre dáta z verejného profilu, ako sú napr. meno, priezvisko a fotka. Pre získanie viacej rozšírených dát je potreba zažiadať Facebook o povolenie.

### Twitter API

Pre využitie Twitter API [16] je nutné vytvoriť online aplikáciu v Twitter Apps[13]. V tejto službe je potrebné vygenerovať unikátne kľúče *Consumer Key (API Key)*, *Consumer Secret (API Secret)*, *Access Token* a *Access Token Secret)*, ktoré je potreba pre API volania. Taktiež je možné nastaviť právomoci týmto kľúčom napr. len pre čítanie, prípadne aj zapisovanie.

Pre vyhľadanie medzi verejnými užívateľskými účtami je možné využiť nasledujúce volanie, kde sa "query" nahradí menom a priezviskom, prípadne užívateľským menom.

*https://api.twitter.com/1.1/users/search.json?q=query*

Odpoveď, ktorá príde je vo formáte JSON a obsahuje dôležité dáta o užívateľovi ako meno, priezvisko, lokáciu či jeho fotku. Výhodou tohto API je, že nevyžaduje OAuth 2.0 tokeny a teda nie je potrebné prihlásenie užívateľa.

[10] https://developers.facebook.com/

[11] Bundle ID - unikátné ID aplikácie na zariadení alebo v simulátore, definované v Xcode

[12] Key Hash - 28 znakový reťazec, ktorý Facebook používa na autentifikicáiu interakcií medzi aplikáciou a Facebookom

[13] https://apps.twitter.com

## 2.3 Existujúce riešenia

V nasledujúcich riadkoch je popísaná webová služba, ktorá sa venuje podobnej problematike vyhľadávania ľudí.

### Pipl.com

Webová aplikácia Pipl [14] je vyhľadávacia služba, ktorá ponúka vyhľadanie informácií o ľuďoch podľa mena, e-mailu, užívateľského mena či telefónneho čísla. Užívateľ zadá jednu z týchto informácií a môže si taktiež zvoliť krajinu, z ktorej by mala byť hľadaná osoba. V prípade nezvolenia konkrétnej krajiny sa vyhľadávajú osoby v rámci celého sveta. Ako výsledok dostane väčšinou niekoľko užívateľských účtov zo sociálnych sietí LinkedIn, Facebook, Google+ či Twitter. Pri každom účte je fotografia a taktiež zopár základných informácií ako krajina, vek či pohlavie. Táto služba ponúka aj vlastné API pre vyhľadanie, ďalšie spracovanie a využitie výsledkov vo vlastnej aplikácii. Toto API je však platená služba, rovnako ako aj ich rozšírené webové vyhľadávanie ľudí.

[14] http://www.pipl.com

# Kapitola 3

# Vývoj mobilných aplikácií pre platformu iOS

Existuje niekoľko možností ako vyvíjať aplikácie pre platformu iOS. Prvou z nich je vývoj *natívnej*[1] aplikácie. Tá sa vyvíja v programovacom jazyku Objective-C alebo modernejšej variante Swift a pre takýto vývoj je potrebné mať vývojové prostredie Xcode.

Druhou možnosťou je vývoj *cross-platform*[2] aplikácie, teda multiplatformovej, ktorá beží pod jedným zdrojovým kódom na viacerých operačných systémom rovnako. Pre takýto vývoj aplikácie je možné si vybrať z viacerých frameworkov. Najznámejšie sú React Native ktorý využíva programovací jazyk JavaScript, Xamarin[3], Ionic[4] či PhoneGap[5].

V tejto práci je využitý framework React Native [3] , ktorého výhody a nevýhody sú rozobraté v nasledujúcich riadkoch. Kvôli uľahčeniu práce sa spolu s ním najčastejšie využíva sada nástrojov Expo, ktorá urýchľuje vývoj a testovanie aplikácie. Pre dizajn grafického užívateľského rozhrania a všetkých základných komponent je možné využiť nejakú multiplatformovú knižnicu, ktorá jednotlivé komponenty zobrazuje špecificky pre každú platformu. Medzi tri najznámejšie patria NativeBase, React Native Elements[6] a Shoutem[7]. Kvôli prirodzenému vzhľadu komponent so špecifickým dizajnom pre jednotlivé operačné systémy Android a iOS bola v tejto aplikácií využitá knižnica NativeBase.

## 3.1 React

JavaScriptová knižnica React [2] je popúlarna pre svoju možnosť jednoduchého vytvárania interaktívnych užívateľských rozhraní. Za jeho vývojom stojí firma Facebook a Instagram. Ako opensource bol React vydaný v roku 2013, hoci ho Facebook už niekoľko rokov pred tým aktívne využíval. Po jeho vydaní bol však React veľmi kritizovaný z dôvodu jeho unikátnosti a miešania HTML kódu s programovaním. Avšak po vydaní tutoriálu, ktorý vysvetľoval React pri riešení konkrétnych problémov sa ukázalo, že išlo len o nepochopenie myšlienky a začal narastať na popularite. Dnes je React veľmi populárny a využíva ho mnoho známych veľkých firiem.

[1] Natívna aplikácia - aplikácia vyvíjaná priamo na konkrétnu platformu
[2] Cross-platform - https://en.wikipedia.org/wiki/Cross-platform
[3] https://www.xamarin.com/
[4] https://ionicframework.com/
[5] https://phonegap.com/
[6] https://react-native-training.github.io/react-native-elements/
[7] http://www.shoutem.com/

React je postavený na vytváraní jednotlivých komponent, ich zapuzdrení a znovupoužívaní v kóde, čo šetrí čas a vytvára lepšiu prehľadnosť v aplikácií. Tieto komponenty si riadia svoj vlastný stav a je možné ich skladať do zložitejších celkov. Na jednoduchý zápis využíva React syntax **JSX**, ktorej ukážku je možné vidieť vo výpise 3.1. Jednotlivé komponenty sú volané syntaxou jazyka HTML (XML), kvôli čomu bol aj React zo začiatku nepochopený. S HTML však nemajú nič viac spoločné. React využil len jeho jednoduchú a známu syntax, ktorá je pre ľudský mozok lepšie čitateľná a pochopiteľná, než zápis pomocou funkcií. React je často označovaný aj za *View* v MVC[8] frameworku. Nie je to však celkom pravda, nakoľko dané komponenty majú aj určitú logiku, čiže predstavujú aj časť *Controller*.

```
class HelloMessage extends React.Component {
    render() {
        return (
            <div>
                Hello {this.props.name}
            </div>
        );
    }
}

ReactDOM.render(
    <HelloMessage name="Taylor" />,
    mountNode
);
```

Výpis 3.1: Príklad jednoduchej komponenty v Reacte.

## 3.2 React Native

React Native je pomerne nový framework vydaný v roku 2015, ktorý aplikuje architektúru a koncept Reactu na natívne mobilné aplikácie a taktiež využíva túto JavaScriptovú knižnicu. Jeho účelom je vyvíjať cross-platformové aplikácie. Prináša to niekoľko výhod, ale samozrejme aj nevýhod. Výsledná aplikácia je na nerozoznanie od natívnej aplikácie napísanej vo Swifte, Objective-C alebo Jave. React Native jednoducho spája základné komponenty bežných iOS a Android aplikácií za použitia JavaScriptu.

Keďže však JavaScript nie je jazyk, ktorý beží natívne v mobilnom zariadení, je potrebné použiť tzv. "bridge", ako je možné vidieť na obrázku 3.1. Ten umožní spustenie JavaScriptu a komunikáciu s procesorom telefónu. Znamená to, že telefón musí spúšťať *JavaScript Runtime Environment*, čo je vlastne behové prostredie pre JavaScript komunikujúce s natívnym kódom telefónu. V tomto prostredí je následne spustený kód React Native [5].

**Výhody:**

- jednoduchý prehľadný kód ľahký na učenie
- až 80% kódu sa dá využiť pre viaceré platformy
- znovupoužiteľné komponenty, možnosť zakomponovať React Native už do existujúcej aplikácie

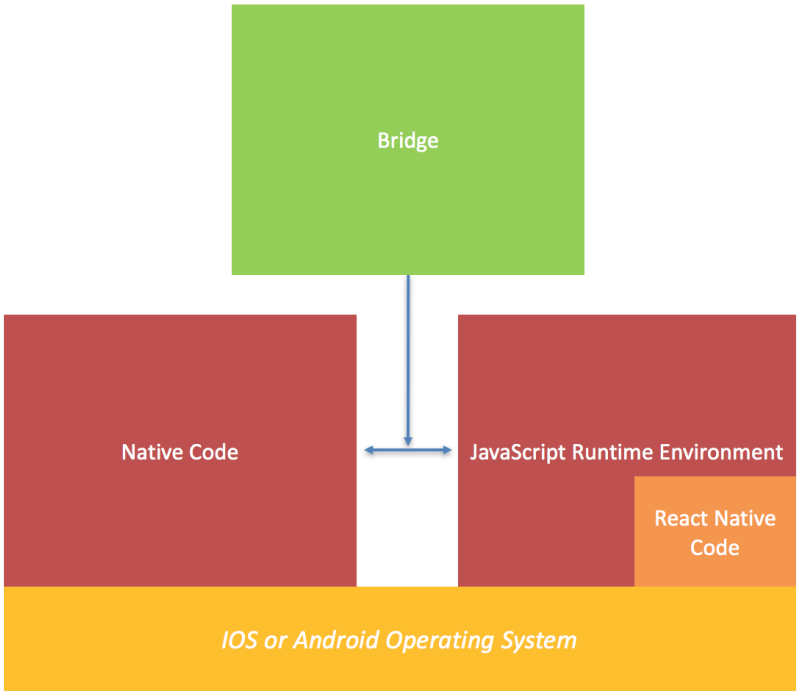
[8] MVC - Model–view–controller

- šetrí čas strávený pri vývoji, rovnako aj náklady na vývoj
- aplikácia je napísaná v JavaScripte a nie sú potrební vývojári zvlášť pre každý operačný systém
- možnosť použitia natívneho kódu z dôvodu optimalizácie, či úpravy natívnych komponent
- možnosť vidieť zmeny v kóde okamžite vďaka funkcii *Hot reloading*, ktorá automaticky skompiluje JavaScript vždy po uložení súboru s kódom

**Nevýhody:**

- pri implementácii niektorých špecifických aktivít (kamera, rôzne senzory zariadenia) je nutné využiť natívny kód
- tým, že je to pomerne nový framework a stále sa vyvíja, tak nie všetky funkcie sú dostupné hneď po vydaní novej aktualizácií systému
- aplikácia môže byť pri práci s niektorými API mobilu menej výkonná keďže zdieľa kód viacerých platforiem

React Native je taktiež populárny kvôli tomu, že ho využívajú známe aplikácie ako Facebook, Skype, Instagram, Tesla či Airbnb.




Obrázek 3.1: Architektúra React Native [15]

## 3.3 Expo

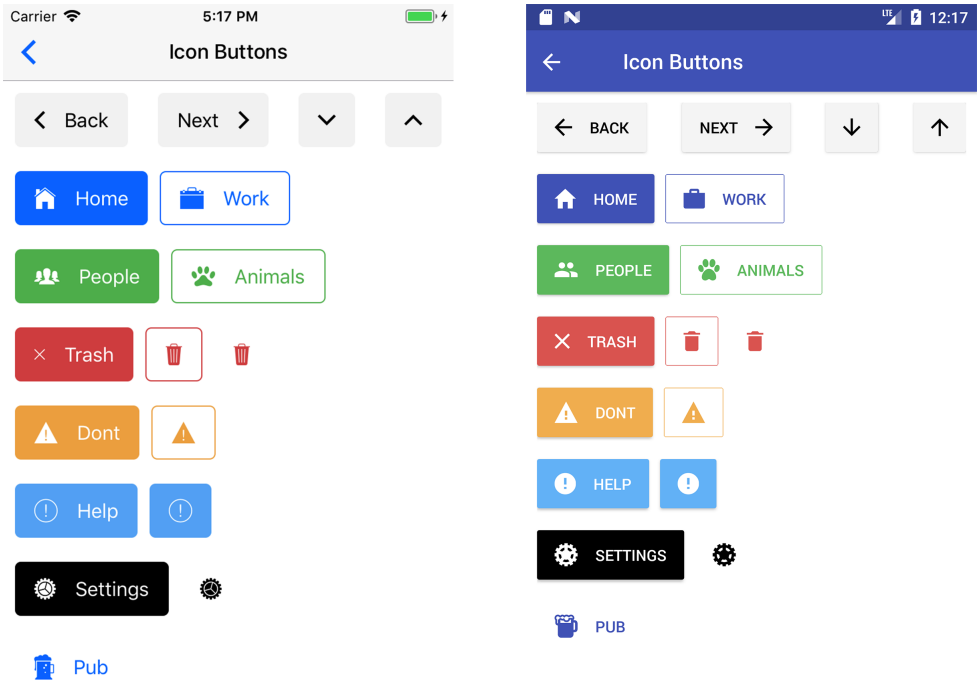
Sada nástrojov a knižníc Expo [1] umožňuje tvoriť natívne aplikácie pre iOS a Android. Toto SDK[9] poskytuje prístup k systémovým funkciám zariadení ako napríklad kamera, kontakty či lokálne úložisko. Nie je teda potreba písať žiadny natívny kód. Uľahčuje prácu s mnohými API, ktoré obsahuje mobilné zariadenie prípadne integruje rôzne internetové služby, ktoré sa pri vývoji často využívajú ako napríklad Facebook prihlásenie či databáza Firebase.

Aplikácia pre mobil Expo tiež dovoľuje simulovať tvorenú aplikáciu priamo na reálnom zariadení bez použitia simulátorov v Xcode alebo v Android studiu. Podoporuje tzv. Hot reloading čo je kompilácia JavaScriptu a okamžité zobrazenie zmien po uložení upravovaného súboru. Následné testovanie a publikácia aplikácie na verejnosť je možná aj bez zverejnenia aplikácie v App Store či Google Play. Stačí poznať URL adresu aplikácie, na ktorej sa nachádza QR kód[10], ktorý sa jednoducho naskenuje v mobilnej aplikácií Expo a a daná aplikácia sa spustí.

## 3.4 NativeBase

NativeBase [10] je open-source knižnica, ktorá ponúka cross-platformové UI[11] komponenty pre React Native. Tieto komponenty sa ľahko používajú a majú natívny vzhľad danej platformy, na ktorej sa spúšťajú (obrázok 3.2). Výhodou je teda jeden JavaScriptový kód, ktorým sa dá vytvoriť natívne užívateľské rozhranie. Veľkou výhodou je taktiež to, že vďaka využitiu modulárneho návrhového vzoru v tejto knižnici je možné jednotlivé základné komponenty ľubovoľne meniť a upravovať.

[9] SDK - Sofware Development Kit, je v preklade súbor nástrojov pre vývoj softvéru
[10] QR kód - https://sk.wikipedia.org/wiki/QR_kód
[11] UI - User Interface



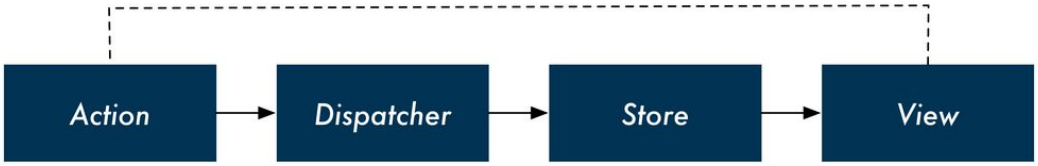

Obrázek 3.2: Zobrazenie tlačidiel na na rozdielnych platformách - iOS a Android. Prevzaté z dokumentácie [10]

## 3.5 Architektúra Flux

Správa stavov v Reacte môže byť pri väčších projektoch dosť zložitá a neprehľadná. Komponenty najvyššej úrovne spravujú stav, ktorý prúdi smerom nadol do podradených komponent. Akákoľvek mutácia stavu sa vždy uskutoční na vrchu, v nadradenej komponente. Najväčším problémom je tesné prepojenie medzi užívateľskými interakciami a zmenami stavov. Pri komplexnejších aplikáciach môže interakcia užívateľa ovplyvniť mnoho rôznych nezlúčiteľných častí stavov. Z tohto dôvodu bola vymyslená architektúra, respektíve návrhový vzor Flux popísaný v zdrojoch [7] a [4].

Flux je aplikačná architektúra zložená zo 4 častí, ktoré určujú jednosmerný dátový tok. Radí sa skôr ku návrhovým vzorom ako frameworkom, keďže k nemu nie je potrebné mať mnoho ďalšieho kódu. Diagram je možné vidieť na obrázku 3.3.

**Dispatcher**, **Store** a **View** sú nezávislé uzly so zreteľnými vstupmi a výstupmi. Akcie sú zas jednoduché objekty obsahujúce dáta a typ akcie. Princíp je taký, že z **View** sa odošle akcia do uzla **Dispatcher**, ktorý rozhodne aké úložisko, teda **Store** sa má využiť. Ten nakoniec príjme túto akciu a rozhodne, aké zmeny stavov by mali nastať. Po tom ako sa zmeny aktualizujú sú zobrazené ako nový stav vo **View**.




Obrázek 3.3: Diagram architektúry Flux [9]

## 3.6 Databáza

Existuje niekoľko možností ukladania dát pre mobilnú aplikáciu. Medzi ne patria rôzne offline databázy ale aj zdieľané online služby, ktoré poskytujú cloud databázy na vzdialených serveroch.

V tejto aplikácií je potrebné uchovávať perzistentné dáta lokálne v zariadení, teda bez potreby prístupu k internetu. Perzistentné dáta su také, ktoré sa ukladajú do pamäti telefónu a ostanú uložené aj po vypnutí aplikácie, prípadne telefónu. React Native poskytuje API s názvom *AsyncStorage*, čo je vlastne jednoduché, nezašifrované, asynchrónne a perzistentné úložisko zároveň, ktoré sa dá využiť v celej aplikácii. AsyncStorage je na pozadí obsluhovaný natívnym kódom a ukladá hodnoty pod zvoleným kľúčom, čiže ako klasická *key-value*[12] databáza. Najčastejším spôsobom uloženia je serializovanie objektu do formátu JSON a následné uloženie pod dynamicky generovaným kľúčom. Príklad práce s úložiskom *AsyncStorage* je možné vidieť vo výpise 3.2.

```
let user_object1 = {
    name: 'Martin',
    age: 30,
    note: 'Test note'
};

AsyncStorage.setItem("User:id:1", JSON.stringify(user_object1));

AsyncStorage.getItem("User:id:1").then((value) => {
    console.log(value);
})

// log z~konzoly
// => {'name':'Martin','age':30,'note':'Test note'}
```

Výpis 3.2: Príklad serializovania a uloženia objektu pod kľúčom "User:id:1". Následne prebieha vytiahnutie objektu a vypísanie do konzoly.

[12]Key-value - https://en.wikipedia.org/wiki/Key-value_database

# Kapitola 4

# Návrh riešenia

V tejto kapitole je popísaná celá analýza, ktorá začínala definovaním problému, prieskumom cieľovej skupiny a následne definovaním užívateľských potrieb[17]. Ďalej je tu popísaný návrh užívateľského rozhrania, ktorý vychádzal z tejto analýzy, dátového modelu a spôsobu vyhľadávania fotiek ľudí.

## 4.1 Definícia problému

V dnešnej rýchlej dobe je pre veľa ľudí náročné nájsť si čas na rôzne veci v živote a preto sa každá ušetrená minúta počíta. Človek ako spoločenský tvor sa pravidelne s niekým stretáva, či už pracovne, alebo len tak vo svojom voľnom čase. Preto sa ukázalo ako vhodné rozšíriť aplikáciu z vyhľadania účastníkov schôdzky aj o menšiu správu schôdzok. Aplikácia sa bude teda zameriavať hlavne na pracovné stretnutia, konkrétne na ich uľahčenie.

Príprava na také pracovné stretnutie nie je vôbec jednoduchá, hlavne v prípade, že človek má daných stretnutí v jednom dni viac a má málo času na ich prípravu. Obzvlášť tieto problémy zažívajú hlavne manažéri a obchodníci, ktorí často cestujú a majú v krátkom časovom horizonte veľa schôdzok, prípadne ich majú hneď po sebe bez nejakých menších či väčších prestávok. Úlohou tejto bakalárskej práce bolo teda vytvoriť mobilnú aplikáciu pre maximálne zjednodušenie príprav na tieto stretnutia a rovnako aj efektívne zapisovanie poznámok na daných schôdzkach.

Hoci je táto aplikácia primárne určená pre vyššie spomenutú cieľovú skupinu, využije ju prakticky každý, kto sa niekedy zúčastní nejakého stretnutia, napr. pracovného pohovoru. Dĺžka takéhoto stretnutia môže presiahnuť aj jednu hodinu a je veľmi ťažké zapamätať si všetky potrebné informácie, ktoré by sa človeku mohli zísť, nehovoriac o tom, ak má človek daných stretnutí v jeden deň viac. Aplikácia bola teda navrhnutá na základe analýzy cieľovej skupiny a definície konkrétnych užívateľských potrieb s ohľadom hlavne na efektívnosť a rýchlosť pri príprave na schôdzku a následné zaznamenanie informácií z účasti na schôdzke.

## 4.2 Prieskum cieľovej skupiny

Pred návrhom efektívneho užívateľského rozhrania je nutné si v prvom rade uvedomiť, kto bude cieľovou skupinou, aby mohla byť aplikácia správne navrhnutá potrebám danej cieľovej skupiny. Na základe analýzy problému je možné ľahko definovať cieľovú skupinu, ktorou budú všetci ľudia nad 18 rokov, hlavne ale manažéri a obchodníci, ktorí majú časté

schôdzky. Do tejto cieľovej skupiny bude však spadať každý, kto už niekedy bol na nejakom stretnutí.

### Persony

Táto persona predstavuje typického užívateľa aplikácie, obchodníka Tomáša.

- **Meno:** Tomáš
- **Vek:** 54 rokov
- **Rodinný stav:** ženatý
- **Zamestnanie:** obchodník s 10 ročnou praxou
- **Bydlisko:** Brno
- **Záľuby a obľúbené veci:** Tomáš rád trávi čas so svojou rodinou a chodí do prírody. Rovnako však nezanedbáva svoju prácu, ktorá ho baví. Rád využíva svoj iPhone a chce využiť jeho plný potenciál, preto si do neho zapisuje všetky informácie ohľadom daných stretnutí. Je to tiež praktickejšie a rýchlejšie ako nosiť a používať pri týchto schôdzkach notebook. Stretáva sa väčšinou s 5-10 ľuďmi a v priemere má za týždeň 4 takéto stretnutia. Taktiež má Tomáš veľký záujem o nové technológie.

Druhá persona zobrazuje Ivanu, ktorá si hľadá prácu.

- **Meno:** Ivana
- **Vek:** 27 rokov
- **Rodinný stav:** slobodná
- **Zamestnanie:** bez práce
- **Bydlisko:** Bratislava
- **Záľuby a obľúbené veci:** Ivana rada číta a športuje. Je zvyknutá využívať počítač na prácu a v osobnom živote využíva hlavne smartphone a občas na zábavu aj svoj tablet. Aplikácie na smartphone využíva najviac pre komunikáciu s priateľmi.

### Scenáre použitia

Uvažujme používateľa aplikácie Tomáša.

- Tomáš ako obchodník neustále pracuje so zákazníkmi a chodí od jedného stretnutia k druhému. Má teda menej času na zisťovanie si bližších informácií o konkrétnych ľuďoch, ktorí majú byť s ním na stretnutí. Častokrát nevie koho očakávať a teda by mu mohla táto aplikácia zjednodušiť prípravu na tieto stretnutia. Rovnako by mu pomohlo, ak by si mohol zapísať výsledok stretnutia, prípadne nejaké informácie k účastníkom stretnutia a neskôr sa k nim vrátiť.

Uvažujme teraz používateľa Ivanu.

- Keďže Ivana je momentálne bez práce, snaží sa nejakú nájsť a chodí po rôznych pohovoroch. Keď si človek hľadá pracu, neustále sleduje nové ponuky, ktoré ho zaujali, posiela životopisy a následne ho pozývaju na pohovory. V prípade, že má človek niekoľko pohovorov v priebehu týždňa, prípadne dňa, je obzvlášť tažké sa na tieto stretnutia pripraviť. Vopred naštudované informácie o firme a ľuďoch, s ktorými má mať pohovor sa mu môžu pliesť, a preto by bolo vhodné pre neho možnosť dohľadať si tesne pred pohovorom jednoducho a rýchlo tieto informácie. Počas pohovoru, prípadne po ňom si zas ďalšie poznámky zapisovať. Takéto niečo by nakoniec mohlo pomôcť Ivane rozhodnúť sa pre správny výber práce, keďže bude mať zaznamenané aj dojmy a vlastné pocity o ľuďoch z firmy, ktorí sa zúčastnili pracovného pohovoru.

### Užívateľské potreby

Na základe analýzy užívateľov, vytvorených person a zobrazených scenárov použitia je možné definovať nasledujúce užívateľské potreby, ktoré sú zoradené podľa priority.

1. Aplikácia bude mať možnosť na základe zadaných mien vyhľadať fotky ľudí a prípadne nejaké informácie o nich.
2. Rýchle vyhľadanie a zobrazenie daných informácií na čo najmenej krokov, aby užívatelia ušetrili čas.
3. Pripojenie na internet bude potrebné len pre vyhľadanie ľudí, inak musí aplikácia pracovať offline.
4. Užívateľ bude môct zobraziť najbližšie stretnutia podľa dátumu.
5. Užívateľ bude mať možnosť rýchleho pridania poznámok k daným ľuďom v rámci stretnutia.
6. Užívateľ bude môcť rýchlym spôsobom pridať užívateľa ku schôdzke.
7. Užívateľ bude mať možnosť zobraziť a doplniť informácie o danej osobe.
8. Užívateľ bude mať možnosť zobraziť a upraviť informácie o schôdzke.
9. Musí byť dohľadateľná kompletná história schôdzok so všetkými poznámkami a účastníkmi.

## 4.3 Návrh dátového modelu

Jednou z najzákladnejších častí vývoja aplikácie je navrhnúť vhodný dátový model, ktorý zobrazuje štruktúru a spôsob ako budú uchovávané dáta vo vybranej databáze. Táto štruktúra vychádza z užívateľských potrieb a návrhu užívateľského rozhrania, na základe ktorého bolo definované, aké dáta je potreba užívateľom zobraziť. Na obrázku 4.1 je zobrazený ER diagram[1] tejto aplikácie.

[1] Entity-relationship diagram - https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model

**Entita** *User*

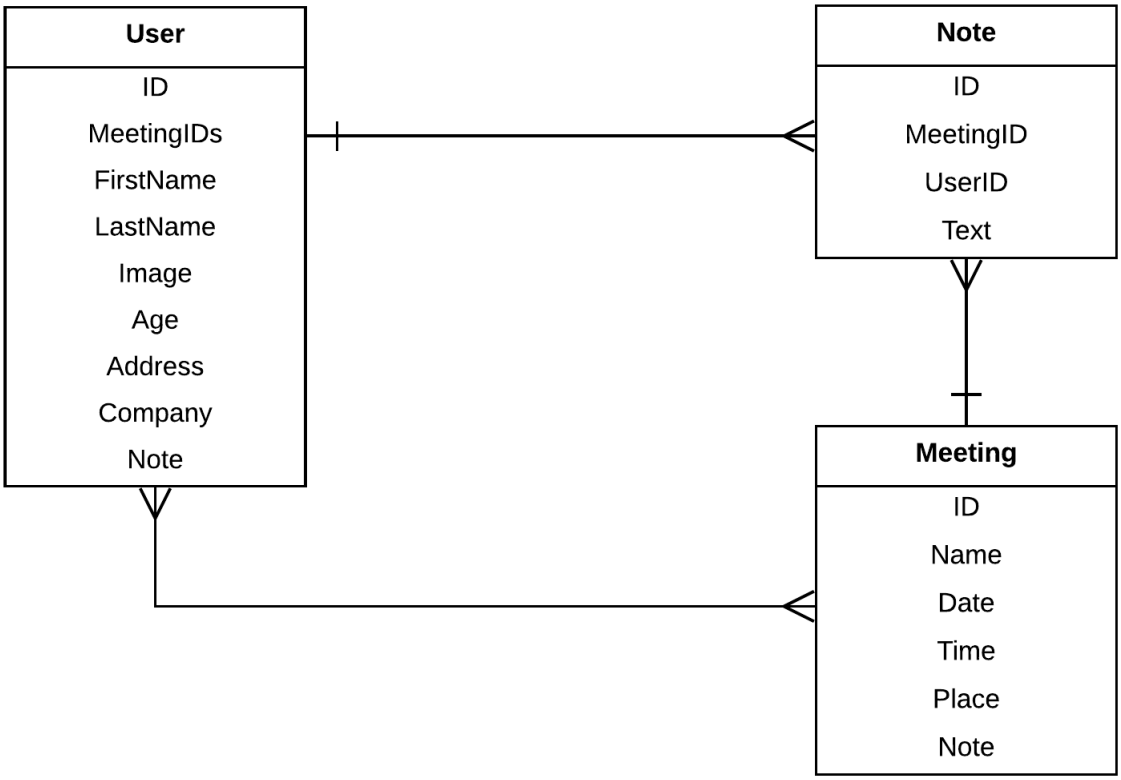
Základná entita *User* zobrazuje informácie, ktoré sa budú ukladať o užívateľovi. Medzi základné údaje patrí meno, priezvisko, obrázok, vek, adresa, názov spoločnosti či poznámka. Každý užívateľ musí mať pridelené unikátne ID, ktoré ho odlišuje od ostatných. Okrem toho bude uchovávať ešte ID každej jednej schôdzky, ku ktorej je priradený.

**Entita** *Meeting*

Tu sú zobrazené všetky potrebné informácie schôdzky, začínajúc unikátnym ID. Ďalej sa tu uchováva názov, dátum s časom, miesto a poznámka o schôdzke.

**Entita** *Note*

Entita *Note* uchováva poznámku, ktorá je priradená k užívateľovi a zároveň aj ku schôdzke. Potrebuje teda uchovávať okrem vlastného ID aj ID schôdzky a užívateľa, ku ktorým bola poznámka vytvorená.



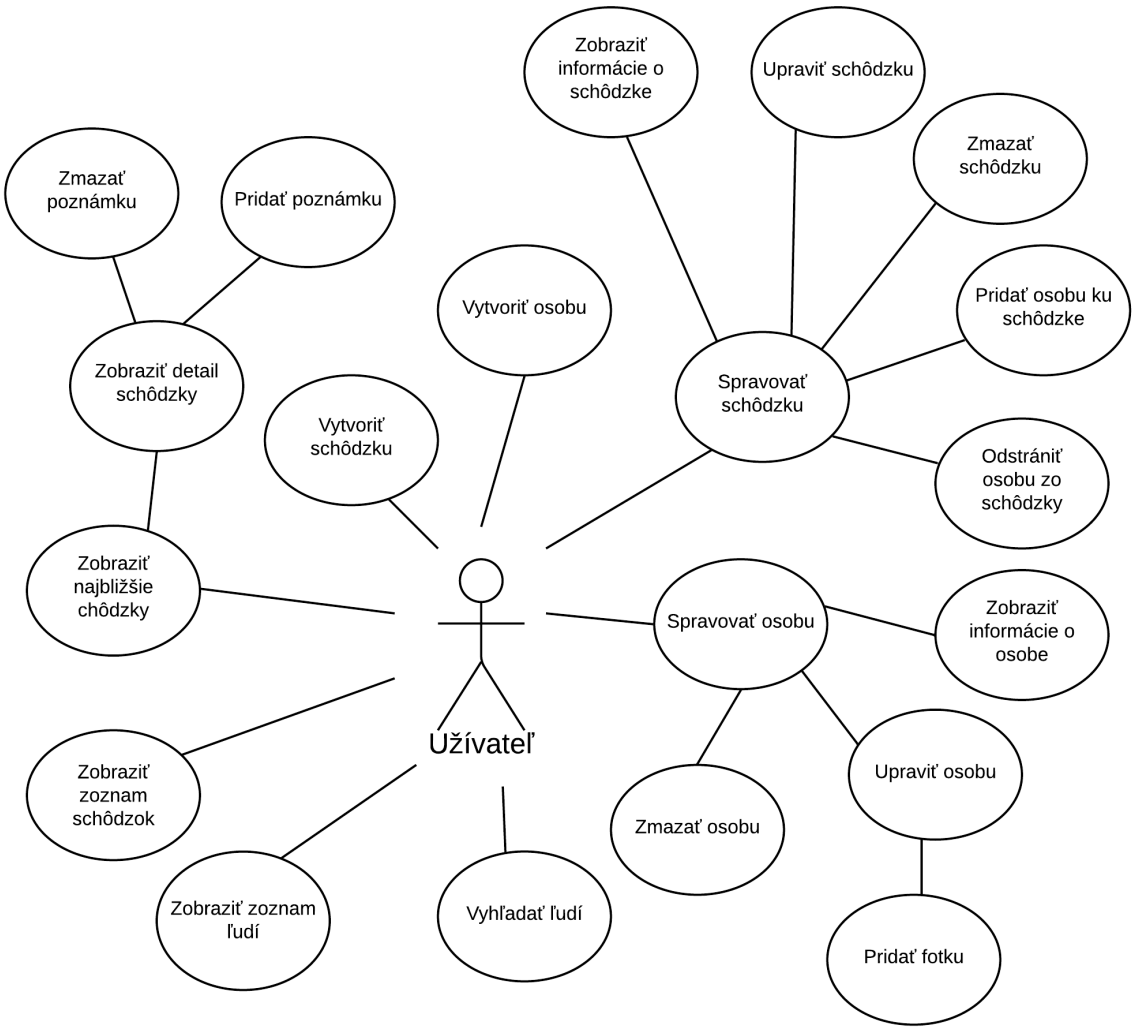

Obrázek 4.1: ER diagram - dátový model

## 4.4 Návrh užívateľského rozhrania

Aplikácia by okrem funkčnosti mala poskytnúť aj dostatočne intuitívne, jednoduché a použiteľné rozhranie. Výsledkom analýzy hlavných cieľov aplikácie a užívateľských potrieb je nasledujúci návrh užívateľského rozhrania. Každá navrhnutá obrazovka a jej prvky vychádzajú z užívateľských požiadavkov a riešia určité vopred definované problémy.

### Prípady použitia

Na obrázku 4.2 je zobrazený diagram prípadov použitia, ktorý je navrhnutý jazykom UML[2]. Tento diagram zobrazuje základné operácie, ktoré by mala aplikácia podporovať na základe definovaných užívateľských potrieb. Tieto operácie by sa mali premietnuť do funkčnosti a jednotlivých prvkov v užívateľskom rozhraní.




Obrázek 4.2: Diagram prípadov použitia

[2]UML - https://sk.wikipedia.org/wiki/Unified_Modeling_Language

Štruktúra aplikácie je členená do troch hlavných obrazoviek, ktoré sa dájú prepínať pomocou menu, ktoré sa nachádza v spodnej časti každej z nich a je konzistetné v rámci celej aplikácie. Sú to nasledovné obrazovky:

- Najbližšie schôdzky
- Zoznam schôdzok
- Zoznam ľudí

### Najbližšie schôdzky

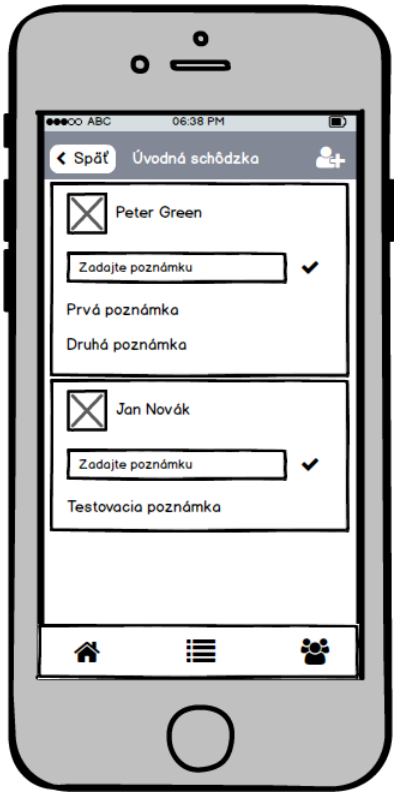
Z užívateľského požiadavku bolo potrebné navrhnúť jednoduché a prehľadné zobrazenie najbližších stretnutí a umožnenie pridávania poznámok k nim. Preto vstupnou obrazovkou sú práve **Najbližšie schôdzky** (obrázok 4.3), ktoré sa zobrazia hneď po spustení aplikácie. Je to z dôvodu, aby boli najbližšie stretnutia veľmi ľahko dostupné. Nachádzajú sa na nej teda karty stretnutí zoradených podľa dátumu zhora nadol. Jednotlivé stretnutia sú prehľadne oddelené dátumami ku ktorým patria, takže si ich nie je možné pomýliť. V rámci každého stretnutia je k dispozícii jeho názov, pod ktorým sa znovu nachádza dátum stretnutia. Ku každej karte stretnutia sú pripísané ešte tri dôležité informácie a to sú:

- čas
- miesto stretnutia
- počet ľudí

Všetky hlavné informácie o danom stretnutí sa nachádzajú priamo na tejto obrazovke, takže nie je nutné otvárať ďalšiu obrazovku s bližšími informáciami o stretnutí.

Ďalej z užívateľských požiadavkov vyplýva rýchle pridávanie poznámok k ľuďom v rámci aktuálneho alebo blízkeho stretnutia, ktoré sa má uskutočniť. Preto po otvorení detailu stretnutia (obrázok 4.4) na tejto obrazovke je zobrazený výpis ľudí, ktorí sa schôdzky zúčastnia a je tu navrhnutý spôsob rýchleho pridávania, zobrazenia a mazania poznámok. Taktiež bolo potrebné navrhnúť pridávanie ľudí priamo z tohto pohľadu, ktoré je umožnené tlačidlom v pravej hornej časti obrazovky.


Obrázek 4.3: Najbližšie stretnutia



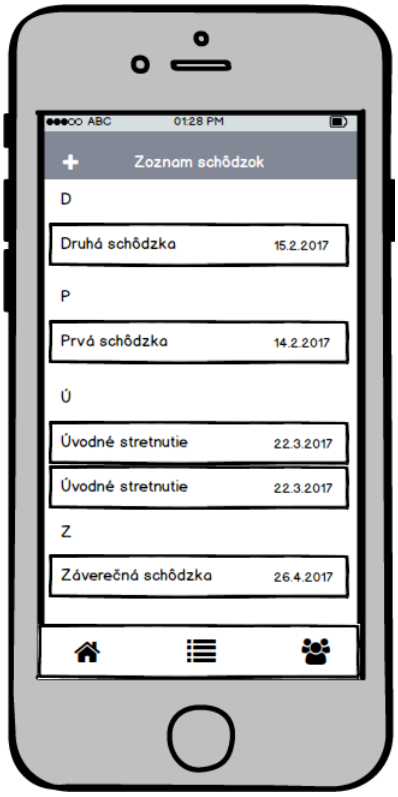

Obrázek 4.4: Detail najbližších stretnutí

### Zoznam schôdzok

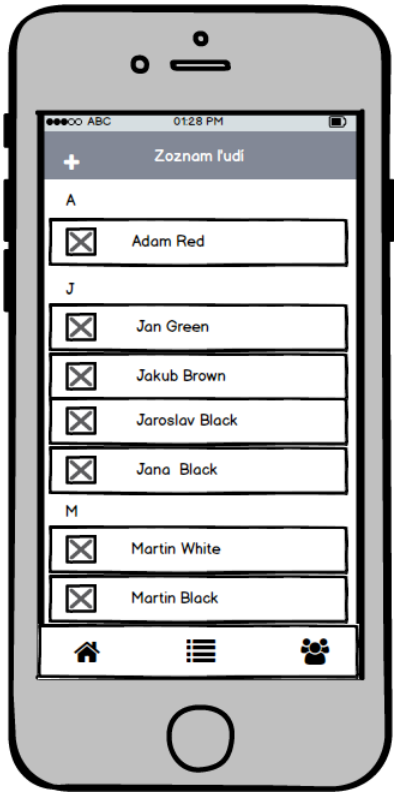
Na základe užívateľského požiadavku bolo potrebné pridať kompletnú históriu schôdzok nezávisle od dátumu. Najlepší spôsob ako toto vyriešiť bol vytvoriť zoznam všetkých schôdzok zoradených kvôli prehľadnosti podľa abecedy. Táto obrazovka na obrázku 4.5 je vlastne presný opak obrazovky **Najbližšie schôdzky**. Nachádzajú sa tu len abecedne zoradené schôdzky s dátumom, bez ďalších nepotrebných informácii. Úlohou tohto zoznamu je zobraziť čo najviac schôdzok na obrazovke.

### Zoznam ľudí

Táto obrazovka zobrazená na obrázku 4.6 je veľmi podobná s predchádzajúcou, kvôli konzistentnosti a nutnosti zobrazenia všetkých osôb. Jej štruktúra je stavaná rovnako, pretože ide tiež o zoznam. Tentokrát sú však abecedne zoradené a zobrazené všetky osoby uložené v aplikácií. Pri každej osobe sú zobrazené len dôležité informácie pre zoznam, a to je zmenšený náhľad fotky s menom a priezviskom.




Obrázek 4.5: Abecedný zoznam schôdzok

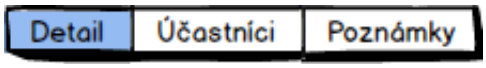



Obrázek 4.6: Abecedný zoznam ľudí

### Detail schôdzky a osoby

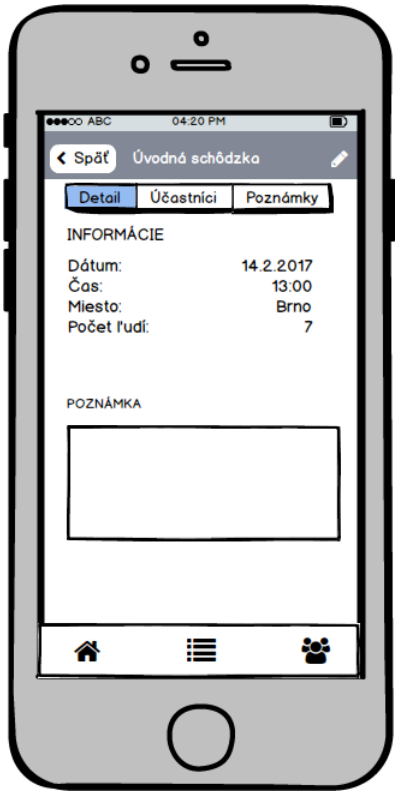
V užívateľských potrebách bola definovaná potreba zobrazenia bližších informácií o schôdzkach a o osobách. Taktiež bolo potrebné zobraziť zoznam zúčastnených osôb nejakej schôdzky a výpis poznámok. Všetky tieto informácie je teda nutné nejakým jednoduchým a prehľadným spôsobom zobraziť na jednej obrazovke.

Ako spôsob riešenia boli využité tzv. **Taby** (obrázok 4.7), čo sú vlastne horizontálne záložky, ktoré poskytujú konzistentnú navigáciu medzi obrazovkami. Môžu obsahovať text, ikonu alebo ich kombináciu. Týmto spôsobom je možné rozdeliť jednu obrazovku ľahko, rýchlo a prehľadne na viac častí. Zobrazenie detailu schôdzky, jej účastníkov či zoznamu poznámok je za pomoci tabov bezproblémové. Takýto spôsob navigácie poznajú užívatelia operačných systémov iOS a rovnako aj Android.




Obrázek 4.7: Ukážka tabov schôdzky

Jednotlivé detaily schôdzky, obrázok 4.8, a osoby na obrázku 4.9 prehľadne zobrazujú rozšírené informácie o danej schôdzke prípadne osobe aj s poznámkou. Následná úprava či zmazanie týchto údajov je možné pomocou ikony v pravej časti hlavičky.




Obrázek 4.8: Detail schôdzky


Obrázek 4.9: Detail osoby

### Vyhľadávanie a pridávanie ľudí

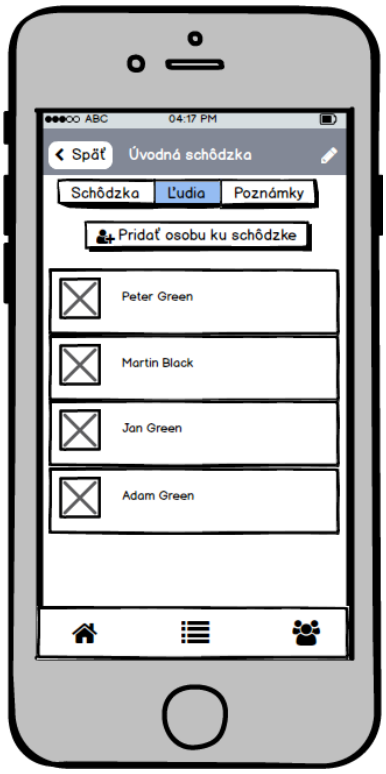
Vyhľadávanie ľudí je najdôležitejšou funkciou aplikácie. Z užívateľských požiadavkov je známe, že bolo potrebné vymyslieť a navrhnúť spôsob zadávania hľadaných ľudí a zobrazenia výsledkov s fotkami a informáciami o ľuďoch. Taktiež bolo nutné vymyslieť rýchle pridávanie týchto ôsob ku schôdzke na čo najmenej krôkov, kvôli ušetreniu času.

Z tohto dôvodu bola navrhnutá obrazovka na obrázku 4.10. Je to jeden z tabov každej schôdzky, kde je možné vidieť všetkých jej účastníkov v prípade, že nejakých má. Pridať ľudí ku schôdzke je možné práve z tohto miesta pomocou jednoduchého tlačidla, ktoré sa nachádza na vrchu zoznamu účastníkov. Po rozkliknutí tohto tlačidla sa zobrazí obrazovka 4.11.

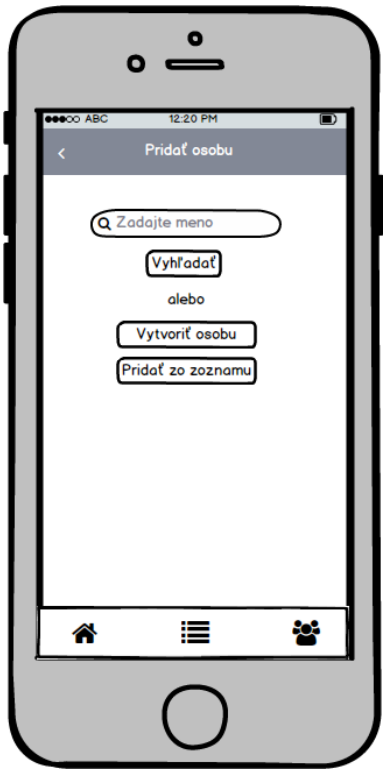
Táto obrazovka rieši problém pridávania ľudí ku schôdzke a ich vyhľadanie, prípadne vytvorenie. Návrh tejto obrazovky ponúka niekoľko funkcionalít. Prvou z nich je vyhľadanie osoby na internete jednoduchým zadaním mena a priezviska do textového poľa. Štruktúru výsledkov hľadania zobrazuje obrázok 4.12. Nachádzajú sa na ňom vyhľadané osoby s fotkami, menom a prípadne nejakými ďalšími informáciami. Z dôvodu nutnosti zobrazenia čo najviac osôb na jednej obrazovke sú tieto výsledky hľadania riešené pomocou zoznamu. Užívateľ si následne môže vybrať jednu z týchto osôb, ktorú chce pridať ku schôdzke. V prípade, že želanú osobu sa nepodarilo nájsť, je možné využiť tlačidlo pre manuálne vytvorenie osoby zadaním mena a priezviska, prípadne ďalších rozšírených informácií.

Posledné, tretie tlačidlo na tejto obrazovke zobrazuje zoznam už vopred vytvorených ľudí, ktorých je možné pridať ku schôdzke.

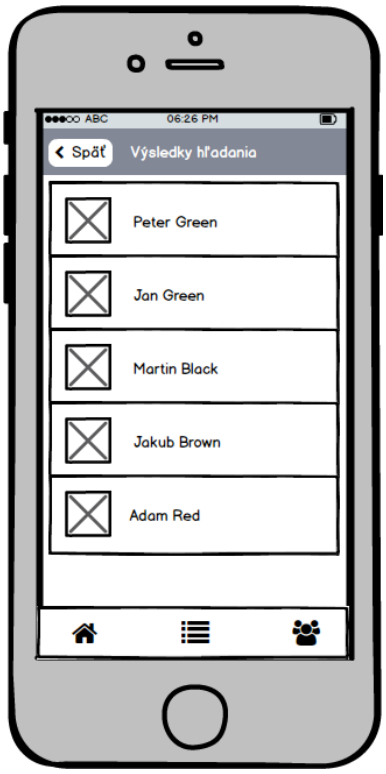
Z dôvodu prehľadnosti bolo nutné oddeliť vyhľadávaciu časť s textovým poľom od ďalších tlačidiel s inou funkcionalitou pomocou slova *"alebo"*. Užívatelia by si totiž mohli myslieť, že toto textové pole sa dá využiť aj pri manuálnom vytváraní osôb prípadne pri pridávaní zo zoznamu.




Obrázek 4.10: Zoznam účastníkov




Obrázek 4.11: Pridávanie ľudí ku schôdzke




Obrázek 4.12: Výsledky hľadania

## 4.5 Návrh vyhľadávania fotiek ľudí

Posledným krokom návrhu je vybrať spôsob a navrhnúť vhodné riešenie pre funkciu, ktorá má za úlohu vyhľadanie ľudí s fotkami na internete. V kapitole 2.2 sú popísané a zanalyzované rôzne platformy a API, pomocou ktorých je takéto vyhľadanie uskutočniteľné.

Prvý návrh spočíval vo využití Graph API od Facebooku, ktoré ponúkalo Search API s koncovým bodom **search**. Výsledky, ktoré vracia toto API v základných povoleniach od Facebooku stačili pre potreby tejto aplikácie. Pre verejnosť bolo dostupné meno, priezvisko, fotka a lokácia. Do volania API teda stačilo predať reťazec obsahujúci meno s priezviskom a access token, ktorý sa generuje pomocou OAuth.2.0 protokolu prihlásením do vlastného Facebook účtu. S týmto prihlasovaním bolo veľmi nápomocné SDK Expo popísané v podkapitole 3.3, pretože už vo svojom základe obsahuje API pre podporu Facebook prihlásenia. Facebook však svoje Search API kvôli veľkým zmenám v ochrane užívateľských dát dňa 4.4.2018 zrušil[3] a bolo potreba hľadať iné riešenie.

[3] https://developers.facebook.com/docs/graph-api/changelog/breaking-changes#search-4-4

Ako druhé riešenie bolo zvolené využitie Twitter API, ktoré je popísané v podkapitole 2.2. Toto API ponúkalo vyhľadávanie taktiež na základe mien a ako výsledky ponúkalo rovnaké údaje ako Graph API, čiže meno, priezvisko, fotka a lokácia pre ľudí, ktorí ju majú vyplnenú. Pre využitie nepotrebovalo ani prihlásenie do Twitter účtu, keďže všetky potrebné kľúče sa generovali v online vytvorenej aplikácií na vývojárskej stránke Twitteru. Z tohto dôvodu bolo lepšie využiteľné pre túto aplikáciu.

### Vyhľadanie a spojenie ľudí z viacerých zdrojov dát

V prípade existencie viacerých zdrojov dát (napríklad z viacerých sociálnych sietí), by bolo jednou z možností agregácia dát pre získanie viacerých informácií o určitej osobe, keďže každý zdroj môže ponúknuť iné informácie. Agregovať by sa mohlo napríklad na základe fotografie využitím nejakej služby pre porovnávanie dvoch fotografií. Týmto spôsobom by sa porovnali fotografie ľudí z viacerých zdrojov a v prípade nájdenia nejakej zhody by sa údaje mohli zjednotiť.

Takýto spôsob však vyžaduje oveľa viac komplexnejšie riešenie a nie je súčasťou ani cieľom tejto aplikácie, keďže aplikácia mala ponúknuť aj určitú správu vyhľadaných osôb a schôdzok.

# Kapitola 5

# Implementácia, testovanie a vyhodnotenie

Pre implementáciu tejto aplikácie som sa rozhodol použiť framework React Native, ktorý je popísaný v podkapitole 3.2 a bolo použité vývojové prostredie WebStorm.

V tejto kapitole budem popisovať dôležité časti vývoja aplikácie vrátane implementácie užívateľského rozhrania. Ďalej budú popísané použité nástroje a knižnice a nakoniec testovanie aplikácie spolu s vyhodnotením testov.

## 5.1 Využitie architektúry Flux

Využitie architektúry Flux popísanej v podkapitole 3.5, je v tejto aplikácií kľúčové pre správu stavov a úložiska. V zdrojových kódoch v zložke **flux** sa nachádzajú tri zložky, ktoré predstavujú hlavné časti aplikácie a každá z nich potrebuje vlastné úložisko, ktoré je spravované pomocou Fluxu. Tieto časti sú `Meeting (schôdzka)`, `Note (poznámka)`, `User (užívateľ)`. Ďalej sa v zložke flux nachádza ešte súbor `Dispatcher`, ktorý rozosiela akcie do týchto jednotlivých častí.

Každá z týchto častí obsahuje nasledujúce 4 súbory. Postupne budú popísané a ukázané na príklade schôdzky.

- MeetingActions
- MeetingConstants
- MeetingItem
- MeetingStore

### MeetingActions

V súbore `MeetingActions` sa nachádzajú všetky dostupné akcie schôdzok, ktoré su volané z jednotlivých komponent. Tieto komponenty predstavujú uzol **View** v diagrame 3.3. Akcia je jednoduchá funkcia, ktorá môže predstavovať napríklad vytvorenie a aktualizáciu údajov schôdzky (výpis 5.1). Ako parameter funkcie vstupujú dáta objektu schôdzky. Z tejto funkcie sa zavolá `Dispatcher`, ktorému sú predané dáta a typ akcie, čo je vlastne jednoduchá konštanta.

```
function createOrUpdateMeetingItem (data) {
    Dispatcher({
        type: MeetingConstants.MEETING_CREATE_UPDATE,
        data: data,
    });
}
```

Výpis 5.1: Akcia pre vytvorenie a aktualizáciu schôdzky.

### MeetingConstants

Tu sa nachádzajú jednotlivé konštanty, ktoré využíva flux pri schôdzkach. Ide o konštanty akcií ako napríklad vytvorenie alebo zmazanie schôdzky, ale nachádza sa tu aj konštanta `MEETING_EVENT_CHANGE`, ktorá zohráva dôležitú úlohu spúšťaní nejakej udalosti, teda eventu[1]. Taktiež sa tu nachádza kľúč, pod ktorým sa ukladajú objekty do úložiska *AsyncStorage*.

### MeetingItem

V tomto súbore sa nachádza trieda, na ktorú sa mapujú všetky objekty vyťahované z *AsyncStorage*. Obsahuje teda len gettery, čo sú metódy vracajúce konkrétny jeden parameter. `MeetingItem` obsahuje napríklad gettery ako `getId()`, `getName()`, `getDate()`, `getTime()`, `getPlace()` a `getNote()`.

### MeetingStore

*MeetingStore* je jedna z najdôležitejších častí. Je to vlastne konštanta obsahujúca funkcie, ktoré priamo pracujú s *AsyncStorage*. Nachádzajú sa tu funkcie pre prácu s úložiskom popísané v podkapitole 5.2. Jednotlivé funkcie dokážu vyťahovať objekty z úložiska podľa rôznych parametrov napr. ID osoby či ID schôdzky, prípadne ich kombináciou.

Ďalej sa tu nachádza funkcia `dispatchIndex()`, ktoré je volaná z dispatcheru a jej parameter je tzv. "payload", ktorý obsahuje typ akcie a dáta. Na základe tohto typu sa v konštrukcii `switch`, ktorá sa tu nachádza, rozhodne aká akcia sa má vykonať a zavolá sa príslušná funkcia, napríklad pre vytvorenie schôdzky.

### Listenery a eventy

Dôvod, pre ktorý bol flux vymyslený je práve pre jeho použitie a správu stavov pomocou listenerov a eventov. V `MeetingStore` sa nachádzajú ešte 3 ďalšie dôležité funkcie a tými sú `addChangeListener()`, `removeChangeListener()` a `emitChangeListener()`. Funkcia `addChangeListener()`, ako vyplýva už z názvu, pridáva listener na nejaký event. To znamená, že ak dôjde k vyvolaniu eventu, v tomto prípade názov eventu nesie konštanta `MEETING_EVENT_CHANGE`, tak sa vykoná určitý `callback`, teda nejaká funkcia predaná ako parameter. Druhá funkcia `addChangeListener()` len odstraňuje počúvanie, teda reakciu na daný event. Treťou funkciou je `emitChangeListener()`, ktorá priamo konkrétny event spustí.

V tejto aplikácii to funguje nasledovným spôsobom. Pri načítavaní komponenty, ktorá napríklad zobrazuje detail schôdzky sa zaregistruje listener, ktorý v prípade spustenia eventu znovu načíta a zobrazí informácie o schôdzke. Tento event sa spustí po aktualizovaní

[1] Event - https://en.wikipedia.org/wiki/Event_(computing)

a upravení informácii o schôdzke. Týmto spôsobom sa zaistí, že vždy po upravení informácii v schôdzke sa tieto aktualizované informácie zobrazia a prenesú do všetkých komponent a obrazoviek, ktoré majú zaregistrovaný listener na daný event.

## 5.2 Práca s databázou

Každá schôdzka, osoba a poznámka sa ukladá do systémového úložiska AsyncStorage (3.6). Všetky volania asynchrónnych metód z AsyncStorage sa nachádzajú v jednotlivých triedach `Store`, pomocou ktorých sú obsluhované.

Do úložiska sa ukladajú priamo objekty, ktoré sú serializované do formátu JSON. Ako kľúč, pod ktorým sa jednotlivé objekty ukladajú, je využitá konštanta `STORE_KEY_ITEM` obsahujúca reťazec zložený z názvu triedy, ukladaného objektu, slova "id" a konkrétneho ID čísla objektu. Všetky tieto časti reťazca sú oddelené dvojbodkou. Výsledný reťazec môže vyzerať nasledovne `@MeetingStore:meeting:id:1`.

Pre vytiahnutie objektu z úložiska sa využívajú asynchrónne metódy, ako je možné vidieť vo výpise 5.2.

```
async getItemById (id) {
    return await AsyncStorage.getItem(MeetingConstants.STORE_KEY_ITEM + id)
        .then(result => _mapToItem(JSON.parse(result)))
}
```

Výpis 5.2: Príklad metódy pre vytiahnutie schôdzky podľa ID.

Každý objekt sa vyťahuje pomocou konštanty `STORE_KEY_ITEM`, ku ktorej sa pridá ID objektu. Vytiahnutý objekt je namapovaný pomocou nasledujúcej metódy 5.3. Tá vráti objekt konkrétnej triedy ku ktorej patrí. V tomto prípade je to trieda `MeetingItem`.

```
function _mapToItem (obj) {
    return _.assign(new MeetingItem(), obj)
}
```

Výpis 5.3: Metóda pre mapovanie objektov.

Pre vytiahnutie všetkých objektov z úložiska sa využíva metóda 5.4.

```
 async getAllItems () {
    return (await AsyncStorage.multiGet(await MeetingStore.keys()))
        .map(result => _mapToItem(JSON.parse(result[1])))
}
```

Výpis 5.4: Metóda pre vytiahnutie všetkých objektov.

Tá zavolá metódu `keys()` (5.5), ktorá na základe regulárneho výrazu ukrytého pod konštantou `STORE_KEY_REGEXP` (napr. `/@MeetingStore:meeting:id:.+/`) vráti všetky kľúče tohto úložiska, bez ohľadu na ID. Tie sa využijú vo volaní `AsyncStorage.multiGet()`, ktoré vytiahne a následne namapuje všetky objekty znova pomocou metódy `_mapToItem()`.

```
 async keys () {
    return (await AsyncStorage.getAllKeys())
        .filter(key => STORE_KEY_REGEXP.test(key))
}
```

Výpis 5.5: Metóda pre vytiahnutie všetkých kľúčov na základe regulárneho výrazu..

V prípade vytvárania alebo mazania objektov sa využívajú metódy z AsyncStorage `setItem()` a `removeItem()`.

## 5.3 Implementácia užívateľského rozhrania

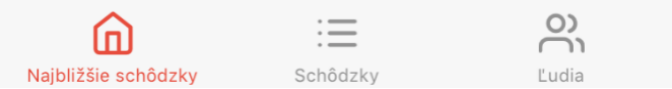
Užívateľské rozhranie tejto aplikácie sa skladá z veľkého množstva prvkov a komponent. Najdôležitejšie a najzaujímavejšie časti tohto rozhrania sú popísane v tejto podkapitole.

### Navigácia

Základom implementácie užívateľského rozhrania je navigácia. Pre implementáciu navigácie bol využitý balíček *React Navigation*. Tento balíček ponúka tri druhy navigácie a tými sú:

- TabNavigator
- StackNavigator
- DrawerNavigator

V tejto aplikácií sú využité prvé dva typy. Najskôr som rozmýšľal nad využitím `DrawerNavigator`, ale neskôr po lepšom premyslení som usúdil, že pre aplikáciu s takýmto počtom obrazoviek bude lepšie použiť `TabNavigator`. Všetky navigácie použité v aplikácii sa nachádzajú v zložke **navigations**. Navigácia použitá v hlavnom menu, je teda typu `TabNavigator` (obrázok 5.1) a nachádza sa v súbore `AppNavigation.` Pomocou tohto menu sa dá presúvať medzi troma hlavnými obrazovkami, ktoré sú zobrazené pomocou ikony a názvu obrazovky.




Obrázek 5.1: Ukážka TabNavigator

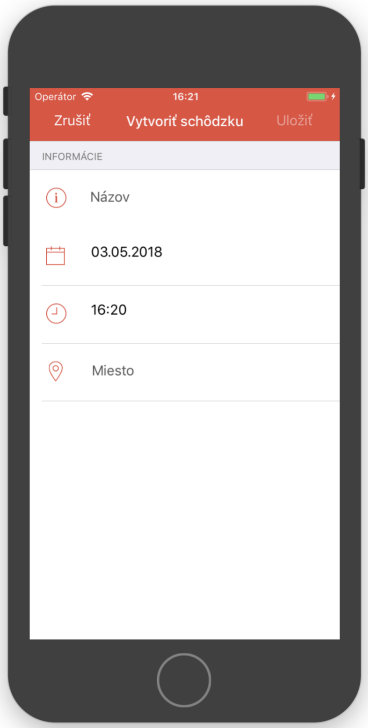
Každá z týchto troch obrazoviek má v tejto navigácii vnorený vlastný súbor so `Stack-Navigator` navigáciou. Pre **Najbližšie schôdzky** je to napríklad súbor `NextMeeting-Navigation`. Tento súbor musí obsahovať všetky odkazy na obrazovky (aj vnorené), ku ktorým sa dá z úvodnej obrazovky dostať. Ďalej tento typ navigácie, ako už z názvu vyplýva, využíva princíp zásobníku. To znamená, že sa jednotlivé obrazovky ukladajú za seba a pre vrátenie na pôvodnú obrazovku je nutné prejsť cez každú, predtým otvorenú obrazovku šípkou späť, ako je možné vidieť na obrázku 5.2.


Obrázek 5.2: Ukážka šípky späť pri StackNavigator

### Modálne okná

Modálne okná sa v tejto aplikácii používajú pri vytváraní alebo upravovaní osôb a schôdzok, či pri zobrazovaní výsledkov hľadania alebo vyberaní zo zoznamu ľudí. Sú využité z toho

dôvodu, aby užívateľ pri otvorenom modálnom okne nemal možnosť pracovať s inými časťami aplikácie a aby sa po zavrení nemal možnosť k nemu vrátiť, ako je to napríklad pri `StackNavigatore`. V aplikácií sú využité pri otváraní takéhoto okna viaceré animácie a prechody. Pri obrazovke vytvárania objektu sa používa prechod **slide**, čo znamená vysunutie modálneho okna zo spodu obrazovky. Pri úprave objektu je to zase prechod **fade**, ktorý označuje plynulé zmiznutie a zobrazenie novej obrazovky. Využitie konkrétne týchto prechodov bolo inšpirované natívnym rozhraním operačného systému iOS, ktorý ich využíva pri podobnej funkcionalite. Ukážka zobrazenia modálneho okna pre vytvorenie schôdzky je na obrázku 5.3.




Obrázek 5.3: Modálne okno vytvárania schôdzky

### Obrazovky a komponenty

Všetky obrazovky sú prehľadne štrukturované v zložke **screens**. Delia sa na dve hlavné časti - obrazovky osôb a obrazovky schôdzok, ktoré sú následne členené už na konkrétne typy podľa toho, čo zobrazujú alebo vykonávajú. Každý súbor obrazovky alebo komponenty obsahuje ešte súbor so štýlmi, ktoré sú veľmi podobné CSS štýlom.

Jednotlivé časti obrazoviek sa nazývajú komponenty a sú umiestnené v zložke **components**. Obrazovky sú delené na komponenty z toho dôvodu, že tieto menšie časti môžu byť znovupoužiteľné v iných obrazovkách a tým sa odstraňuje duplicita v zdrojovom kóde.

Header je jediná komponenta nachádzajúca sa na každej obrazovke. Predávajú sa mu parametre obsahujúce titulok obrazovky a jednotlivé ikony na ľavej a pravej strane. Pri renderovaní zoznamov sa využívajú vždy 2 komponenty. Komponenta `List`, napríklad `MeetingList` je vložená priamo do obrazovky `MeetingListScreen` a renderujú sa v nej

jednotlivé objekty schôdzok, ktoré predstavuje komponenta `MeetingListItem`. Príklad tejto komponenty je zobrazený vo výpise 5.6.

```
const MeetingListItem = ({ item, onPress }) => (
    <List>
        <ListItem button
            onPress={() => onPress(item.getId())}
            style={ styles.listItem }
        >
            <Body style={ styles.body}>
                <Text>{item.getName()}</Text>
            </Body>
            <Right style={ styles.right }>
                <Text note>{item.getDate()}</Text>
            </Right>
        </ListItem>
    </List>
);
```

Výpis 5.6: Príklad komponenty MeetingListItem

## 5.4 Vyhľadanie ľudí pomocou Twitter API

Celé spojenie a práca s TwitterAPI sa nachádza v aplikácií v zložke **services/TwitterApi**. Pre využitie tohto API bolo najskôr nutné založiť aplikáciu v Twitter Apps[2] a zadať jej názov a popis. Ďalej bolo treba vygenerovať kľúče pre spojenie s API. Tieto kľúče sú uložené v súbore `TwitterApiKeys` pod konštantami `CONSUMER_KEY`, `CONSUMER_SECRET`, `ACCESS_TOKEN` a `ACCESS_TOKEN_SECRET`.

Obsluha API sa nachádza v súbore `TwitterApiFetchService` kde jeho spojenie zabezpečuje balíček *React Native Twitter*. Pre vytvorenie klientského spojenia je potrebné predať všetky API kľúče (tokeny) ako parameter pre objekt `twitter` z tohto balíčka. Následne je možné vyhľadanie ľudí na základe zadaného reťazca. Ten sa posunie ako parameter koncovému bodu **users/search**, ktorý vráti vyhľadaných ľudí. Príklad obsluhujúcej funkcie je možné vidieť vo výpise 5.7

```
const {rest} = twitter(tokens);

async getUsers (term) {
    return rest.get('users/search', {q: term});
}
```

Výpis 5.7: Spojenie s API a vyhľadanie ľudí

Vyhľadanie ľudí sa zavolá v triede `UserSearchResultScreen`, ktorá aj zobrazuje výsledky. Hľadaný reťazec je do tejto triedy predaný ako parameter z triedy `UserSearchIndexScreen`, v ktorej sa nachádza textové poľe pre zadanie vyhľadávaného mena a prezviska. Výsledky sú vrátené vo formáte JSON a hneď po vyhľadaní prebieha ich filtrovanie na základe fotografie. V prípade, že užívateľ žiadnu fotografiu nemá, vo výsledkoch sa nezobrazí. Nakoniec sú tieto výsledky zobrazené pomocou komponenty `SearchResultList`, ktorá renderuje jednotlivé osoby komponentou `SearchResultListItem`. Okrem výsledného zobrazenia fotky, mena,

[2] https://apps.twitter.com

priezviska prípadne lokácie je možné taktiež otvoriť profil osoby na Twitteri, kde sa nachádzajú ďalšie informácie.

## 5.5 Použité nástroje a knižnice

V tejto sekcii sa nachádzajú nástroje a ďalšie knižnice, ktoré boli použité v tejto aplikácii.

### Nástroje

Základným nástrojom pre vývoj každej aplikácie je vývojárske prostredie. Pre implementáciu tejto aplikácie bolo využité **Webstorm IDE**[3] od spoločnosti JetBrains[4]. Je to moderné javascriptové IDE, ktoré bolo vybrané hlavne z dôvodu, pretože má podporu syntaxe React JSX.

Ďalším potrebným nástrojom bolo vývojové prostredie **Expo XDE**, ktoré sa používa spolu s SDK popísaným v podkapitole 3.3. V tomto vývojovom prostredí je možné spúšťať, reštartovať, zdieľať a vytvárať projekty na reálnom zariadení, v simulátore, či debugovať aplikáciu pomocou konzoly.

### Knižnice

Pre vytvorenie funkčnej aplikácie bolo potrebné využiť niekoľko balíčkov a knižníc, ktoré ponúkajú rôznu podpornú funkcionalitu. Aby bolo možné inštalovať a spravovať tieto balíčky bol v tejto aplikácií využitý **Node Package Manager (npm)**. Je to najväčší JavaScriptový správca balíčkov na platforme Node.

Pre úvodné vytvorenie React Native aplikácie sa využíva balíček **Create React Native App (CRNA)**, ktorý umožňuje najrýchlejšie a najjednoduchšie vytvorenie aplikácie bez akejkoľvek nutnej konfigurácie, alebo inštalácie nástrojov potrebných pre vývoj v natívnom kóde, ako sú Xcode alebo Android Studio. Pri použití **CRNA** je aplikácia napísaná len v čistom JavaScripte bez možnosti použitia natívneho kódu.

Navigácia a smerovanie v aplikácií medzi jednotlivými obrazovkami bolo riešené pomocou knižnice **React Navigation**. Táto knižnica ponúka tri základné typy natívnych navigácií a to StackNavigator, TabNavigator a DrawerNavigator s veľkými možnosťami konfigurácie a jednoduchej implementácie.

Kvôli využitej architektúre Flux, ktorá používa eventy sa v aplikácií využíva aj balíček **Bullet**. Je to jednoduchý a konzistentný systém komunikácie v rámci aplikácií. Využíva *pub/sub*[5] model ako formu asynchronnej komunikácie. V aplikácií sa používa na spúšťanie eventov architektúry Flux. Ikonky využité v aplikácií sú zase získané z balíčka **React Native Vector Icons**, ktorý obsahuje množstvo rôznych knižníc s vektorovými ikonami. V aplikácií sú použité ikony z knižníc Ionicons a Feather. Podporné funkcie, napríklad pre prácu s reťazcami alebo poliami sú získané z knižnice **Lodash**, ktorá je exportovaná z Node.js. Pre spojenie s Twitter API a vyhľadanie ľudí sa využíva balíček **React Native Twitter**, čo je vlastne Twitter API klient s rôznymi možnosťami.

[3] Integrated Development Environment – integrované vývojové prostredie
[4] https://www.jetbrains.com/
[5] Publish/subscribe

## 5.6 Testovanie a vyhodnotenie

Testovanie je dôležitou súčasťou pri vývoji aplikácie. Počas vývoja testovanie prebiehalo na emulátore Xcode v mobilnej aplikácii Expo. Rovnako bolo však využité aj reálne zariadenie iPhone 6s s touto nainštalovanou aplikáciou.

Proces testovania užívateľského rozhrania prebiehal počas vývoja s koncovými užívateľmi. Týmto užívateľom bola na začiatku krátko predstavená aplikácia a jej hlavný cieľ. Testovania sa zúčastnilo celkovo 10 užívateľov, ktorí dostali nasledujúce úlohy a boli pozorovaní pri ich plnení.

1. Vytvorte schôdzku s názvom Úvodná schôdzka, s dátumom a časom 17.5.2018, 12:00 v meste Brno.
2. Vytvorte osobu Ján Novák nasledujúcimi údajmi: vek 30, bydlisko Bratislava, firma 123 s.r.o.
3. Pridajte vytvorenej osobe fotku.
4. Vytvorenú osobu pridajte ku schôdzke.
5. Druhú osobu pridajte ku schôdzke pomocou vyhľadávača.
6. Napíšte k osobe v rámci schôdzky 1 poznámku.
7. Zobrazte schôdzky, ktorých sa zúčastnila daná osoba.
8. Zobrazte zoznam zúčastnených osôb nejakej schôdzky.
9. Upravte vek ľubovolnej osobe.
10. Zmažte nejakú schôdzku.

Počas týchto úloh prebiehalo pozorovanie práce užívateľa s aplikáciou so zameraním na zrozumiteľnosť a prehľadnosť návrhu grafického rozhrania. Pozorovala sa taktiež rýchlosť plnenia úloh a či sa užívatelia na niektorej z úloh nezasekli. Na konci testovania užívatelia dostali krátky dotazník, ohľadom užívateľského rozhrania. Graf výsledkov dotazníka je možné vidieť v podkapitole výsledky testovania.

## Výsledky testovania

Po dokončení testovania sa zisťovalo, ktoré úlohy robili najväčší problém a či sa užívatelia vedeli rýchlo zorientovať v aplikácii. Medzi najväčšie problémy, ktoré sa ukázali patrili nasledovné časti úloh:

- manuálne vytvorenie a pridanie osoby ku schôdzke
- orientácia na obrazovke pridávania osôb

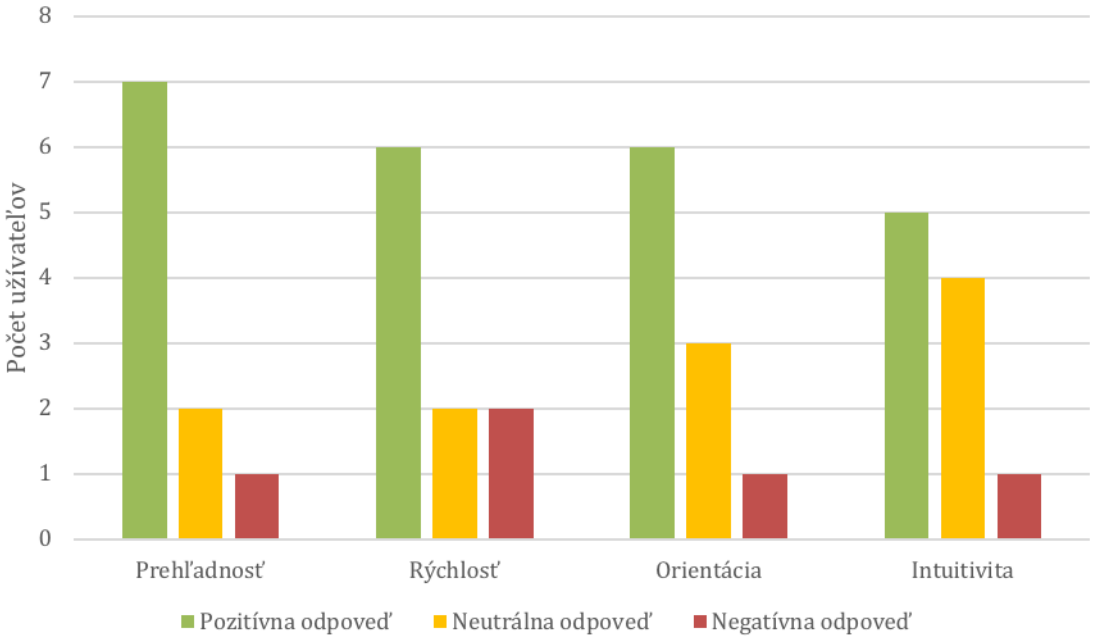
Tieto problémy sa ukázali takým spôsobom, že väčšina užívateľov sa na tejto úlohe dlhšie zdržala a nevedela nájsť vhodný spôsob ako ju vyriešiť. Prvý problém vznikol z toho dôvodu, pretože užívatelia presne nevedeli, že existuje funkcia, ktorá dokáže vytvoriť užívateľa a zároveň ho aj pridať ku schôdzke. Táto funkcia bola totiž k dispozícií len pri detaile obrazovky **Najbližšie schôdzky**, na ktorej sa zapisujú poznámky. Pre lepšie využitie však

bola pridaná aj k obrazovke, kde sa pridávajú a vyhľadávajú ľudia. Po tejto menšej úprave ju užívatelia vedeli ľahšie nájsť a lepšie využiť aj v prípade, keď zrovna nepísali poznámku.

Rovnako bolo niekoľko podnetov na celú obrazovku pridávania osoby ku schôdzke, kde si užívatelia často mýlili vyhľadávacie textové pole a využívali ho aj pri ďalších možnostiach pridávania ľudí, hoci pre inú funkcionalitu, ako vyhľadanie ľudí toto pole nefunguje. Tento problém bol vyriešený zmenou rozloženia tejto obrazovky a oddelenia vyhľadávača od ďalších možností pridávania osôb. Zvyšok testovania však dopadol pozitívne a pri plnení ďalších úloh užívatelia nenarazili na nejaké problémy.

Ďalšie postrehy užívateľov, ktoré som získal na základe ich spätnej väzby sa týkali výberu vhodných ikoniek a ich umiestnenia, prípadne premiestnenia niektorých zobrazovaných informácií na miesta, kde im dávajú väčší zmysel. Ukázalo sa totiž, že niektoré ikonky nie vždy zrozumiteľne odzrkadľovali funkcionalitu.

Všetky tieto pripomienky boli po testovaní zapracované do aplikácie. Počas testovania sa taktiež ukázalo niekoľko drobných chýb z hľadiska funkčnosti aplikácie, ktoré boli úspešne odstránené. Dostal som aj návrh na novú funkcionalitu, ktorá by zahrňovala *push notifikácie* s upozorneniami blížiacej sa schôdzky. Po dokončení testovania užívatelia vyplnili dotazník ohľadom celkovej intuitivity aplikácie. Väčšina udelených hodnotení ohľadom intuitivity aplikácie bola pozitívna príp. neutrálna. Z odpovedí, je teda možné usúdiť, že užívateľské rozhranie je možné pokladať za prehľadné a intuitívne, čo bolo vlastne cieľom návrhu tohto rozhrania.




Obrázek 5.4: Graf výsledkov dotazníka B

# Kapitola 6

# Záver

Cieľom tejto bakalárskej práce bolo navrhnúť a implementovať mobilnú aplikáciu pre platformu iOS, ktorá umožňuje vyhľadávanie ľudí, teda účastníkov schôdzky a ich fotiek na internete. Zadanie aplikácie bolo značne rozšírené aj o správu týchto schôdzok a ľudí, spolu s možnosťou písania rôznych poznámok a jej účelom je teda uľahčenie a zefektívnenie samotných schôdzok a prípravy na nich.

Na začiatku som preštudoval postup správneho návrhu mobilných aplikácií a rôzne možnosti pre vyhľadanie fotiek ľudí. Pre úspešnú implementáciu som si tiež musel vybrať niekoľko z mnoha technológií pre vývoj mobilných aplikácií a naštudovať ich.

Návrh aplikácie vychádza z definície problému a prieskumu cieľovej skupiny, ktorý bolo treba uskutočniť kvôli definovaniu užívateľských potrieb. Na základe týchto informácií bolo možné navrhnúť užívateľské rozhranie, ktoré spĺňuje tieto potreby.

Aplikácia bola následne tvorená za pomoci frameworku React Native v programovacom jazyku JavaScript a vývojovom prostredí WebStorm. Základ správneho objektového návrhu a využitiu rôznych architektúr, ktoré sa dajú jednoducho a prehľadne rozširovať som získal zo štúdia technológií v teoretickej časti.

Počas implementácie som sa stretol s niekoľkými problémami, hlavne pri rôznych obmedzeniach ohľadom používania API pre vyhľadanie ľudí. Najväčším problémom bolo, keď bolo vypnuté Graph API od Facebooku, ktoré som už mal integrované do aplikácie. Všetky problémy sa mi však podarilo úspešne vyriešiť a aplikácia mohla podstúpiť ďalšiu fázu, ktorou bolo testovanie a vykonanie experimentov. Vo výsledkoch testovania sa odrazil správny návrh užívateľského rozhrania, na základe po väčšine kladných odpovedí v dotazníkoch od testerov.

Z hľadiska budúcnosti aplikácie je možné rozšírenie vyhľadania ľudí o ďalšie zdroje, prípadne vylepšenie alebo pridanie rôznych ďalších informácií ku správe schôdzok a ľudí.

Výsledkom tejto bakalárskej práce je funkčná aplikácia, ktorá je otestovaná a spustiteľná na reálnom zariadení.

# Literatura


[1] 650 Industries, Inc: *Expo*. [Online; navštívené 28.04.2018].
URL https://expo.io/

[2] Facebook Inc.: *React - A JavaScript library for building user interfaces*. [Online; navštívené 28.04.2018].
URL https://reactjs.org/

[3] Facebook Inc.: *React Native · A framework for building native apps using React*. [Online; navštívené 28.04.2018].
URL https://facebook.github.io/react-native/

[4] Boduch, A.: *Flux Architecture*. Packt Publishing, 2016, ISBN 1786465817.

[5] Eisenman, B.: *Learning React Native: Building Native Mobile Apps with JavaScript*. O'Reilly, 2015, ISBN 1491929006.

[6] Facebook: *Graph API*. [Online; navštívené 28.04.2018].
URL https://developers.facebook.com/docs/graph-api

[7] Facebook Inc.: *Flux | Application Architecture for Building User Interfaces*. [Online; navštívené 28.04.2018].
URL https://facebook.github.io/flux/

[8] Felicia Rogers: *Design Thinking For The Rest Of Us by Felicia Rogers*. [Online; navštívené 28.04.2018].
URL https://www.decisionanalyst.com/blog/designthinking/

[9] Fullstack: *Fullstack React: Intro to Flux and Redux*. [Online; navštívené 28.04.2018].
URL https://www.fullstackreact.com/p/intro-to-flux-and-redux/

[10] GeekyAnts: *NativeBase | Essential cross-platform UI components for React Native*. [Online; navštívené 28.04.2018].
URL https://nativebase.io/

[11] Google Developers: *Custom Search | Google Developers*. [Online; navštívené 28.04.2018].
URL https://developers.google.com/custom-search/

[12] Google Developers: *Google+ Platform for Web | Google Developers*. [Online; navštívené 28.04.2018].
URL https://developers.google.com/+/web/

[13] Hartson, R.; Pyla, P. S.: *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*. Elsevier Science, 2012, ISBN 0123852412.

[14] LinkedIn Corporation: *Home | LinkedIn Developer Network*. [Online; navštívené 28.04.2018].
URL https://developer.linkedin.com/

[15] Pete Heard: *React Native Architecture : Explained!* [Online; navštívené 28.04.2018].
URL https://www.logicroom.co/react-native-architecture-explained/

[16] Twitter, Inc.: *Docs — Twitter Developers*. [Online; navštívené 28.04.2018].
URL https://developer.twitter.com/en/docs

[17] Unger, R.: *A project guide to UX design : for user experience designers in the field or in the making*. New Riders, 2012, ISBN 9780321815385.

# Příloha A

# Obsah DVD

Priložené DVD obsahuje:

- zdrojové kódy aplikácie
- text bakalárskej práce vo formáte PDF
- zdrojové súbory textu bakalárskej práce v systéme LaTeX
- súbor README.txt obsahujúci manuál k aplikácii
- prezentačné video
- plagát

# Příloha B

# Dotazník k testovaniu

1. Ako by ste zhodnotili prehľadnosť užívateľského rozhrania? (ako v škole)

   *Najlepšia* 1 2 3 4 5 *Najhoršia*

2. Ako by ste zhodnotili rýchlosť práce s aplikáciou? (ako v škole)

   *Najlepšia* 1 2 3 4 5 *Najhoršia*

3. Ako by ste zhodnotili obtiažnosť vykonávaných úloh?

   - Úlohy boli jednoduché
   - Niekedy som si nebol istý, ako danú úlohu vykonať
   - Úlohy boli obtiažne

4. Ako by ste zhodnotili orientáciu v aplikácii? (ako v škole)

   *Najlepšia* 1 2 3 4 5 *Najhoršia*

5. Ako by ste zhodnotili celkovú intuitivitu v aplikácii? (ako v škole)

   *Najlepšia* 1 2 3 4 5 *Najhoršia*

6. Čo by ste v aplikácií vylepšili príp. čo Vám v nej chýbalo?

# Příloha C

# Plagát