VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

MOBILNÍ APLIKACE PRO ZAJIŠTĚNÍ SPORTOVNÍCH AKTIVIT

MOBILE APPLICATION FOR SPORTS ACTIVITIES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ĽUBOŠ HLIPALA**
AUTHOR

VEDOUCÍ PRÁCE **Mgr. Ing. PAVEL OČENÁŠEK, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/21250/2017/xhlipa01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Hlipala Ľuboš**
Obor: Informační technologie
Téma: **Mobilní aplikace pro zajištění sportovních aktivit**
Mobile Application for Sports Activities
Kategorie: Počítačové sítě

Pokyny:

1. Nastudujte problematiku vyhledávání vhodných POI na mapě vzledem k jejich vzájemné vzdálenosti. Nastudujte Google Maps API a Android SDK a následně vyberte vhodné vývojové prostředí.
2. Analyzujte požadavky na mobilní aplikaci pro zajištění sportovních událostí, vzájemné komunikace účastnících se sportovců, vyhodnocování aktivit apod.
3. Na základě požadavků a dle instrukcí vedoucího práce mobilní aplikaci navrhněte.
4. Implementujte navrženou aplikaci.
5. Aplikaci otestujte v reálném prostředí.
6. Diskutujte získané výsledky a možnosti dalšího rozšíření.

Literatura:

- BEIGHLEY, Lynn. a Michael MORRISON. Head first PHP. Sebastopol, CA: O'Reilly, 2009. ISBN 978-0-596-00630-3.
- CHURCHER, Clare. Beginning database design. New York: Distributed to the book trade worldwide by Springer-Verlag New York, 2007. ISBN 978-1590597699.
- CALVERT, Kenneth L. a Michael J. DONAHOO. TCP/IP sockets in Java: practical guide for programmers. 2nd ed. San Francisco: Morgan Kaufmann Publishers, 2002. ISBN 978-1558606852.
- THORNSBY, Jessica. Android UI Design. Birmingham: Packt Publishing, 2016. ISBN 978-1785887420.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

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: **Očenášek Pavel, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Táto bakalárska práca popisuje tvorbu mobilnej aplikácie, ktorá má za úlohu osloviť užívateľov so spoločnou záľubou v športe a poskynúť im vhodnú platformu pre zdieľanie tejto záľuby. Naimplementovaný systém pozostávajúci z mobilnej aplikácie pre platformu Android a aplikačného servera špeciálne navrhnutého pre tento systém ďalej vytvára priestor pre vznik novej komunity užívateľov, ktorí medzi sebou zdieľajú informácie o svojich športových aktivitách a vytvárajú nové športové udalosti. V práci budú objasnené postupy a prístupy pri tvorbe tohto systému pomocou Android SDK, jazyka Java a MySQL databázy.

Abstract

This bachelor thesis describes a creation of a mobile application that aims to reach users with a common interest in sport and to provide them with a suitable platform for sharing this hobby. An implemented system consists of the mobile application for Android platform and especially designed application server, offers a creation of a user community, whose members can share information about their sport activities or create new sport events. This work also enlightens logic and approaches of the system creation process using Android SDK, Java and MySQL database.

Kľúčové slová

Android platfroma, Android SDK, klient, server, databáza, aplikačný protokol, bod záujmu, Google Maps API, mobilná aplikácia, šport

Keywords

Android platform, Android SDK, client, server, database, application protocol, point of interest, Google Maps API, mobile application, sport

Citácia

HLIPALA, Ľuboš. *Mobilní aplikace pro zajištění sportovních aktivit.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Mgr. Ing. Pavel Očenášek, Ph.D.

Mobilní aplikace pro zajištění sportovních aktivit

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pana Ing. Mgr. Pavla Očenáška PhD. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Ľuboš Hlipala
14. mája 2018

Poďakovanie

V tejto sekcii by som chcel poďakovať školiteľovi tejto práce pánovi Ing. Mgr. Pavlovi Očenáškovi PhD. za jeho odbornú pomoc, čas a pripomienky.

Obsah

Kapitola 1

Úvod

V roku 2016 trpelo podľa štatistík Svetovej zdravotníckej organizácie (World Health Organisation, ďalej len „WHO“) 1.9 miliardy osôb starších ako 18 rokov nadváhou, pričom až 650 miliónov z nich obezitou alebo chorobou spojenou s obezitou - ako napríklad diabetes melitus, či kardiovaskulárne ochorenia. Tieto choroby boli v roku 2012 najčastejšou príčinou úmrtia[30]. Hlavným dôvodom nadváhy a obezity je podľa výskumov a údajov WHO energetická nevýváženosť prijatých a spotrebovaných kalórii jedinca. Daná inštitúcia zároveň tvrdí, že nadváhe, obezite a chorobám spôsobených obezitou sa dá predísť.

V minulosti človek vykonával fyzicky omnoho náročnejšie aktivity v početnejšom množstve ako je tomu teraz, čo zabezpečovalo väčší denný výdaj energie. V dnešnej dobe je dostupnosť jedla v krajinách, ktoré disponujú vysokým percentom ľudí v nadváhe, fyzicky oveľa menej náročnejšia a dostupňejšia. V konečnom dôsledku to znamená, že sa príjem energie zvýšil a jej výdaj sa znížil.

Vhľadom k vyššie uvedeným faktom som sa zamyslel, akým spôsobom by bolo možné v dnešnej rýchlej dobe plnej informačných technológii využiť možnosti modernej techniky k tomu, aby napomohla ľuďom, ktorí by radi začali športovať, no z dôvodu nechute športovať osamote alebo aj iných obdobných dôvodov sa vyhýbajú myšlienke návštevy športoviska, či vykonávania športovej aktivity získať chuť a odvahu k cvičeniu. Mobilná aplíkácia by nielen uľahčila, ale aj spríjemnila vykonávanie športových aktivít prostredníctvom možnosti zaistenia a vytvárania spoločných skupinových udalostí so zameraním na vykonávanie fyzickej činnosti prostredníctvom športu.

V tejto práci je mojim cieľom vytvoriť mobilnú aplikáciu na platformu Android, ktorej hlavnou úlohou bude vyhľadávanie osôb so záujmom o vykonávanie spoločných športových aktivít, umožnenie následnej komunikácie medzi potencionálnymi športovcami, odporúčanie dostupných športovísk v blízkom okolí, vytvorenie športovej udalosti, ktorej sa môže zúčastniť viacero osôb alebo aj napríklad hodnotenie vykonávaných aktivít či osôb ako takých.

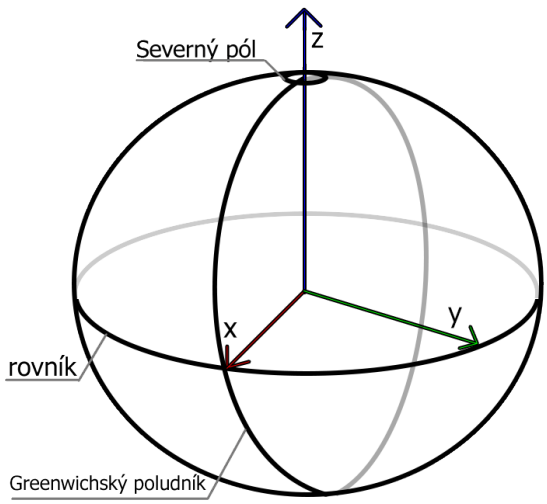
Kapitola 2

Súradnicové systémy

V dnešnej dobe je potrebné lokalizovať určitý bod v ľubovoľnom priestore. V našom prípade je nutné sa zamerať na lokalizovanie bodu na planéte Zem, ktorá sa vyznačuje jej geoidným tvarom. Odvetvie geodézie dnes definuje množstvo základných formiem určenia pozície bodu na planéte Zem.

2.1 Kartézsky súradnicový systém

Kartézsky súradnicový systém patrí medzi najstaršie súradnícové systémy, ktoré sa frekventovane využívajú dodnes. Tento systém bol definovaný v 17. storočí francúzkym matematikom Reném Descartesom.[8] Hlavnou myšlienkou je určenie počiatočného bodu a osí, ktoré sa zväčša označujú písmenami abecedy. V prípade, že je určený počiatočný bod a všetky osi, ktorých počet závisí od počtu dimenzií, je možné bod unikátne reprezentovať na základe pozície od počiatočného bodu.



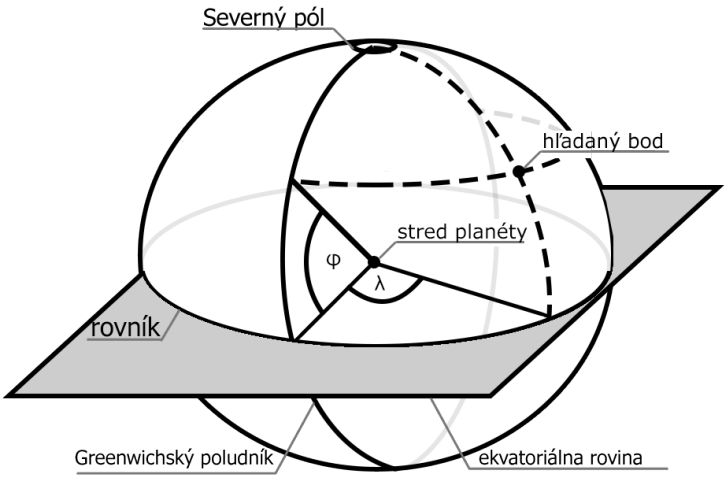
Obr. 2.1: Kartézsky súradnicový systém aplikovaný na planétu Zem.
zdroj [5], spracovanie: vlastné

Tento súradnicový systém môže byť využívaný v priestoroch, ktoré využívajú rôzne počty rozmerov. Kartézsky súradnicový systém dokáže unikátne definovať bod v n-rozmernom priestore [8]. Ak tento systém aplikujeme na planétu Zem, potom jeho nulovou súradnicou bude miesto ťažiska planéty. Následne bude platiť, že os x smeruje smerom na Greenwichský poludník a spolu s osou y definujú rovinu, ktorá prechádza rovníkom. Os z smeruje z počiatočného bodu súradnicového systému, prechádza severným pólom planéty a je na rovinu, určenou osami x a y, kolmá. Samotný systém je vizualizovaný na obrázku 2.1.

2.2 Geografický súradnicový systém

Ako už samotný názov napovedá, jedná sa o súradnicový systém využívaný v odvetví geografie. Systém pozostáva z troch parametrov, pomocou ktorých je možné definovať ľubovoľný bod na povrchu planéty Zem.

Prvým z parametrov je latitúda ϕ. Tá určuje veľkosť uhla, ktorý hľadaný bod zviera s ekvatoriálnou rovinou. Môže nadobúdať hodnoty od $+90°$ až po $-90°$, pričom hodnotu $0°$ nadobúda priamo na ekvatoriálnej rovine. Hodnota latitúdy smerom na sever stúpa a smerom na juh naopak klesá. Longitúda λ je druhým parametrom tohto súradnicového systému. Určuje uhol, ktorý bod zviera od stredu planéty s Greenwichským poludníkom. Longitúda môže nadobúdať hodnoty od $+180°$ do $-180°$ a platí, že stúpa smerom na východ a klesá na západ. Posledným parametrom je výška h, ktorá definuje vzdialenosť bodu od povrchu referenčného elipsoidu, ktorý predstavuje geometrický objekt podobný planéte Zem.[25] Geografický súradnicový systém je vyobrazený na obrázku 2.2

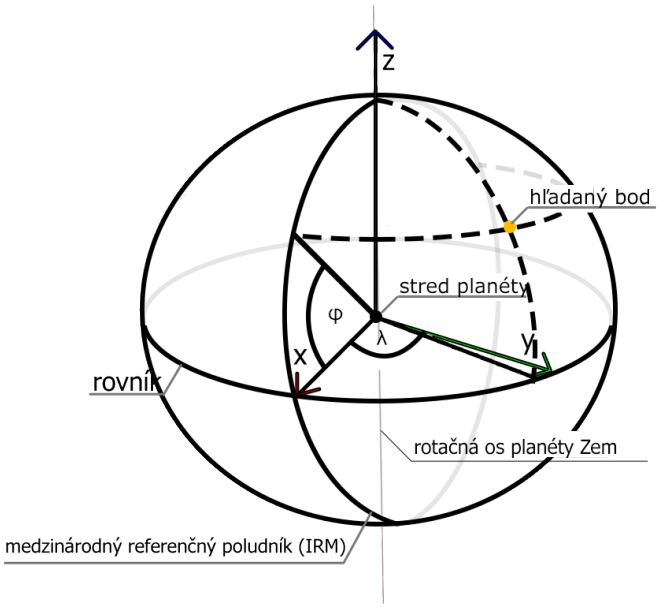


Obr. 2.2: Geografický súradnicový systém.
zdroj [24], spracovanie: vlastné

2.3 Geodetický systém WGS84

Svetový geodetický systém z roku 1984 (ďalej len „WGS84“) definuje štandard pre súradnicový systém planéty Zem, ale aj parametre referenčného telesa, ktoré svojim tvarom blízko pripomína tvar planéty Zem[24]. WGS84 bol v roku 1984 definovaný ministerstvom obrany Spojených štátov amerických[7] a je využívaný aj v dnešnej dobe, čoho dôkazom je aj to, že ho využívajú aj známe verejné služby akými sú Google Maps, Google Earth alebo aj GPS zariadenia.

Počiatočným bodom súradného systému je rovnako ako aj už pri skôr spomenutých systémoch v ťažisku planéty. Systém tiež využíva longitúdu λ a latitúdu ϕ, avšak longitúda nadobúda hodnotu 0° na medzinárodnom referenčnom poludníku (ďalej len „IRM“), ktorý je vzdialený približne 102 metrov smerom na východ od Greenwichského poludníka. Os z je súhlasná s rotačnou osou Zeme, os x smeruje na IRM a y je na dané osy kolmá tak, že dotvára pravotočivý súradný systém s longitúdou a latitúdou, ktoré nadobudajú rovnakú škálu hodnôt ako je to spomenuté u geografického súradnicového systému 2.2. [24]



Obr. 2.3: Súradnicový systém geodetického systému WGS84.
zdroj [24], spracovanie: vlastné

Kapitola 3

Problematika vyhľadávania POI

Táto kapitola bližšie špecifikuje pojem bod záujmu a následne sa bude venovať problematike spôsobu vyhľadávania týchto bodov pomocou rôzných prístupov.

3.1 POI

POI, celým názvom point of interest, v preklade bod záujmu (ďalej len „BZ“), označuje lokalizáciu miesta, vyselektovaného na základe poznatkov a zadaných parametrov od užívateľa. Toto miesto môže užívateľ príslušnej aplikácie, pracujúcej s vyhľadávaním BZ, považovať za dôležité alebo inak zaujímavé. Môže sa jednať o rôzne typy miest napríklad reštaurácie, autopredajne, športové centrá atď. [27]

Tento termín je najčastejšie používaný v oblasti kartografie, špecificky v jej digitálnych variantách ako napríklad v systémoch GIS (Geografický informačný systém) a navigačných systémoch GPS (Globálny systém určenia polohy). BZ špecifikuje minimálne hodnotu latitúdy a longitúdy, avšak môže byť rozšírený aj o iné potrebné informácie.[31]

3.2 Vyhľadávanie vhodných POI

Každý užívateľ je unikátny, preto je nutné aby zobrazenie BZ pre danú osobu zodpovedalo jeho záujmom a správaniu. Ak má človek obľubu v automobiloch, tak by nebolo vhodné, aby systém vyobrazil pre neho ako BZ botanickú záhradu [32]. Za týmto účelom boli vyvinuté rôzne techniky filtrovania, ktoré pomocou vhodných algoritmov vyberajú vhodné BZ pre daného užívateľa.

3.2.1 Kolaboratívne filtrovanie

Kolaboratívne filtrovanie (angl. collaborative filtering), ďalej len „CF“, je často využívaná technika, ktorú dnes využíva množstvo systémov, ktoré BZ odporúčajú. Táto technika definuje až niekoľko metód, ktoré filtrovanie zabezpečujú. Metódy CF sú rozdelené do dvoch kategórií, ktoré sa nazývajú kolaboratívne filtrovanie využívajúce model (angl. model-based CF) a kolaboratívne filtrovanie pracujúce s pamäťou (angl. memory-based CF). Kolaboratívne filtrovanie pracujúce s pamäťou sa ďalej rozdeľuje na kolaboratívne filtrovanie založené na podobnosti používateľov (angl. user-based CF) a na kolaboratívne filtrovanie založené na podobnosti objektov (angl. item-based CF) [31].

Kolaboratívne pracujúce s pamäťou

Kolaboratívne filtrovanie založené na podobnosti používateľov funguje tak, že na začiatku vyhľadá podobnosť užívateľov na základe ich hodnotenia rôznych BZ využívaním meraní podobnosti, napríklad pomocou kosínovej podobnosti[31]. Následne pre každý nový BZ vypočíta skóre, ktoré je vyčíslené pomocou váženého priemeru všetkých historických hodnotení užívateľov, ktorí sú podobní danému užívateľovi.

Výpočet podobnosti dvoch užívateľov je znázornený v rovnici 3.1 a samotný vypočet skóre pre daný BZ v rovnici 3.2.

$$w_{i,k} = \frac{\sum_{l_j \in L} c_{i,j} c_{k,j}}{\sqrt{\sum_{l_j \in L} c_{i,j}^2} \sqrt{\sum_{l_j \in L} c_{k,j}^2}} \tag{3.1}$$

$$\hat{c}_{i,j} = \frac{\sum_{u_k} w_{i,k} \cdot c_{k,j}}{\sum_{u_k} w_{i,k}} \tag{3.2}$$

V rovniciach 3.2 a 3.1 majú jednotlivé označenia tento význam. Označenie u_i a u_k označuje užívateľov z množiny užívateľov U. Premenná l_j určuje bod záujmu z množiny L. Prvok $w_{i,k}$ je konečná podobnosť užívateľov u_i a u_k. Parameter $c_{k,j}$ určuje, či užívateľ u_k už navštívil v minuosti bod l_j. V prípade, že u_k navštívil bod l_j, hodnota $c_{k,j}$ nadobúda hodnotu 1, inak 0. Premenná $c_{i,j}$ predstavuje to isté ako $c_{k,j}$, ale je mierená na užívateľa u_i. Nakoniec $\hat{c}_{i,j}$ je konečné skóre pre bod l_j prislúchajúci užívateľovi u_i. [31]

Kolaboratívne filtrovanie založené na podobnosti objektov zas funguje na inom princípe, to takom, že vyhľadáva BZ, ktoré sú podobné bodom záujmu, o ktoré už daný užívateľ mal v minulosti záujem, navštívil ich alebo ohodnotil.

Kolaboratívne filtrovanie využívajúce model

Kolaboratívne filtrovanie využívajúce model používa zoskupovanie rôznych užívateľov v tréningovej databáze do malých skupín na základe ich ohodonocovacích vzorov[31]. Základom tohoto spôsobu odporúčenia je strojové učenie a využívanie rôznych algoritmov, ktoré ho podporujú. Problémom je však vysoká výpočetná náročnosť a taktiež to, že tento typ systémov nedokáže pokryť tak vysokú škálu rôznych typov užívateľov ako dokáže kolaboratívne filtrovanie pracujúce s pamäťou.[31]

3.2.2 Časovo závislé vyhľadávanie

Časovo závislé vyhľadávanie vhodných BZ rozširuje spôsob odporúčania BZ v závislosti na aktuálnom čase, pretože človek ako taký, vykonáva určitý čas špecifické činnosti. Napríklad je oveľa menej pravdepodobné, že človek pôjde doobeda navštíviť nejaký nočný klub, ako to, že v tomto čase bude chcieť navštíviť plaváreň. Preto tento typ vyhľadávania definuje dva rôzne typy správania človeka a to časové správanie a priestorové správanie[32], na základe ktorých dokáže vybrať vhodné BZ.

Časové správanie

Čas je nutné považovať za dôležitý aspekt pri skúmaní denných aktivít užívateľa. Toto správanie je možné odpozorovať na základe nahlasovania užívateľovej polohy v priebehu časových úsekov. Systém na základe histórie sledovania polohy užívateľa dokáže analyzovať a následne aj využiť všetky zozbierané údaje potrebné pre čo najlepšie odporúčanie BZ

pre užívateľa v danom čase. Pre ilustráciu, ak veľké množstvo užívateľov navštevuje v čase obeda reštauráciu, a len malá časť užívateľov navštevuje v rovnakom čase bar, tak systém uprednostní ako vhodný BZ pre užívateľa reštauráciu pred barom. V prípade, že dvaja užívatelia disponujú podobným časovým správaním, je veľmi pravdepodobné, že v rovnakom čase budú mať totožný BZ[32].

Priestorové správanie

Po časovom správaní je ďalším dôležitým faktorom pri skúmaní denných aktivít priestorové správanie, ktoré sa zameriava na okolie v ktorom sa užívateľ nachádza. Faktom je, že užívatelia majú tendenciu si vybrať práve kontrétny BZ na základe vzdialenosti od ich aktuálnej polohy. Vďaka tomuto poznatku priestorové vyhľadávanie počíta s tým, že je väčšia pravdepodobnosť, že osoba navštívi BZ v jeho blízkosti ako BZ, ktorý je od neho viac vzdialený.[32]

3.3 Meranie vzdialenosti vzájomných bodov

Mobilná aplikácia, ktorá je predmetom tejto práce, bude vyhľadávať body na rôznych svetových pozíciách. Faktom je, že človek môže byť vždy v jednom časovom okamihu práve na jednom mieste, preto je ideálnym riešením zobrazovať iba dôležité BZ najmä v jeho blízkosti. Z toho dôvodu je nutné definovať spôsob, akým sa táto vzdialenosť bude merať.

Keďže mobilné zariadenia a GPS zariadenia pracujú hlavne s geodetickým systémom WGS84 2.3, je potrebné aby výpočet zároveň v sebe zahŕňal vlastnosti tohto systému.

Samotný vzorec na výpočet vzájomnej vzdialenosti bodov predstavuje rovnica 3.3, kde D je výsledná vzdialenosť bodov, d predstavuje polomer planéty Zem, ktorý je podľa geodetického systému WGS 6378km, ϕ_A a ϕ_B sú hodnoty latitúd v bodoch A a B, premenné λ_A, λ_B predstavujú hodnoty longitúd v daných bodoch.[29] V prípade, že je potrebné merať vzdialenosť bodov v imperiálnych jednotkách, tak je nutné dosadiť namiesto parametra d polomer planéty v míľach.

$$D = d \cdot \arccos(\cos(\phi_A)) \cdot \cos(\phi_B) \cdot \cos(\lambda_B - \lambda_A) + \sin(\phi_A) \cdot \sin(\phi_B)) \tag{3.3}$$

Kapitola 4

Vývojové prostredie a nástroje Android

Android je operačný systém, ktorého základom je jadro na báze Linuxu. Jedná sa o open source projekt vyvinutý firmou Google. Dnes je najrozšírenejším operačným systémom pre mobilné zariadenia rôzneho typu, ako napríklad telefóny či tablety. Tento operačný systém je dostupný aj pre neprenosné zariadenia ako televízie. So zvyšovaním popularity operačného systému stúpa aj dopyt zákazníkov po nových možnostiach využívania zariadení, ktoré na ktorých Android beží. Aby si operačný systém udržal svoju vedúcu pozíciu na svetovom trhu, je potrebné aby vývoj napredoval a ponúkal čo najväčšiu škálu možností. Preto dnes firma Google poskytuje mnoho prostriedkov, ktoré slúžia na zjednodušovanie vytvaránia aplikácii pre operačný systém Android.

4.1 Android Studio

Android Studio bolo v predstavené v roku 2013 ako oficiálne vývojové prostredie (ďalej len „IDE“) pre operačný systém Android. Tento produkt je založený na základe populárneho IDE InteliJ IDEA vytvoreného firmou JetBrains. Aj napriek tomu, že je InteliJ IDEA komerčným produktom, tak Android Studio je k dispozícii pre vývojárov zdarma.

Android studio je priamo prispôsobené na vývoj aplikácii pre Android, preto obsahuje množstvo podporných modulov na spravovanie jednotlivých verzií nástrojov pre vývoj softvéru, ktoré sú bližšie popísané v kapitole 4.2. Umožňuje nastavenie ich automatických aktualizácii, či inštalovania a odinštalovania jednotlivých modulov. Taktiež umožňuje prístup k dokumentácii a rôznym štatistikám, ktoré sú pre vývojara dôležité. Android Studio nezanevrelo ani na grafické rozhranie pre ladenie tvoreného programu, grafický editor pre vytváranie grafického užívateľského rozhrania aplikácie a virtuálny stroj na testovanie a monitorovanie aplikácie pre ľubovoľné verzie operačného systému Android.

Vďaka prehľadnému rozhraniu a ponuke vyššie uvedených funkcii sa Android Studio javí ako ideálne IDE pre túto prácu.

4.2 Android SDK

Android SDK je súbor nástrojov na vývoj softvéru pre platformu Android. Ponúka širokú škálu prostriedkov, ktoré uľahčujú vývoj a tvorenie produktov pre operačný systém Android. SDK podporuje tvorbu v programovacích jazykoch Java alebo Kotlin a je dostupné

pre operačné systémy Windows, Linux, Mac OS X. Samotné SDK je rozdelené na viacero modulov, ktoré sú špecifické svojim využitím a funkcionalitou [26].

1. Platform-tools
2. SDK-tools
3. Build-tools
4. Android Emulator

Platform-tools

Android SDK Platform-Tools sú jedným z modulov Android SDK. Tento modul ponúka nástroje adb (Android debug bridge), systrace a fastboot, ktoré umožňujú prepojenie s platformou operačného systému Android, s ktorou aktuálne vývojár pracuje a sú potrebné pre vývoj aplikácii pre Android. V prípade aktualizácie verzie tohto modulu stačí, aby bol nainštalovaný najnovší, pretože vždy sú verzie spätne kompatibilné.[18]

Systrace je veľmi užitočný nástroj z tohto komponentu. Ako už názov napovedá, `systrace` slúži na sledovanie stavu platformy počas behu programu. Dokáže vyobraziť zaťaženie jednotlivých jadier procesorov, využitie pamäťového priestoru aplikácii, množstvo spotrebovaných internetových dát a škálu iných potrebných informácii vhodných najmä pre optimalizáciu aplikácie.[20]

Ďalším veľmi užitočným nástrojom je adb, ktorý umožňuje komunikáciu s napojeným zariadením. Celá komunikácia funguje na báze klient-server aplikácii. Zariadenie, na ktorom prebieha vývoj posiela inštrukcie napojenému zariadeniu, aké príkazy má vykonať. Dokáže napríklad automaticky nainštalovať a spustiť aplikáciu do zariadenia a sprístupniť shell, čo je tradičné rozhranie pre UNIX-ové systémy.[10]

SDK-tools

Modul SDK-tools ponúka široké spektrum funkcionality. V minulosti boli spojené s Build-tools a obashovali aj nástoje na zostavenie programu, či jeho ladenie. Situácia sa však neskôr zmenila a tieto dva moduly boli od seba oddelené.

Dnes komponent SDK-tools sprístupňuje kompletný set knižníc a funkcií potrebných pre samotnú implementáciu aplikácií na zariadenia využívajúce operačný systém Android.[19]

Build-tools

Android SDK Build-tools, v preklade nástroje pre zostavenie programu, sú časťou SDK a sú zodpovedné za konečné zostavenie aplikácie. Aplikačný súbor, ktorý je pomocou týchto nástrojov zostavený, sa vyznačuje príponou „.apk“ v jeho názve. Od samotného počiatku bol tento modul rozšírený o niekoľko nových dôležitých súčastí.

Jednou zo súčastí je nástroj zipalign, ktorý ponúka dôležitú pamäťovú optimalizáciu. Základom je to, že tento nástroj zarovná nekomprimované dáta vo výslednom súbore tak, aby bola ich relatívna vzdialenosť od počiatku súboru rovnaká, čo má za následok nižšiu pamäťovú náročnosť aplikácie v čase jej behu.[23]

Android Emulator

Android Emulator je komponentom, ktorý umožňuje spustenie aplikácie vytvorenej pre operačný systém Android na platforme, na ktorej prebieha vývoj aplikácie. Je to veľmi

vhodný prostriedok, pretože umožňuje simuláciu behu aplikácie na rôznych typoch zariadení, pre ktoré je produkt určený, bez toho, aby bolo zariadenie zakúpené, čo vývojárovi ušetrí náklady na nákup zariadenia, ale aj uľahčí testovanie aplikácie na rôznych verziách operačného systému Android. Taktiež ponúka grafické rozhranie pre všetky ovládacie prvky, ktoré zariadenie v aktuálnej verzii podporuje. [14]

4.3 Google Maps API

Google Maps API predstavuje aplikačné rozhranie k službe Google Maps, ktorá reprezentuje široké spektrum rôznych funkcií umožňujúcich integráciu a vyobrazovanie mapy v aplikácii.[1] Aplikačné rozhranie tiež umožňuje získavanie informácií z rozsiahlej databázy dát, ktorou firma Google disponuje, a sú v nej uchované potrebné a najmä aktuálne informácie o rôznych miestach, adresách, povolených rýchlostiach na rôznych úsekoch ciest, prevod súradníc z geodetickeho systému WGS84 2.3 na adresy a ďalšie iné funkcie.

Google Maps API taktiež poskytuje výpočet vzdialenosti, vyobrazenie najkratšej cesty a čas potrebný na presun z bodu A do bodu B pomocou rôznych typov dopravných prostriedkov, napríklad chôdzou, bicyklom, automobilom, autobusom, vlakom či lietadlom.

Aplikačné rozhranie nezabudá ani na prispôsobenie štýlu zobrazenej mapy[1]. Pre každú vrstvu mapy (cesty, cyklochodníky, vodu, pevninu ...) je možné nielen nastaviť ľubovoľné sfarbenie textu, ale aj samotného útvaru, ktorý predstavuje daný objekt na mape. Taktiež je možné jednotlivé vrstvy skryť.

Veľkou výhodou Google Maps API je jeho multiplatformové využitie. Môže byť využité pri webových aplikáciách, produktoch pre mobilné zariadenia s operačných systémom iOS od firmy Apple Inc. a samozrejme pre operačný systém Android. Pre platformu Android je sprístupnenie využívania služieb Google Maps pomocou aplikačného rozhrania neobmedzené a zdarma.

Kapitola 5

Analýza a špecifikácia požiadavok

Táto kapitola sa bude venovať analýze požiadavok pre aplikáciu, ktorá je predmetom tejto práce. Na začiatku kapitoly bude analyzovaných niekoľko prípadov aplikácii s podobným zámerom, následne sa text bude zaoberať cieľovou skupinou, pre ktorú je aplikácia zameraná a nakoniec sa sa zameria na technické a funkčné požiadavky aplikácie.

5.1 Existujúce riešenia

Aj keď už existuje akási predstava o podobe, funkcionalite a cieľovej skupine aplikácii, je vhodné si rozobrať niekoľko existujúcich alebo podobných produktov, ktoré majú podobné zameranie alebo využívajú obdobné techniky. Analýzou existujúcich riešení je možné získať nové poznatky o problematike a tiež pohľady na riešenia problémov, ktoré rozšíria rozhľad a pomôžu zefektívniť konečný produkt.

5.1.1 Meetup

Meetup je mobilná aplikácia slúžiaca ako sociálna sieť [1] , vhodná na vyhľadávanie a vytváranie akýchkoľvek udalostí v reálnom svete, ktoré organizujúca osoba v aplikácii zaevidovala. Meetup umožňuje vyhľadávanie podujatí v okolí užívateľa a následnu interakciu. Umožňuje potvrdenie účasti, pridávanie fotografií, tiež ponúka možnosť viesť diskusiu a hodnotiť podujatia po ich ukončení.

Za kladné možno považovať možnosť socializácie a vytváranie širších okruhov osôb s rovnakým záujmom ako má daný užívateľ. Negatívnymi prvkami aplikácie sú nemožnosti písania súkromných správ určitým skupinám osôb alebo neprehľadný kalendár podujatí, ktorý neoznámi zmeny ohľadom podujatia zúčastnením užívateľom hneď, ale až po určitom čase, čo spôsobuje nespokojnosť a chaos medzi samotnými zúčastnenými.

Na účely tejto bakalárskej práce je možné sa inšpirovať jej schopnosťou vytvárania podujatí a následnými operáciami s nimi alebo taktiež aj hodnotiacou časťou aplikácie. Pre porovnanie, aplikácia bakalárskej práce bude na rozdiel od aplikácie Meetup poskytovať možnosť vzájomnej komunikácie medzi účastníkmi podujatia či lepšou rýchlosťou notifikovania o prípadných zmenách týkajúcich sa udalosti.

[1] Dostupné na: https://play.google.com/store/apps/details?id=com.meetup

5.1.2 Eventbride

Eventbride [2] – Aplikácia podobná vyššie spomenutej. Jej hlavnou úlohou je vyhľadávanie podujatí v používateľovom blízkom okolí na základe jeho osobných preferencií a ním vopred zadaných požiadaviek na následné správne odporúčanie vybraných podujatí. Pozitívom je personalizácia aplikácie pre lepšie vyhľadávanie pre osobu zaujímavých udalostí, zobrazovanie článkov týkajúcich sa daných podujatí či jej mnohojazyčnosť. Negatívom je absencia možnosti chatu alebo inej interakcie medzi samotnými užívateľmi. Pre potreby tejto práce je zaujímavá jej práca s polohou používateľa či následné vyhľadávanie bodov záujmu v spolupráci s aktuálnou polohou, no už spomínaná absencia možnosti interakcie osôb činí aplikáciu tejto bakalárskej práce minimálne v tomto aspekte o niečo lepšou.

5.1.3 Party With

Party With [3] je aplikácia dostupná pre operačné systémy Android a iOS. Aj keď o tom samotný názov a cieľ aplikácie na prvý pohľad nevypovedá, tak sa jedná o produkt, ktorý je najpodobnejší tomu, čo má byť výsledkom tejto práce.

Party With slúži na spájanie ľudí, ktorí majú záujem o nočnú párty. Aplikácia umožňuje vytvoriť udalosť (v tomto prípade párty) u ktorej sú špecifikované konkrétne informácie ako miesto, čas a typ párty, ale tiež obsahuje nástenku, kde dáva možnosť užívateľom pridávať rôzne návrhy, otázky alebo správy. Umožňuje tiež potvrdenie účasti iných užívateľov na danú akciu.

Dôležitou funkciou je aj to, že vyhľadáva párty alebo ľudí, čo majú v danej chvíli rovnaké záujmy. Rozdeľuje užívateľov na viac skupín. Na tých, čo majú záujem o veľkú párty a tých, ktorí uprednostňujú menšie spoločenské akcie. Zároveň umožňuje priame nakontaktovanie osoby pomocou chatu, čo je veľkou výhodou oproti už spomenutým aplikáciam.

Okrem toho, že sa aplikácia nezameriava na šport ale na párty, tak za nevýhodu je možné považovať chýbajúcu podporu zobrazovania udalostí a osôb na mape. Udalosti a iných užívateľov dokáže zobrazovať iba v zoznamoch, čo môže pôsobiť neprehľadne.

5.2 Cieľová skupina

Pri vývoji aplikácií je vždy nutné sa zamyslieť nad otázkou, koho by mala osloviť. Športová aktivita je špecifická činnosť, pri ktorej je nutné sa zamerať na rôzne parametre, ktoré profiluju osoby so záujmom o vykonávanie športovej činnosti prostredníctvom mobilnej aplikácie.

Spoločné záujmy užívateľov

Ako je všeobecne známe, človek je bytosťou spoločenskou, ktorá vo väčšine prípadov uprednostňuje skupinovú prácu pred individuálnou, čo možno vidieť najmä v odvetví športu. Pravdaže, v každom spoločenstve sa nájdu výnimky. Preto sociálna skupina, na ktorú sa bude produkt zameriavať, je charakteristická svojim záujmom o športovanie. Jedná sa o ľudí, ktorí majú záujem športovať, no vyhľadávajú k tomu niekoho, kto by sa k nim pridal, či už z toho dôvodu, že sa cvičiť hanbia, alebo ich proste nebaví cvičiť o samote.

[2] Dostupné na: https://play.google.com/store/apps/details?id=com.eventbrite.attendee
[3] Dostupné na: https://play.google.com/store/apps/details?id=com.pwal

Vekové rozmedzie užívateľov

Veková škála užívateľov nie je v striktnom rozmedzí, no predpokladá sa, že mnoho starších ľudí, najmä v dôchodkovom veku, nie je s dnešnými technickými prostriedkami stotožneních. Avšak aj v tejto skupine ľudí existujú výnimky, čo neznamená, že to bude hlavná cieľová skupina. Ale keďže cieľom je vytvoriť čo najlepšiu aplikáciu, tak bude nutné sa pri vývoji produktu obzvlášť zamerať aj na prehľadné a intuitívne grafické uživateľské rozhranie, ktoré umožní ľuďom jednoduchšie využívanie služieb.

Taktiež je nutné podotknúť, že by táto aplikácia mala byť prístupná osobám až od tínedžerského veku, samozrejme s dozorom plnoletých do dovršenia veku plnoletosti. Dôvodom tohto obmedzenia je fakt, že aplikácia dokáže prepojiť osobu s cudzou osobou, čo môže byť v niektorých prípadoch nebezpečné.

5.3 Technické špecifikácie

Pri vytvárani aplikácii je nutné špecifikovať aj technické požiadavky na aplikáciu, pretože je neekonomické a neefektívne implementovať aplikáciu na platformu, ktorú dnes už nikto nepoužíva, alebo naopak na platformu, ktorá je inovatívna no má k nej prístup len malé množstvo ľudí.

5.3.1 Operačný systém

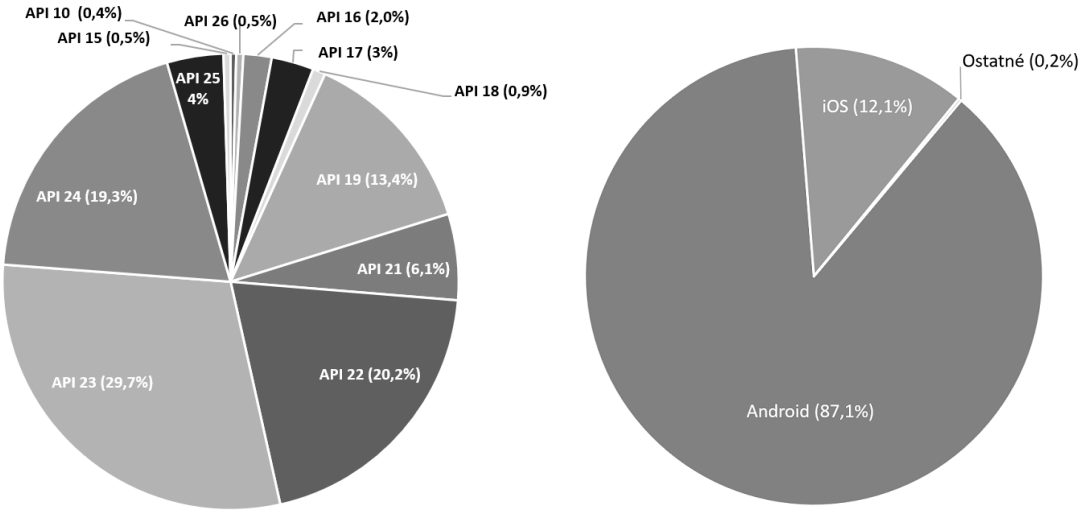
Podľa portálu statista.com v roku 2017 vedú štatistky podielu zákazníkov kupujúcich mobilné zariadenia dva operačné systémy, ktoré pokrývajú dohromady 99.8% kupujúcich. Je to operačný systém Android od firmy Google, ktorý ma podiel 87,1%, a operačný system iOS od firmy Apple Inc., ktorý disponuje podielom zákazníkov v pomere 12,1%. Zvyšnú časť si medzi sebou delia ostatné operačné systémy. Predchádzajúce štatistiky sú dôvodom toho, že konečný produkt tejto práce bude implementovaný pre mobilné telefóny využívajúce operačný systém Android.

Verzia systému a API

S každou novou verziou systému vydáva firma Google novú verziu aplikačného rozhrania (ďalej len„API“). Pre každé API platí, že je spätne kompatibilné so staršiou verziou. To znamená, že zariadenie ktoré má v sebe nainštalované operačný systém s API verziou 27, dokáže spustiť aplikácie využívajúce API 27 a nižšie. Opačne to však neplatí. Preto sa naskytá problém, akú verziu API je potrebné využiť pri implementácii. Hlavné jadro problému nastáva v tom, že príliš novou verziou API sa pokryje len malé percento užívateľov operačného systému, a príliš starou verziou sa síce pokryje celá množina zákazníkov, no aplikácia nebude môcť využívať niektoré nové funkcie, čo môže mať za následok isté problémy pri implementácii, ktoré samotný vývoj aplikácie značne zneefektívnia a predĺžia.

Aktuálna verzia API je 27, ktorá pokrýva menej ako pol percenta zariadení. Veľmi viditeľný percentualny rozdiel pokrytia užívateľov a verzie API je medzi verziami 19 a 20.

Verzia 20 je veľmi inovatívna a ponúka mnohé rozšírenia ako rozšírené štýovanie zobrazenia notifikácii, čo je v prípade aplikácie, ktorá sa bude tvoriť, veľmi užitočným prvkom. Ďalej táto verzia ponúka nový typ grafického zobrazenia material design, ktorý pôsobi reálnejšie a krajšie oproti starším verziám[11]. Prináša aj veľkú zmenu v práci so zoznamami, kde ponúka grafický widget RecyclerView.



Obr. 5.1: Podiel aktívne využívaných API (vľavo)[13] a podiel operačných systémov využívaných v mobilných zariadeniach v roku 2017 (vpravo)[28].

Ale aj napriek týmto skvelým funkciám pokrýva iba 79,1% aktívnych zariadení, čo je oproti verzii 19 až o 13.4% menej[13]. Nie je možné počítať ani s tým, že sa v blízkej budúcnosti toto percento zníži, pretože analýzou dostupných telefónov, bol nadobudnutý poznatok, že mobilné zariadenia, ktoré v sebe majú zabudovanú verziu operačného systému Android s API verziou 19, sú v obchodoch stále dostupné. Z tohto dôvodu sa bude pri implementácii využívať verzia API 19.
Na obrázku 5.1 vľavo nie sú znázornené verzie API, ktoré majú podiel využitia nižší ako 0.5%.

5.3.2 Internetové pripojenie

Už z úvodného popisu je jasné, že aplikácia bude musieť využívať internetové pripojenie, a to už minimálne z toho dôvodu, že bude vyhľadávať užívateľov, ktorí sú tiež pripojení na rovnakej aplikácii.

Pri mobilných zariadeniach sa ako hlavná prekážka vyníma obmedzovanie prijatých a odoslaných dát. Kvôli tomu, že internetové pripojenie na mobilných zariadeniach je mobilnými operátormi spoplatňované a spotrebovaním dát úmerne rastie aj celková cena za pripojenie, je potrebné aby aplikácia spotrebu dát znížila vždy iba na potrebné minimum. Preto je nutné vybrať vhodný aplikačný protokol.

Existuje množstvo protokolov, ktoré však obsahujú špecifické dáta pre funkcionalitu, kvôli ktorej boli vytvorené. Keďže cieľom je, aby aplikácia využívala čo najmenší objem dát z internetu, tak sú tieto protokoly nevhodné, pretože by obsahovali zbytočné redundantné data. Preto sa javí ako ideálne riešenie návrh vlastného aplikačného protokolu, ktorý bude posielať iba dáta, ktoré aplikácia potrebuje. Tým sa vývoj aplikácie síce predlží, no v konečnom dôsledku to umožní redukovať veľkosť odoslaných a prijatých dát na minimum, ktoré táto aplikácia pre svoju funkcionalitu vyžaduje.

5.4 Funkčné požiadavky

Tvorba mobilnej aplikácie v sebe nutne zahŕňa aj proces skúmania vhodných funkčných požiadavok. Analýzou existujúcich aplikácií 5.1 sa vymedzili ich kladné a záporné funkcionality, ktoré budú na účely tejto aplikácie či už použité alebo inak vhodne doplnené o nové funkcie, ktoré v testovaných produktoch chýbali, alebo boli použité nevhodne.

5.4.1 Vytváranie skupinových udalostí

Na základe analýzy iných obdobných aplikácii som dospel k záveru, že používatelia vychádzajúc z ich ľudskej podstaty, potreby vzájomnej sociálnej interakcie, majú v obľube vykonávať rôzne aktivity spoločne s inými osobami. Túto možnosť im však v kombinácii s inými funkčnými požiadavkami na podobné aplikácie na trhu momentálne neponúkajú. Z toho dôvodu som sa rozhodol do tejto baklárskej práce zakomponovať aj vytváranie skupinových športových udalostí, ktoré budú umožňovať vytváranie skupinových športových udalostí s vopred určeným časom, miestom vykonávania aktivity alebo aj prípadnou interakciou prihlásených účastníkov pomohla vyplniť túto dieru na trhu mobilných aplikácii zaoberajúcich sa podobnou problematikou. Užívateľ bude mať možnosť prihlásenia sa k danej vytvorenej udalosti, pozvať iných užívateľov k tejto akcii alebo aj návhrom možnej zmeny týkajúcej sa času a miesta konania športovej udalsoti.

5.4.2 Zobrazovanie na mape

Zobrazovanie miesta konania vopred dohodnutých aktivít na mape bolo v rámci testovania vyššie uvedených aplikácií v každom jednom spomenutom prípade považované užívateľmi za či už v prípade využitia tejto funkcie za pozitívnu funkciu, či v prípade jej absencie to používatelia označili ako negatívum celého konečného produktu. Preto bude aplikácia tejto bakalárskej práce disponovať funkciou zobrazovania vopred určených miest na mape, ktorá bude implementovaná pomocou Google Maps API 4.3. Bude sa jedať o zobrazovanie pozícií ľudí v reálnom čase, pri ktorých po rozkliknutí ich pozície zobrazí bližsie informácie a umožní osobu nakontaktovať. Zobrazované budú taktiež aj iné miesta ako fitness centrá, parky, posilňovne, športové haly, bežecké okruhy alebo iné zariadenia zaoberajúce sa športom.

5.4.3 Chatovacia služba

Chatovacia služba patrí medzi základné funkcie každej jednej úspešnej aplikácie vo sfére sociálnych sietí na trhu. Je to spôsbené tým, že si ľudia v dnešnej dobe zvykli na možnosť rýchleho kontaktu s inou osobou prostredníctvom rôznych zariadení pripojených na internet. Aj z tohto dôvodou bude konečná aplikácia zahŕňať chatovaciu službu, ktorá bude slúžiť za účelom individuálnej komunikácie medzi všetkými užívateľmi používajúcimi túto aplikáciu, kolektívnej komunikácie medzi prihlásenými účastníkmi vybranej športovej udalosti, zlepšenia bežnej medziľudskej interakcie a v neposlednom rade aj za účelom felxibilného informačného kanálu pre zúčastnené osoby, ktorý by mal informovať o prípadných zmenách týkajúcich sa organizácie danej udalosti.

5.4.4 Vyhľadávanie iných užívateľov

Každá správna aplikácia, ktorá sa zameriava na socializáciu ľudí, by mala už vo svojej základnej podstate poskytovať vyhľadávanie iných užívateľov. Táto funkcia slúži na rozšírenie

zoznamu priateľov alebo na vyhľadávanie ľudí s rovnakými záujmami či potrebami. Ďalšou úlohou tohto nástroja je samozrejme prostredníctvom osobného odporúčania prípadne pozvánky prilákať potenciálneho užívateľa k účasti na konkrétnej udalosti. V prípade navrhovanej aplikácie sa bude jednať o športové udalosti a preto bude aj táto funkcia súčasťou práce.

5.4.5 Hodnotenie užívateľov

Užívateľské hodnotenia vo všeobecnosti slúžia ostatným používateľom aplikácií ako vodítko pri rozhodovaní v každom aspekte, čo má v konečnom dôsledku dopad na to, ako sa bude používanie aplikácie alebo jej časti vyvíjať.

V aplikácii, ktorá je predmetom bakalárskej práce, bude hodnotenie užívateľov inkorporované vo forme možnosti samotného užívateľa ako jedinca hodnotiť športové udalosti, športoviská, alebo aj iné zariadenie, v ktorom užívateľ vykonával aktivitu.

Aplikácia bude navyše umožňovať hodnotenie schopností a vlastností iných užívateľov, ktorý sa zúčastnili spoločnej športovej udalosti a nepriamo prispeje k filtrácii užívateľov, ktorých úmysly nie sú v súlade s myšlienkou navrhovanej aplikácie.

Kapitola 6

Návrh implementácie

Táto kapitola sa bude venovať návrhu celého systému pre potreby fungovania aplikácie. Najprv sa bude kapitola venovať návrhu mobilného klienta a jeho užívateľského rozhrania. Následne sa zamerá na databázu, aplikačný server a aplikačný protokol.

6.1 Mobilný klient

Aplikácia, ktorá bude výstupom tejto práce, bude vo svojej podstate fungovať ako klient, prostredníctvom ktorého je umožnená komunikácia užívateľa so samotným aplikačným serverom 6.3. To v konečnom dôsledku znamená, že aplikácia nebude mať priamy prístup do databázy 6.2, ale bude obdržiavať dáta prostredníctvom komunikácie s aplikačným serverom.

Komunikácia klienta a servera využíva aplikačný protokol, ktorý je detailne popísaný v kapitole 6.4.

6.1.1 Implementačný jazyk

Vzhľadom na to, že sa jedná o aplikáciu vytvorenú priamo pre operačný systém Android a na implementováciu je využité vývojové prostredie Android Studio 4.1, tak sa možnosť výberu implementačného jazyka obmedzuje na jazyky Kotlin a Java.[12]

S jazykom Kotlin nemám žiadne skúsenosti, čo je opačný prípad jazyka Java. Z tohto dôvodu som sa rozhodol, že pre implementáciu aplikácie bude použitý jazyk Java.

6.1.2 Návrh grafického rozhrania

Aplikačné rozhranie platformy Android „API“ poskytuje vývojárom predpripravené widgety, ktoré dokážu zefektívniť prácu s definovaním samotného výzoru aplikácie.

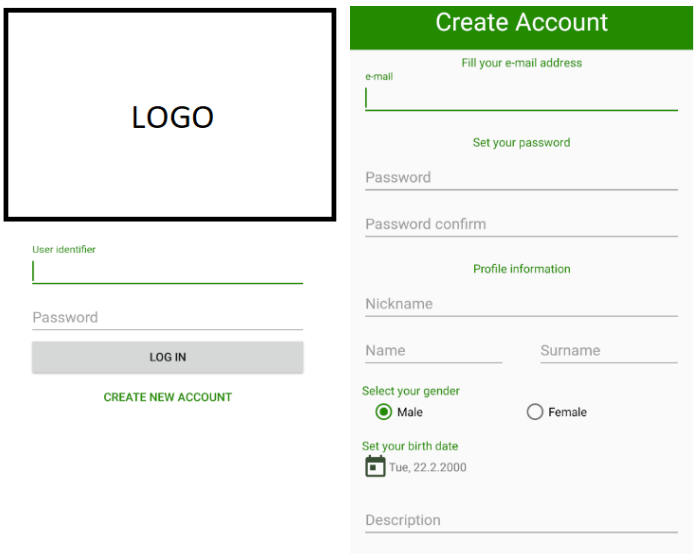
Definovanie rozloženia a sfarbenia jednotlivých widgetov je možné vytvoriť a štylizovať v jazyku Java pomocou dervivátov objektu `View` a ich metódami. Existuje však aj možnosť, ktorá využíva jazyk XML. Pomocou tohto značkovacieho jazyka je možné nadefinovať šablóny pre rôzne rozloženia a za behu programu ich meniť respektíve upravovať. Tento spôsob definovania rozloženia užívateľského rozhrania, napomáha k lepšej čitateľnosti samotného kódu. Dva spomenuté zápisy je možné kombinovať.K objektom definovaným pomocou zápisu v jazyku XML je možné získať refereneciu v jazyku Java pomocou metódy `findViewById()`.[22]

Na navrhovanie užívateľského rozhrania bolo využité Android Studio, ktoré umožňuje pomocou grafického editoru rýchlo navrhnúť užívateľské rozhranie. Výhodou využitia tohto spôsobou návrhu bude, že sa konečný návrh bude dať aplikovať do konečnej aplikácie pomocou menších zmien. Navrhnuté užívateľské rozhranie umožňuje prístup ku všetkým funkciám aplikácie.

Prihlasovanie a registrácia

Prihlasovacie okno sa skladá zo štyroch základných grafických elementov. Obsahuje logo aplikácie pomocou objektu `ImageView`, dve vstupné polia typu `EditText` pre zadávanie prihlasovacích údajov a nakoniec dve tlačidla triedy `Button`. Jedno na odoslanie prihlasovacích údajov a druhé pre zobrazenie registračného formulára, a to v prípade, že si užívateľ potrebuje založiť nový účet.

Registračný formulár pozostáva z niekoľkých vstupných políčok triedy `EditText`, následne z dvoch tlačidiel triedy `RadioButton` pre výber pohlavia a tlačidla pre potvrdenie registrácie, ktoré nie je na nákrese 6.1 viditeľné. Obrázkovú podobu návrhu je možné vidieť na obrázku 6.1.



Obr. 6.1: Prihlasovacie okno(vľavo) a registračný formulár (vpravo).

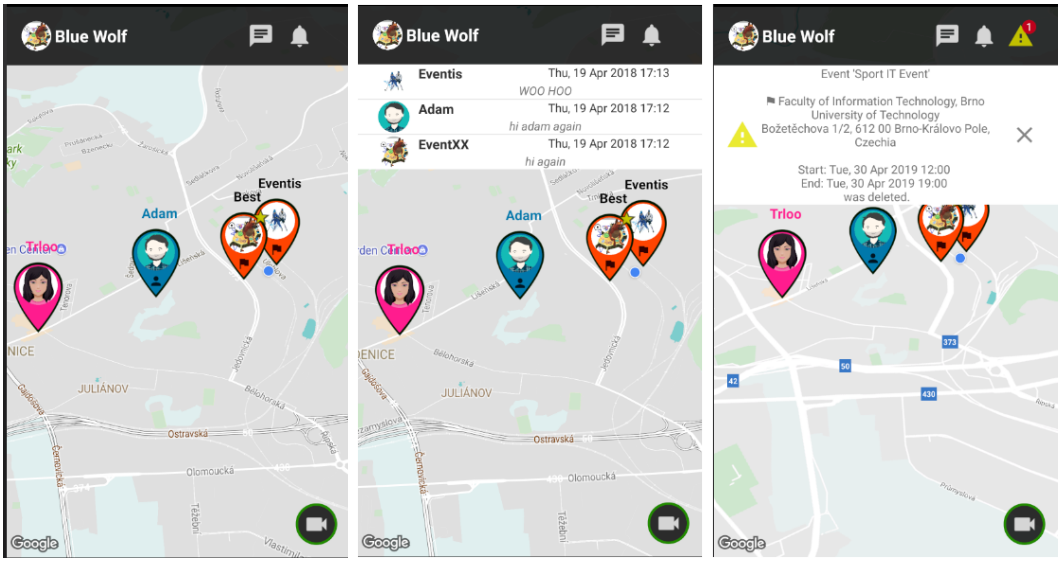
Hlavné okno aplikácie

Hlavná obrazovka sa skladá z hornej lišty a mapy. Horná lišta pozostáva z fotografie užívateľa, za ktorou nasleduje jeho meno a tri tlačidlá typu `Button` na zobrazenie aktualít z chatu, pozvánok a upozornení. Zobrazovanie jednotlivých prvkov v týchto zoznamoch zabezpečuje widget `ListView`. Grafický návrh je zobrazený na obrázkoch 6.2.

Vysúvacie menu

Aplikácia disponuje dvomi vysúvacími menu. Tento významný rys zabezpečuje rozloženie `DrawerLayout`.

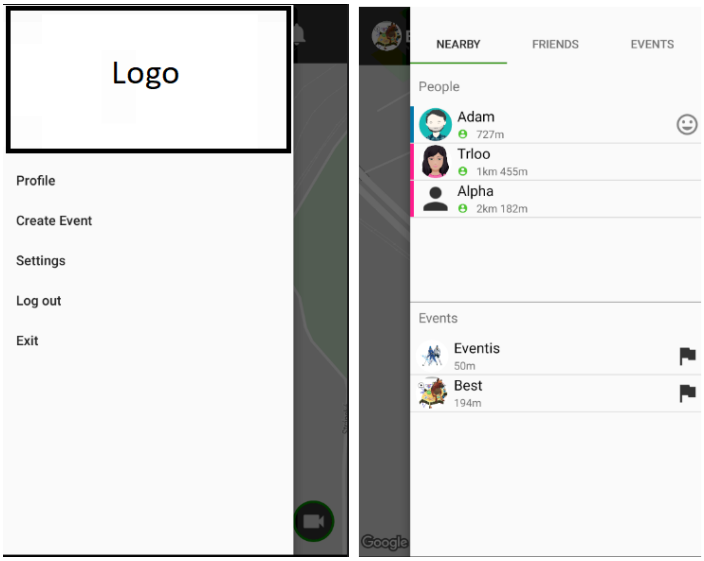
Menu, ktoré sa vysúva pomocou ťahacieho gesta z pravej strany, pozostáva z troch prepínacích kariet. Karta, ktorá sa vždy prednastavene zobrazí ako prvá, obsahuje zoznamy



Obr. 6.2: Hlavná obrázovka aplikácie(vľavo), otvorený zoznam aktuálnych chatov(v strede) a zobrazenie upozornení (vpravo).

užívateľov a udalostí v blízkosti prihláseného užívateľa. ďalšie dve karty obsahujú zoznam priateľov a zoznam udalostí, ktorých sa užívateľ bude účastniť, alebo ktorých je zakladateľom.

Na druhej strane, menu, ktoré sa vysúva z ľavej časti obrazovky obsahuje logo aplikácie a tlačidlá na zobrazenie profilu užívateľa, vytvorenie novej udalosti, odhlásenie sa z aplikácie a ukončenie aplikácie. Pre lepšiu predstavu je možné preskúmať obrázok 6.3



Obr. 6.3: Ľavé vysúvacie menu (vľavo) a pravé vysúvacie menu (vpravo).

Karty

Po kliknutí na užívateľa sa otvorí rozhranie, ktoré pozostáva zo štyroch kariet. V dolnej časti rozhrania sa nachádza navigačné menu triedy `BottomNavigationView`, ktoré umožňuje pohyb medzi jednotlivými kartami a taktiež umožňuje pridať či odobrať priateľa, alebo potvrdiť a zrušiť účasť na udalosti. Medzi jednotlivé karty patria: hlavná karta, chat, hodnotenia, pozvánky.

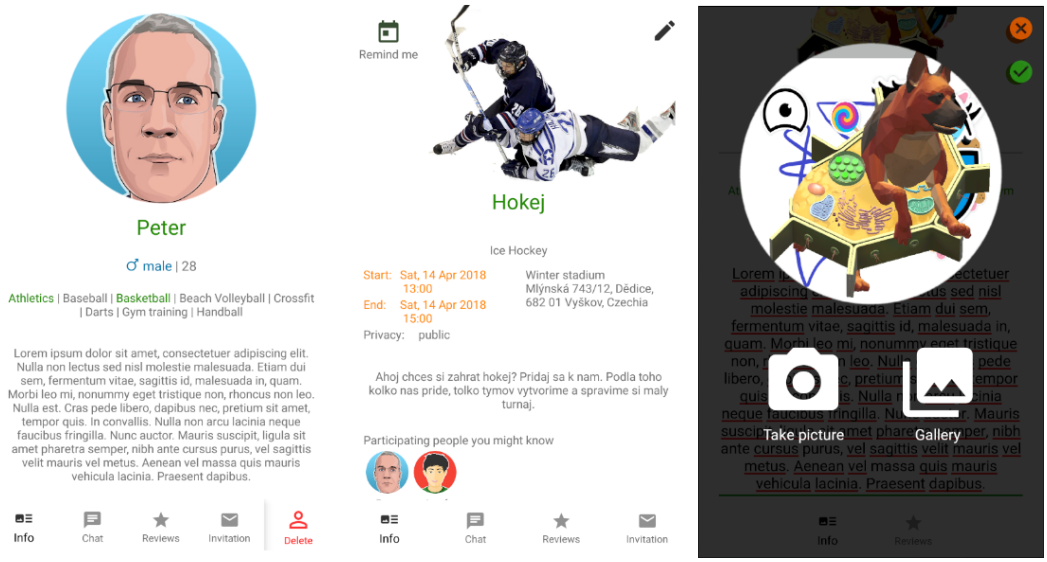
Hlavná karta sa líši pri užívateľovi aj pri udalosti, a zobrazuje základné informácie o danom objekte. V prípade užívateľa, grafické rozhranie pozostáva z fotografie vyobrazenej pomocou objektu `ImageView`, ďalej z názvu, veku, pohlavia a popisu, ktoré sú reprezentované pomocou elementov typu `TextView`. Pri udalosti je to obdobné, no s tým rozdielom, že obsahuje informácie o začiatku, konci, typu, miesta konania a popisu udalosti. V dolnej časti obsahuje hlavná karta udalosti užívateľov, ktorí k nej potvrdili účasť. V ľavej hornej časti karty sa nachádza tlačidlo pre pridanie udalosti do kalendára. V prípade úpravy udalosti alebo profilu sa všetky polia typu `TextView` zamenia za vstupné polia typu `EditText` a ostatné elementy poskytnú rozhranie na ich zmenu po kliknutí na daný element. Pre lepšiu predstavu je grafické rozhranie týchto kariet zobrazené na obázkoch 6.4.

Ostatné karty obsahujú doplňujúce informácie o danom objekte. Jedná sa o chat, hodnotenia a pozvánky k udalostiam.

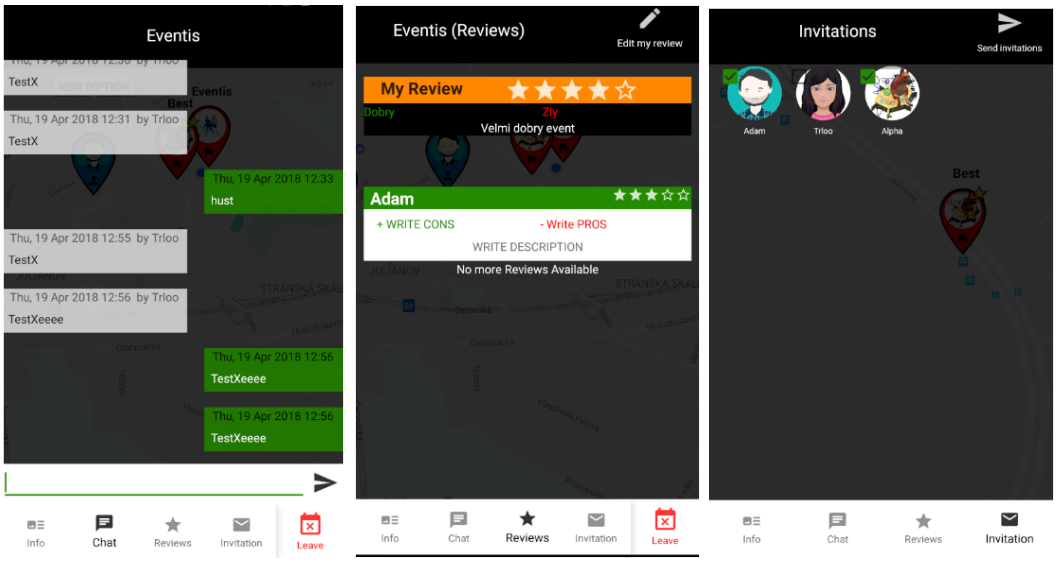
Karta chatu obsahuje v oboch prípadoch konverzáciu, kde správy užívateľa, ktorý aplikáciu používa sú znázornené bielou farbou textu na zelenom pozadí a zarovnané na pravú stranu obrazovky, a správy ostatných užívateľov sú zobrazené čierným textom na bielom pozadí a zarovnané na ľavú stranu obrazovky. Každý grafický element reprezentujúci správu obsahuje datum odoslania správy, autora správy a samotný obsah správy.

Hodnotenia sú zobrazené na osobitnej karte, ktorá umožňuje pridávať, upravovať a mazať jednotlivé hodnotenia, ktorých je užívateľ autorom. Vždy sa ako prvé hodnotenie zobrazí hodnotenie prihláseného užívateľa a je odlíšené od ostatných oranžovou farbou hlavičky a čiernym pozadím. Za ním, smerom na dol, sa zobrazujú hodnotenia ostatných užívateľov. Každé z hodnotení obsahuje číselné vyjadrenie v podobe hviezdičiek, klady, zápory a dodatočnú poznámku.

Karta pozvánok sa obsahom líši ako pri užívateľovi, tak aj pri udalosti. V prípade, že je otvorená karta pozvánok užívateľa, tak sa zobrazia jednotlivé udalosti, na ktoré môže prihlásený užívateľ daného užívateľa pozvať. V prípade, že sa jedná o kartu pozvánok udalosti, tak sa zobrazí zoznam užívateľov, ktorých môže prihlásený užívateľ pozvať na danú udalosť. Samotný zoznam, ktorý karta poskytuje, je realizovateľný za pomoci widgetu `RecyclerView`. Pozvánky je možné odoslať zaškrtnutím jednotlivých políčok zoznamu a následným stlačením tlačidla triedy `Button` v pravom hornom rohu karty. Grafické prevedenie je na obrázkoch 6.5.



Obr. 6.4: Hlavná karta užívateľa(vľavo), hlavná karta udalosti (v strede) a rozhranie pre výber profilovej fotografie (vpravo).



Obr. 6.5: Karta chatu(vľavo), karta hodnotení(v strede) a karta vytvárania pozvánok (vpravo).

6.2 Databáza

Ukladanie dát je neoddeliteľnou súčasťou pre správnu funkcionalitu jednotlivých funkcií, ktoré ponúka aplikácia, a to aj napriek tomu, že k samotným dátam má pristup len pomocou komunikácie s aplikačným serverom. Bez dátového úložiska by sa v podstate nemohli dlhodobo uchovávať žiadne dáta, čo by malo za následok absencie mnohých funkcií, ako je história správ v chate, zobrazovanie a popis jednotlivých udalostí a podobne. Pre uchovávanie dát je preto potrebné implementovať databázu.

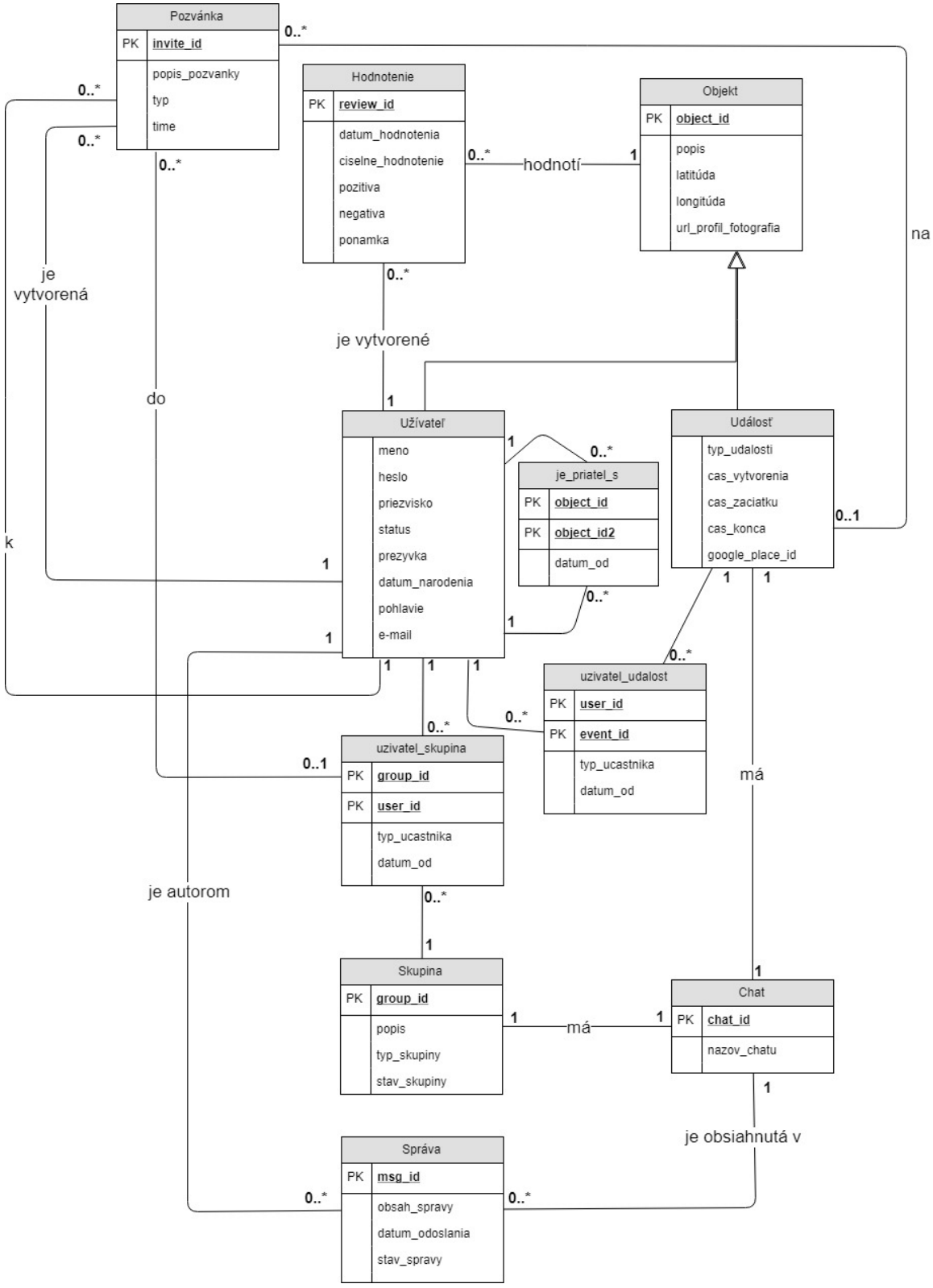
6.2.1 Typ databáz

V dnešnej dobe existujú rôzne typy databáz: relačné databázy, objektovo-relačné a objektové databázy [9]. Relačné databázy sa vyznačujú tým, že uchovávajú konkrétne množiny dát v tabuľkách. Táto koncepcia databázy je odvodená od matematických zásad relačnej algebry.[9] Objektové databázy sa na druhú stranu inšpirovali objektovým prístupom a tvoria triedy, ktoré obsahujú atribúty a metódy. Ich výhodu je, že samotná štruktúra dát v databáze a v programe, ktorý túto databázu využíva, je podobná.[9] Taktiež umožňuje vytvorenie abstraktných dátových typov, ktoré si môže samotný programátor zadefinovať.[9]

Aj napriek všetkým výhodám objektových databáz bude pre účely tejto práce implementovaná relačná databáza. Dôvodom je, že s objektovými databázami nemám žiadne skúsenosti, čo by mohlo ohroziť samotný vývoj aplikácie. Taktiež mám skúsenosti s jazykom MySQL, v ktorom bude táto databáza implementovaná. Pre ukladanie samotných dát bude využitý InnoDB engine.

6.2.2 Návrh databázy

Najčasťejšie sa vizualizuje návrh databázy pocomoc jazyka UML v podobe ER diagramu (entity relationship diagram).[4] Návrh databázy, ktorá bude slúžiť na účel tejto práce, je v podobe ER diagramu znázornená na obrázku 6.6.



Obr. 6.6: ER diagram databázy

6.3 Aplikačný server

Ďalšou súčasťou systému je aplikačný server, ktorý komunikuje s klientom za pomoci socketov a transportného protokolu TCP ako to aplikačný protokol 6.4 vyžaduje. Podľa počtu obsluhovaných klientov môžeme aplikačné servery rozdeliť do dvoch hlavných skupín a to na iteratívne a konkurentné[3].

Hlavný rozdiel medzi iteratívnym a konkuretným serverom je v tom, že iterativný server dokáže v jednom okamihu obsluhovať maximálne jedného klienta. Narozdiel od neho konkurentný server dokáže obsluhovať viacej klientov v jednom okamihu a to vďaka tomu, že pre každého klienta vytvorí nový proces alebo vlákno, ktoré je zodpovedné za obsluhu komunikácie s klientom[3].

Veľkou výhodou iteratívneho serveru je jeho jednoduchšia implementácia a žiadna réžia nad správou procesov alebo vlákien. Aplikácia, ktorá ma byť výstupom tejto práce, však komunikuje so serverom pomocou aplikčného protokolu popísaného v kapitole 6.4, ktorý vyžaduje kontinuálne pripojenie. Pretože aplikáciu využíva viacero užívateľov v jednom časovom okamihu, tak je iteratívny server v tomto prípade nepoužiteľný. Preto je server implementovaný ako konkurentný.

6.3.1 Stavy serveru

Vzhľadom na aplikačný protokol 6.4, ktorý bol návrhnutý pre účely tohto projektu je vhodné rozdeliť jednotlivé príkazy do dvoch skupín, ktoré sú vykonateľné v rôznom stave servera. Server definuje 3 stavy a je implementovaný ako stavový automat.

1. `AUTHORIZATION`
2. `AUTHORIZED`
3. `DISCONNECTED`

AUTHORIZATION

V stave `AUTHORIZATION` sú validné príkazy: `LOG`, `REG` a `EXI`. V prípade úspešného autorizovania užívateľa pomocou príkazu `LOG`, sa zmení stav `AUTHORIZATION` na stav `AUTHORIZED`. V prípade neúspešnej autorizácie sa stav serveru nemení. Pomocou príkazu `EXI` sa ukončí spojenie medzi klientom a serverom a server prejde do stavu `DISCONNECTED`.

AUTHORIZED

Ak sa server nachádza v tomto stave, tak server umožní využívať príkazy: `NOP`, `CED`, `MOV`, `INV`, `JOI`, `MSG`, `RCV`, `EXI`.

V prípade zadania príkazu `EXI` sa server presunie do stavu `DISCONNECTED` a ukončí komunikáciu s klientom. Ak bol využitý iný z príkazov, tak server požiadavku spracuje a podľa pravidiel aplikačného protokolu odošle odpoveď klientovi a server ostáva v stave `AUTHORIZED`.

DISCONNECTED

V tomto stave sa server nachádza, ak bola komunikácia s klientom ukončená. Ak sa server nachádza v stave `DISCONNECTED`, tak je zaručené, že proces alebo vlákno, ktoré obsluhovalo

klienta, môže byť bezpečne ukončené. Vzhľadom k tomu, že klient v tomto stave nie je na server napojený, tak neumožňuje vykonávanie žiadnych z príkazov aplikačného protokolu.

6.3.2 Implementačný jazyk

Konkurentný aplikačný server je možné implementovať v mnohých jazykoch, pre ktoré existujú knižnice na komunikáciu pomocou socketov. Z vlastnej skúsenosti podotýkam, že je vhodnejšie využiť objektovo orientovaný jazyk, kvôli jeho jednoduchšej údržbe a prehľadnosti samotného zdrojového kódu. Veľmi často využívanými objektovo orientovanými jazykmi sú C++, Java. V prípade C++ už mám s vytváraním serveru skúsenosti, no nevýhodou je, že je nutné vytvoriť rôzny kód pre individuálne platformy, čo následne zvyšuje réžiu pri údržbe alebo aktualizácii verzie serveru. Implementáciou serveru v multiplatformovom jazyku Java tento problém nenastáva. Preto je ako implementačný jazyk aplikačného servera použitý jazyk Java.

6.3.3 Nahrávanie súborov

Podľa aplikačného protokou 6.4 sa nahrávanie súborov uksutočňuje pomocou protokolu `FTP` na adresu `upload_url` obdržanú pri zahájovaní komunikácie. Pritom je nutné aby bol názov súboru unikátny a následne vždy zapísať informácie o danom súbore do databázy. Preto je potrebné naimplementovať systém, ktorý sa o generovanie názvu súboru a zápis vytvoreného názvu do databázy postará. Systém je implementovaný v jazyku PHP5.6.

6.4 Aplikačný protokol

Na komunikáciu medzi mobilným klientom a serverom bude využívaný aplikačný protokol, ktorý je definovaný v ďalšom texte. Dôvod využívania vlastného komunikačného protokolou je popísaný v kapitole 5.3.2. Protokol je plne prispôsobený funkčným požiadavkám aplikácie a taktiež je prispôsobený na následné rozšírenie.

6.4.1 Transportný protokol

Ako už je uvedené v kapitole 5.4, aplikácia bude podporovať funkcie, pre ktoré je potrebné dodržiavať poradie jednotlivých packetov a vždy je nutné overiť, či všetky odoslané dáta boli prijaté opačnou stranou. Pre zachovanie týchto vlastností aplikácie nebude ako protokol transportnej vrstvy využitý protokol UDP ale protokol TCP, ktorý chronologické odosielanie a prijímanie dát zaručuje [6].

6.4.2 Základné informácie

V tejto časti textu budú uvedené základné informácie tohto aplikačného protokolu. Text „\0“ určuje ukončovací znak, na základe ktorého sa rozlišuje koniec správy protokolu. Konkrétne sa jedná o nulový znak, teda znak, ktorý nadobúda ASCII hodnotu 0.

Zahájenie komunikácie

Komunikáciu zahajuje po pripojení klienta na server nasledujúcou správou.

$$+OK \quad < proc\_id > @ < ip\_address > \quad < upload\_url > \backslash 0$$

Kde `<proc_id>` určuje identifikačné číslo procesu alebo vlákna, ktoré bolo vytvorené na strane servera pre komunikáciu s novopripojeným klientom. `<ip_address>` je IPv4 alebo IPv6 adresa servera a `<upload_url>` je internetová adresa, na ktorú môže klient nahrávať rôzne súbory, ako napríklad fotografie.

Základné typy odpovedí

Odpovede na jednotlivé príkazy sú rozdelené do dvoch skupín. Prvou skupinou sú kladné odpovede, ktoré sú vo formáte „$+OK\backslash 0$“ a sú odoslané v prípade, že samotné vykonávanie požadovaného príkazu prebehlo v poriadku.

Druhou skupinou sú odpovede záporné, ktoré sú odoslané v prípade, že sa pri spracovaní príkazu vyskytla chyba. Záporná odpoveď má podobu „$-ER \quad < CMD > \quad < msg > \backslash 0$“, kde `<CMD>` je názov príkazu, ktorému odpovedá chyba a `<msg>` je voliteľný argument, ktorý určuje bližší popis chyby.

V prípade, že bude pre určitý príkaz definovaná iná forma odpovede, tak to bude explicitne uvedené.

6.4.3 Notifikácie servera

Dôležitú funkciu, ktorú musí protokol zahŕňať, je upozornenie užívateľa, že nastala istá zmena. Jedná sa hlavne o pozvánku k udalosti. Namiesto toho, aby klient neustále kontroloval stav zmeny, čo vyžaduje spotrebu internetových dát, tak server vykonáva túto aktvitu namiesto neho. V prípade, že nastane zmena, klient musí byť upozornený na túto zmenu aby ju mohol oznámiť užívateľovi zariadenia. Server na notifikáciu neočakáva odpoveď.

Syntax

$$\sim NT \quad < JSON\_notif > \backslash 0$$

Argument `<JSON_notif>` je objekt vo formáte JSON, ktorý obsahuje všetky informácie k danej notifikácii.

6.4.4 Príkaz LOG

Príkaz `LOG` umožňuje prihlásenie užívateľa do systému. Po úspešnom prihlásení môže klient využívať príkazy: `NOP`, `CED`, `MOV`, `INV`, `JOI`, `MSG`, `RCV` popísane ďalej v tomto texte.

Syntax

$$LOG \quad < user\_id > \quad < pass >< \backslash 0 >$$

Kde `<user_id>` predstacuje identifikačné číslo užívateľa, ktorý sa chce prihlásiť do systému. Zároveň argument `<pass>` je prihlasovacie heslo užívateľa s identifikátorom `<user_id>`.

6.4.5 Príkaz RCV

Pomocou príkazu `RCV` klient môže explicitne požiadať o kontrolu aktuality dát v systéme, s ktorými má užívateľ akékoľvek spojenie. Avšak tento príkaz slúži klientovi hlavne na získavanie dat o ľubovoľných objektoch uložených v databáze na základe ich typu a identifikátora.

Syntax

Ak chce aplikácia získať iba základné informácie o všetkých objektoch, tak je syntax nasledovná. Prijimaním hneď všetkých dát, ktoré sú naviazané na užívateľa, by boli neekonomické vzhľadom na využitie internetových dát. Syntax príkazu je nasledovná.

$$RCV\backslash 0$$

$$RCV \quad < JSON\_RCV > \backslash 0$$

Platí, že `<JSON_RCV>` je voliteľný argument. V prípade, že sa vyskytuje v príkaze, potom server eviduje príkaz ako žiadosť o objekt s identifikačným číslom, ktorý je v `<JSON_RCV>` obsiahnutý spoločne s typom objektu, o ktorý klient žiada. V prípade, že príkaz neobsahuje žiaden argument, tak to znamená, že klient žiada explicitne o aktualizáciu dát.

Odpoveď servera

Ak príkaz neobsahuje voliteľný argument `<JSON_RCV>`, tak server odošle odpoveď vo forme notifikácie uvedenej v sekcii 6.4.3.

Inak server odošle odpoveď vo formáte:

$$+OK \quad < JSON\_obj > \backslash 0$$

Tu platí ,že `<JSON_obj>` je reprezentácia už konkretného objektu vo formáte JSON obsahujúca nielen základné, ale všetky dáta.

Pri prípadných komplikáciách server odpovie záporne.

6.4.6 Príkaz MOV

Príkaz `MOV` aktualizuje aktuálnu pozíciu užívateľa na planéte Zem v hodnotách latitúdy a longitúdy geodetického systému WGS84 2.3. Je nutné, aby tento príkaz dokázal spracovať každý server využívajúci tento protokol, pretože bez aktuálnej pozície užívateľa nie je možné vyhľadať iné osoby a miesta v jeho okolí.

Syntax

$$MOV \quad < latitude > \quad < longitude > \backslash 0$$

Kde platí, že argument `<latitude>` je aktuálna hodonota latitúdy a `<longitude>` je aktuálna hodnota longitúdy pozície užívateľa.

6.4.7 Príkaz CED

Príkaz `CED` umožňuje vytvorenie novej udalosti, skupiny, užívateľov a hodnotení, ich následné upravovanie alebo mazanie.

Syntax

$$CED \quad < oper > \quad < JSON\_CED > \backslash 0$$

Kde `<JSON_CED>` je objekt obsahujúci potrebné dáta k vytvoreniu udalosti alebo skupiny užívateľov zapísaný vo formáte JSON. Akcia (vytvorenie, úprava, odstránenie) objektu

<oper>	Odpovedajúca akcia
+	vytvorí nový objekt na základe <JSON_CED>
−	odstráni objekt na základe <JSON_CED>
∼	upraví objekt na základe <JSON_CED>

Tabuľka 6.1: Typy operácii príkazu CED

reprezentovanom <JSON_CED> sa odvíja od hodnoty argumentu <oper>. Jednotlivé hodnoty reprezentujúce akcie sú znázornené v tabuľke 6.1 .

Napríklad, ak sa jedná o založenie udalosti, tak hlavným elementom bude Event a bude obsahovať všetky potrebné atribúty, ktoré danú udalosť reprezentujú. V predchádzajúcom texte je ukázaný ilustračný príklad. Samotná štruktúra objektu môže byť úplne iná. Dôležité je, aby sa v štruktúre zhodovali implementácie klienta aj servera.

Odpoveď servera

Odpoveď servera je v prípade problému pri spracovaní požiadavky záporná. V opačnom prípade je formát odpovede nasledovný.

$$+OK \quad CED \quad < new\_obj\_id > \backslash 0$$

Kde <new_obj_id> je identifikačné číslo novej udalosti alebo skupiny.

6.4.8 Príkaz MSG

MSG umožňuje užívateľovi pridať, upraviť alebo odstrániť ľubovoľnú správu, ku ktorej má prístup.

Syntax

$$MSG \quad < oper > \quad < args > \backslash 0$$

Argument <oper> definuje operáciu, ktorá je vykonaná nad správou alebo chatom. Od typu chcenej operácie sa odvíja ďalšia syntax príkazu (viď. tabuľka 6.2).

<oper>	typ operácie	<args>	Poznámka
+	odoslať novú	<chat_id> <JSON_msg>	<chat_id> je identifikačné číslo chatu, ktorému nová správa prislúcha. <JSON_msg> je správa zapísaná vo formáte JSON.
−	odstrániť správu	<msg_id>	Kde <msg_id> je identifikačné číslo správy, ktorá má byť odstránená.
∼	upraviť správu	<msg_id> <JSON_msg>	<msg_id> je identifikačné číslo správy v chate a <JSON_msg> obsahuje aktualizované hodnoty vo formáte JSON.

Tabuľka 6.2: Typy operácií príkazu MSG

6.4.9 Príkaz INV

Príkaz `INV` patrí medzi zložitejšie príkazy a slúži na pozývanie iných užívateľov do rôznych služieb. Podľa typu pozvania príkaz definuje, o aký typ služby ide.

$$INV \quad < inv\_num > \quad < args > \backslash 0$$

Na základe čísla `<inv_num>` sa odvíjajú aj argumenty označené v popise syntaxe ako `<args>` (viď. tabuľka 6.3).

`inv_num`	typ pozvánky	`<args>`	Poznámka
1	udalosť	`<event_id> [<user_id>]+`	`<event_id>` je identifikačné číslo udalosti. `[<user_id>]+` sú identifikačné čísla užívateľov oddelených medzerou, ktorých chce klient pozvať. Je nutné, aby bolo obsiahnuté najmenej jedno identifikačné číslo užívateľa.
2	zoznam priateľov	`<user_id>`	`<user_id>` je identifkačné číslo užívateľa, ktorého klient žiada o pridanie do zoznamu priateľov.
3	užívateľská skupina	`<group_id> [<user_id>]+`	`<group_id>` je identifikačné číslo skupiny, do ktorej chce užívateľ pozvať iných užívateľov. `[<user_id>]+` sú identifikačné čísla pozvaných užívateľov oddelené medzerou. Je nutné, aby bolo obsiahnuté najmenej jedno identifikačné číslo užívateľa.

Tabuľka 6.3: Typy pozvánok príkazu `INV`

6.4.10 Príkaz JOI

Pomocou tohto príkazu je užívateľovi umožnené pripojiť sa k ľubovoľnej udalosti, ku ktorej bol užívateľ pozvaný iným užívateľom pomocou príkazu `INV`.

Syntax

$$JOI \quad < inv\_id > \backslash 0$$

Kde `<inv_id>` je id pozvánky k určitej udalosti zaslanou inou osobou a obdržanou pomocou notifikácie 6.4.3 od aplikačného servera 6.3.

6.4.11 Príkaz NOP

Každý aplikačný server využívajúci tento protokol definuje čas (tzv. timeout), počas ktorého klient musí preukázať aktivitu od jeho poslednej aktivity, inak server s klientom komuniká-

ciu ukončí. Príkaz `NOP` bol vytvorený kvôli tejto vlastnosti servera a umožňuje obnovovanie poslednej aktivity a udržiavanie spojenia bez toho aby boli odosielané ine data.

Syntax

Syntax pozostáva iba z názvu príkazu a nulového znaku na konci správy. Neobsahuje žiadne argumenty. V prípade iného formátu server považuje príkaz za neplatný a odošle negatívnu odpoveď.

$$NOP\backslash 0$$

6.4.12 Príkaz REG

`REG` umožňuje vytvoriť nový účet pre užívateľa.

Syntax

$$REG \quad < oper > \quad < JSON\_registration > \backslash 0$$

Kde `<JSON_registration>` je objekt reprezentujúci nového užívateľa vo formáte JSON.

6.4.13 Príkaz EXI

Zadaním príkazu `EXI` server ukončí komunkáciu s klientom.

$$EXI\backslash 0$$

6.4.14 Prenos súborov

Súbory môžu byť v databázach ukladané pomocou webových adries, kde je možné dané súbory nájsť. Nahrávanie obrázkov sa vykonáva pomocou protokolu FTP do priestoru prisluchajúcom pre adresu `upload_url`, ktorú pre užívateľa určí server pri zahajovaní komunikácie 6.4.2. Samotný názov pre nahrávaný súbor klient získáva z webovej stránky, ktorá je implementovaná pomocou jazyka PHP. Načítaním stránky a zaslaním informácií o užívateľovi a objekte, ktorému má obrázok prislúchať, PHP skript autorizuje užívateľa a na základe identifikátoru objektu a aktuálneho serverového času vygeneruje názov pre daný súbor. Ak priebeh prenosu súboru nebol ničím prerušený a súbor je bezpečne prenesený, tak sa adresa umiestnenia nového súboru uloží do databázy.

Kapitola 7

Implementácia

Kapitola sa bude venovať implementácii systému, ktorý bol predmetom tejto práce. V prvej časti kapitoly bude popisaný spôsob implementácie aplikačného servera a druhá časť kapitoly sa zameria na implementáciu mobilného klienta.

7.1 Server

Kľúčovými úlohami serveru sú nadviazanie komunikácia s databázou a získanie potrebných dát pre obsluhovaných klientov, udržiavanie stavu dát, ktoré boli už jednotlivým klientom odoslané, aby nedošlo k odosielaniu zbytočných dát, a samozrejme podpora vykonávania všetkých príkazov definovaných aplikačným protokolom popísaným v kapitole 6.4. Samotná stavba programu je v podobe diagramu tried znázornená na obrázku 7.1. V samotnom diagrame sú znázornené najdôležitejšie triedy, atribúty a metódy.

7.1.1 Komunikácia s databázou

Komunikáciu s databázou zabezpečuje trieda `DBManager`. Pripojenie do samotnej databázy a vykonávanie MySQL dotazov zabezpečuje technológia JDBC (Java database connection) , ktorá je súčasťou Java SE. Všetky dotazy, ktoré sú potrebné pre správny chod servera sú namapované v atribúte `qArray` (Query array), ktorý je dátového typu `Map<String, String>`, pričom kľúčom je názov dotazu a hodnotou je dotaz samotný. Trieda `DBManager` v sebe zahrňuje metódy na zahájenie spojenia s databázou, metóda `connect()`, a na ukončenie spojenia s databázou, metóda `disconnect()`. Ďalšie metódy, ktoré trieda obsahuje, slúžia na transformáciu dotazov zo spominaného atribútu `qArray`, na tvar založený na základe vstupných parametrov metódy. Dôsledkom tejto transformácie dotazov je to, že všetky získané dáta z databázy sú mierené na obsluhovaného klienta a jeho požiadavky.

7.1.2 Uchovávanie dát z databázy

Na uchovávanie potrebných dát pre klienta boli implementované triedy `DBAttr` a `DBObject` a jej deriváty.

Trieda `DBAttr`

Trieda bola vytvorená na reprezentovanie a uchovávanie dát jednotlivých stĺpcov databázy. Pozostáva zo štyroch hlavných atribútov. Atribút `String DBColumnName` udržiava informáciu o názve stĺpca v databáze a jeho hodnota je konštantná. Ďalší atribút `String value`

slúži na uchovávanie samotej hodnoty daného stĺpca, ktorý trieda reprezentuje. Tretí atribút `String JSONName` definuje názov atribútu, ktorý reprezentuje hodnotu vo formáte JSON. Atribút `JSONName` sa využíva pri generovaní objektov vo formáte JSON, ktoré sa odosielajú klientovi. Posledným dôležitým prvkom triedy je atribút `boolean changed`. Hodnota `changed` sa nastaví na hodnotu `true` v prípade, že po aktualizácii dát z databázy sa daná hodnota atribútu `value` zmenila. To následne umožňuje pri generovaní notifikácií vybrať iba zmenené položky.

Trieda `DBObject` a jej deriváty

Trieda `DBObject` je základom pre jej deriváty, `DBObject_User, DBObject_Event, DBObject_Group, DBObject_Chat, DBObject_Invitation, DBObject_Message`, ktoré už reprezentujú samotné dáta v logickej štruktúre. To znamená, že napríklad `DBObject_Event` nereprezentuje samotnú reláciu databázy, ale obsahuje všetky potrebné data pre reprezentovanie daného objektu v aplikácii(v tomto prípade udalosti). Taktiež z toho vyplýva, že jednotlivé deriváty obsahujú iné deriváty. Napríklad `DBObject_Event` má ako jeden z atribútov objekt `DBObject_Chat`, ktorý obsahuje hneď pole objektov `DBObject_Message`, ktorý je taktiež derivátom triedy `DBObject`.

Atribútmi triedy sú vektor `Vector<DBAttr> attributes`, ktorý obsahuje objekty typu `DBAttr` a podľa toho, o aký derivát sa jedná, tak sa jeho veľkosť a jednotlivé hodnoty líšia. Trieda taktiež obsahuje atribút `boolean attributeChanged`, ktorý nadobudne hodnotu `true` v prípade, že akýkoľvek objekt z vektoru `attributes` zmenil svoju hodnotu. To predchádza zbytočnému kontrolovaniu každého objektu z vektoru `attributes` v prípade, že nenastala žiadna zmena. Všetky zmeny hodnôt objektu sa získávajú vo formáte JSON pomocou volania metódy `getUpdate()`.

7.1.3 Obsluha klienta a vytváranie notifikácii

Trieda `NotificationManager`

`NotificationManager` je trieda, ktorá spravuje všetky už spomenuté triedy. Trieda na uchovávanie jednotlivých inštancií derivovaných tried využíva štyri vektory `Vector<DBObject_User> users`, `Vector<DBObject_Event> events`, `Vector<DBObject_Group> groups`, `Vector<DBObject_Invitation> invitations` a na uchovávanie dát o užívateľovi, ktorého vlákno obsluhuje, slúži atribút `DBObject_User myProfile`. Trieda `NotificationManager` obsahuje aj odkaz na objekt dátového typu `DBManager` 7.1.1, kde pomocou vyvolávania jeho metód udržuje aktualitu dát, ktoré sa odosielajú na stranu klienta. Trieda takiež definuje niekoľko dôležitých metód. Metóda `checkForUpdates()` volaním metód `updateUsers(), updateEvents(), updateGroups() a updateInvitations()`, vytvorí, aktualizuje alebo vymaže jednotlivé objekty z vektorov `users, events, groups, invitations`, tak, aby zodpovedali aktuálnemu stavu z databázy. Po získaní aktualizácií jednotlivých objektov je možné získať samotné zmeny vo formáte JSON pomocou metódy `getUpdates()`. Spustenie celého cyklu: aktualizovanie hodnôt a získanie aktualizácií umožňuje metóda `execute()`.

Trieda `ServerThread`

Trieda `ServerThread` predstavuje vlákno, ktoré obsluhuje jedného klienta. Hlavnou úlohou je parsovanie prijatých správ od klienta, vytvorenie korešpondujúcej odpovede a taktiež posielanie notifikácií v určitom časovom intervale. Parsovanie samotných správ zabezpečuje metóda `parseMessage()`, ktorá určí, o aký príkaz sa jedná a na základe toho vyvolá

metódu, `executeXYZ()`, kde XYZ je príkaz aplikačného protokolu 6.4. Príslušná metóda požiadavok spracuje a na jeho základe vytvorí odpoveď, ktorú odošle pomocou metódy `sendMessage()`. Generovanie notifikácií prebieha v určitom časovom intervale, ktorý je definovaný konštantou `NOTIFICATION_CHECK_TIMEOUT`. V prípade, že užívateľ pomocou príkazov aplikačného protokolu 6.4 vykonal zmeny hodnôt v databáze, tak sa hodnota `NOTIFICATION_CHECK_TIMEOUT` ignoruje a generovanie notifikácie sa vykoná ihneď.

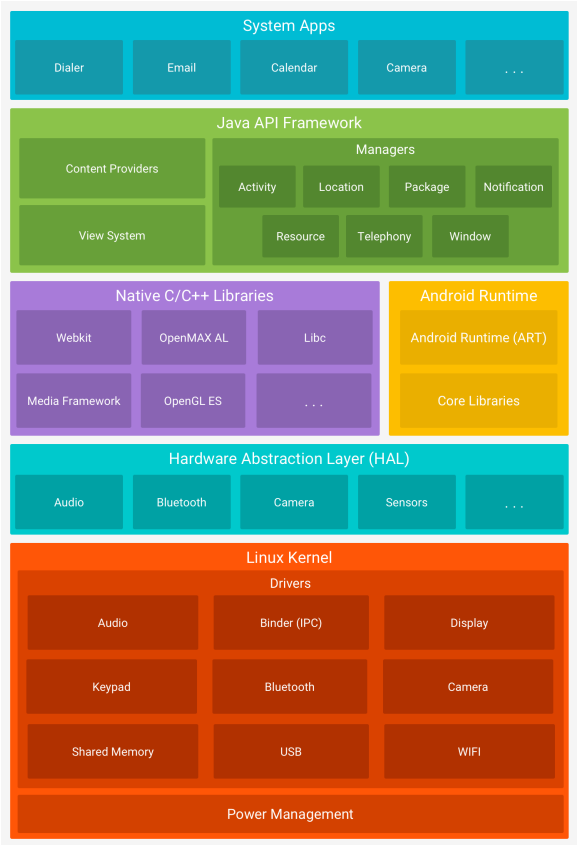
Obr. 7.1: Diagram tried aplikačného serveru

7.2 Klient

Mobilný klient pre platformu Android pozostáva z dvoch hlavných modulov, backendu a frontendu. Backend zaobstaráva prijímanie a odosielanie dát na server, ich následne spracovanie a taktiež spravuje a uchováva prijaté dáta. Frontend vytvára užívateľské rozhranie pre osobu, ktorá aplikáciu v danom čase využíva a na základe jej požiadavok vyvolá potrebné metódy backendu.

7.2.1 Architektúra platformy Android

Achitektúra platformy android pozostáva zo šiestich hlavných komponentov: `Linuxové jadro`, `Hardvérová abstrakčná vrstva`, `Natívne C/C++ knižnice`, `Android Runtime`, `Java API Framework`, `Systémové aplikácie`. Samotná hierarchia týchto komponentov je znázornená na obrázku 7.2



Obr. 7.2: Hlavné komponenty architektúry platformy Android.
zdroj [16]

Linuxové jadro

Linuxové jadro, je najzákladnejším komponentom celej architektúry. Jeho funkciou je spravovanie celého systému. Ma na starosti správu procesov, vlákien, sieťovú komunikáciu, nízkoúrovňovú správu pamäti zariadenia a iné.[2, 16] Služby tohto kompenentu využívajú priamo alebo nepriamo všetky ostatné menované komponenty.[16]

Hardvérová abstrakčná vrstva

Hardvérová abstrakčná vrstva (ďalej len HAL) poskytuje rozhrania pre umožnenie využívania hardvérových prvkov, akými sú napríklad kamera, bluetooth, mikrofón a iné. Pre každý hardvérový prvok je implementovaný samostatný modul knižníc, ktoré práve umožňujú využívanie daných prvkov komponentom Java API Framework.[16] Vďaka HAL je možné napríklad to, že programátor, ktorý vytvára aplikáciu pomocou Java API Frameworku, je schopný využívať v aplikácii kameru zariadenia.

Android Runtime

Android Runtime (ďalej len ART) slúži na spúštanie viacerých virtuálnych strojov na zariadeniach s nízkou pamäťovou kapacitou za pomoci spúštania bajtkódových DEX súborov, ktoré sú špeciálne navrhnuté pre platformu Android takým spôsobom, aby využívali čo najmenej pamäťového priestoru.[2, 16] Do verzie API 21 je namiesto ART využívaná technológiou Dalvik[16]

Natívne C/C++ knižnice

Natívne C/C++ knižnice sú základným stavebným prvkom pre komponenty Android Runtime a HAL. Platforma Android sprístupňuje funkcionalitu týchto knižníc za pomoci komponentu Java API Framework. Príkladom je prístup k OpenGL API a OpenGL ES pre prípad využívania manipulovania 2D alebo 3D grafických prvkov v aplikácii.[2, 16]

Java API Framework

Prístup k celej sade funkcií operačného systému Android umožňujú pre vývojára aplikačné rozhrania tohto komponentu. Predstavujú základné stavebné bloky, ktoré využitím ich jednotlivých súčastí umožňujú vytvorenie aplikácie.[16]

Medzi tieto bloky patrí **View System**, ktorý sa využíva na vytváranie užívateľského rozhrania, **Resource Manager** sprístupňujúci zdroje iné ako súbory obsahujúce zdrojový kód, jedná sa napríklad o obrázky, štýly, zvukové súbory a iné. Ďalším blokom je **Activity Manager**, ktorý spravuje životný cyklus aktivít. Bližší popis toho, čo je aktivita je uvedený v sekcii 7.2.3. Blok **Content Provider** zase sprístupňuje dáta iných aplikácií, alebo zdieľa dáta iným aplikáciám. Poslednou súčasťou je **Notification Manager**, ktorý umožňuje zobrazovanie vlastných upozornení na stavovej lište zariadenia.[2, 16]

Systémové aplikácie

Jedná sa o set aplikácií pre zabezpečenie zakladnej funkcionality zariadenia. Sú to aplikácie, ktoré umožňujú prístup k elektronickej pošte, SMS správam, kontaktom, kalendáru a ďalším.[16]

Tieto aplikácie môžu byť využívané inými aplikáciami. V aplikácii, ktorá je predmetom tejto práce, sa využíva práve systémová aplikácia kalendár, na vytvorenie pripomienky o začiatku, konci a mieste konania udalosti.

Implementácia viacvláknových aplikácii

Pri každom novom spustení aplikácie systém vždy vytvorí nový proces, ktorý aplikácii prislúcha. Taktiež vytvorí nové hlavné vlákno aplikácie. Hlavné vlákno je zodpovedné za

odosielanie rôznych udalostí, ktoré sa týkajú komponentov tvoriacich užívateľské rozhranie. Preto sa vo väčšine prípadov hlavné vlákno zvykne nazývat aj vlákno užívateľského rozhrania (Ui thread). [17] Pravidlom je, že všetky metódy, ktoré akýmkoľvek spôsobom ovplyvňujú jednotlivé widgety užívateľského rozhrania, musia vykonávať zmeny na tomto vlákne.[17]

Preto pri implementácii viacvláknových aplikácií vzniká problém, keď iné vlákno ako vlákno užívateľského rozhrania, potrebuje na základe zmeny dát alebo výpočtov ovplyvniť užívateľské prostredie. Pre prístup k vláknu užívateľského rozhrania z iného vlákna poskytuje metódy `runOnUiThread(Runnable)`, `post(Runnable)`,`postDelayed(Runnable, long)`. [17] Taktiež je možné využitie triedy `AsyncTask`, ktorá implementuje dve základne metódy `doInBackground()`, ktorá sa vykonáva na inom vlákne ako vlákne užívateľského rozhrania a metóda `onPostExecute()`, ktorá sa vykonáva na vlákne užívateľského rozhrania a tým pádom umožňuje zmeny samotných widgetov na základe výsledku.[17]

7.2.2 Backend

Backend je v aplikácii samostatný modul, ktorý zastrešuje všetku funkcionalitu, ktorá je potrebná pre komunikáciu, spracovanie a udržovanie dát zo strany servera. Jedná sa o nezávislý modul, čo znamená, že by fungoval aj samostatne bez modulu Frontend. Diagram tried tejto časti aplikácie je zobrazený na obrázku 7.3. V samotnom diagrame sú znázornené najdôležitejšie triedy, atribúty a metódy.

Trieda `CThread`

Názov triedy `CThread` je odvodený od spojenia connection thread. Ako sam názov napovedá, jedná sa o triedu, ktorej samotná inštancia pracuje na samostatnom vlákne aplikácie. `CThread` zabezpečuje naviazanie kontaktu s aplikačným serverom pomocou objektu typu `SSLSocket`. Na zahájenie a ukončenie spojenia so serverom slúžia metódy `connnectServer()` a `disconnect()`. Samotné čítanie dát zo socketu zabezpečuje metóda `receiveMessage()`, ktorá vráti ako návratovú hodnotu vždy jednu správu. Detekovanie konca správy funguje na základe popisu aplikačného protokolu 6.4.

Trieda `CThread` v sebe uchováva referenciu na objekt typu `MessageParser`. Volaním metód tohto objektu zaistí spracovanie prijatej správy. Bližší popis triedy `MessageParser` je možné nájsť v ďalšom texte.

Trieda `MessageParser`

Táto trieda definuje metódy, za pomoci ktorých je možné prijatú správu dekódovať a na jej základe vykonať odpovedajúce úkony. Najdôležitejšou metódou triedy je `parseMessage()`, ktorá rozloží správu na jej jednotlivé časti, určí, o aký typ odpovede sa jedná, a na základe toho vyvolá ďalšie potrebné metódy. Atribútom triedy `MessageParser` je tiež referencia na objekt typu `ObjectManager`. Dôvodom existencie tohto atribútu je skutočnosť, že pri situácii, kedy klient príjme správu, ktorá obsahuje informácie o objekte, ktorý je pod kontolou objektu na ktorý atribut referuje, tak je nutné aby boli všetky potrebné hodnoty aktualizovaného objektu zapísané.

Trieda `MessageCreator`

Trieda `MessageCreator` je presným opakom triedy `MessageParser`. Síce slúži tiež na komunikáciu so serverom, avšak jej úlohou je vytváranie správ pre server na základe požiadavok užívateľa, ktorý operuje nad danou aplikáciou a samotný zápis správ na komunikačný socket.

Trieda `ObjectManager`

Trieda `ObjectManager` slúži na spravovanie objektov, ktoré klient obdržal zo strany servera. Jedná sa objekty typu `User`, `Event`, `Invitation`, `Group`, `Review`, `Chat`, `ChatMessage`, ktoré sú udržiavané vo vektoroch. `ObjectManager` poskytuje množstvo metód na spravovanie daných objektov a to na pridanie a odobranie objektu, aktualizovanie hodnôt objektu, vyhľadávanie objektov na základe rôznych parametrov a kontroly existencie jednotlivých objektov. Na samotný objekt reprezentujúci prihlaseného užívateľa referuje atribút `currentUser`.

Trieda `StaticLibrary`

`StaticLibrary` je pomocná trieda, ktorá obsahuje výhradne statické metódy. Obsahuje napríklad metódu `calculateDistance()`,ktorá na základe vzorca 3.3 vypočíta vzdialenosť dvoch bodov a to v metrických alebo imperiálnych jednotkách. Ďalšou dôležitou metódou je `isNearby()`, ktorá pomocou metódy a vstupného parametru vzdialenosti určí, či sa jedná o objekt, ktorý je v určitej blízkosti od užívateľa.

Nahrávanie obrázkov

Pre účely nahrávania obrázkov na server boli vytvorené triedy `FTPImageUpload` a `HTTPImageUrlGen`. Postup nahrávania obrázka je nasledovný.

Pomocou objektu triedy `HTTPImageUrlGen` pripojí na adresu `upload_url` získanú na základe aplikačného protokolu 6.4. Využitím metódy POST zašle všetky potrebné informácie o objekte, ktorému obrázok prislúcha, a taktiež informácie potrebné na autorizáciu užívateľa. Po zaslaní týchto informácií PHP skript, ktorý beží na strane serveru, skontroluje, či sa jedná o užívateľa, ktorý má právo na zmenu obrázku pre daný objekt.

V prípade, že je všetko v poriadku, tak skript vygeneruje názov súboru na základe jeho identifikačného čísla a systémového času na strane servera. Následne sa pomocou objektu triedy `FTPImageUpload` nahraje obrázok na server a uloží sa jeho nový názov do databázy.

Pri implementácii triedy `FTPImageUpload` bola využitá knižnica *Apache Commons Net 3.6* a pri implementácii triedy `HTTPImageUrlGen` knižnica *Apache HttpComponents 4.3*.

Obr. 7.3: Diagram tried backendu mobilného klienta

7.2.3 Frontend

Na rozdiel od backendu, frontend zastrešuje zobrazovanie dát na uživateľskom rozhraní. Frontend sprostredkuváva prístup a samotné dáta backendu užívateľovi a definuje správanie aplikácie na základe gest užívateľa. Preto patrí medzi najdôležitejšie prvky implementácie konečného produktu. Diagram tried frontendu je možné vidieť na obrázku 7.5. V samotnom diagrame sú znázornené najdôležitejšie triedy, atribúty a metódy.

Notifikovanie o zmenách

Na notifikovanie frontendu bola vytvorená trieda `UiNotifier`. Jej metódy volá prevažne trieda `MessageParser` ale aj derviáty triedy `AsyncTask` popísane v nasledujúcom texte.

Trieda definuje metódy pre notifikovanie rôzných skupín grafického užívateľského rozhrania. Napríklad metóda `notifyAllUsersUiElements()`, zabezpečí to, že pri zmene hodnoty objektu, ktorý reprezentuje ľubovoľného užívateľa, sa zobrazí aj na samotnom grafickom rozhraní aplikácie. Taktiež existujú metódy `notifyAllEventsUiElements()`, ktorá zabezpeči zobrazenie zmien pre objekty reprezentujúce udalosti. Na aktualizovanie grafického rozhrania chatu je implementovaná metóda `notifyChat()` a presun kamery na mape na pozíciu zameraného užívateľa zabezpečuje metóda `updateCameraPosition()`. V prípade, že je nutné aktualizovať všetky prvky grafického rozhrania, je využitá funkcia `notifyEverything()`.

AsyncTask

Deriváty triedy `AsyncTask`, bližšie popísanej v sekcii 7.2.1, boli využívané na načítavanie hodnotení a správ chatu zo serveru `AsyncLoadReviews` a `AsyncLoadChat`, kde sa pomocou metódy `doInBackground()` čaká na odpoveď servera na zaklade ktorej sa mení samotné užívateľské rozhranie pomocou metódy `onPostExecute()`. Ďalší derivát tejto triedy `DownloadBitmap`, zabezpečoval sťahovanie obrázkov v pozadí a následné zobrazenie na užívateľskom rozhraní.

Aktivity

Aktivita (`Activity`) je základným komponentom aplikácie, ktorá umožňuje iterakciu užívateľa s aplikáciou a riadi jej tok. Jednoducho povedané, aktivita predstavuje pre užívateľa jednu obrazovku aplikácie.[15] Samotný životný cyklus aktivity je znázornený na obrázku7.4

Aplikácia, ktorá je predmetom tejto práce, sa skladá z troch hlavných a niekoľkých vedľajších aktivít. Hlavnou aktivitou je `MainActivity`, ktorá sa spúšťa po spustení samotnej aplikácie a umožňuje prihlásenie užívateľa alebo možnosť vytvorenia registrácie. Aktivita `SignInActivity` predstavuje obrazovku, ktorá poskytuje užívateľské rozhranie a samotný proces registrácie užívateľa. Treťou najhlavnejšou aktivitou je aktivita `MapActivity`, ktorá predstavuje hlavné prostredie aplikácie.

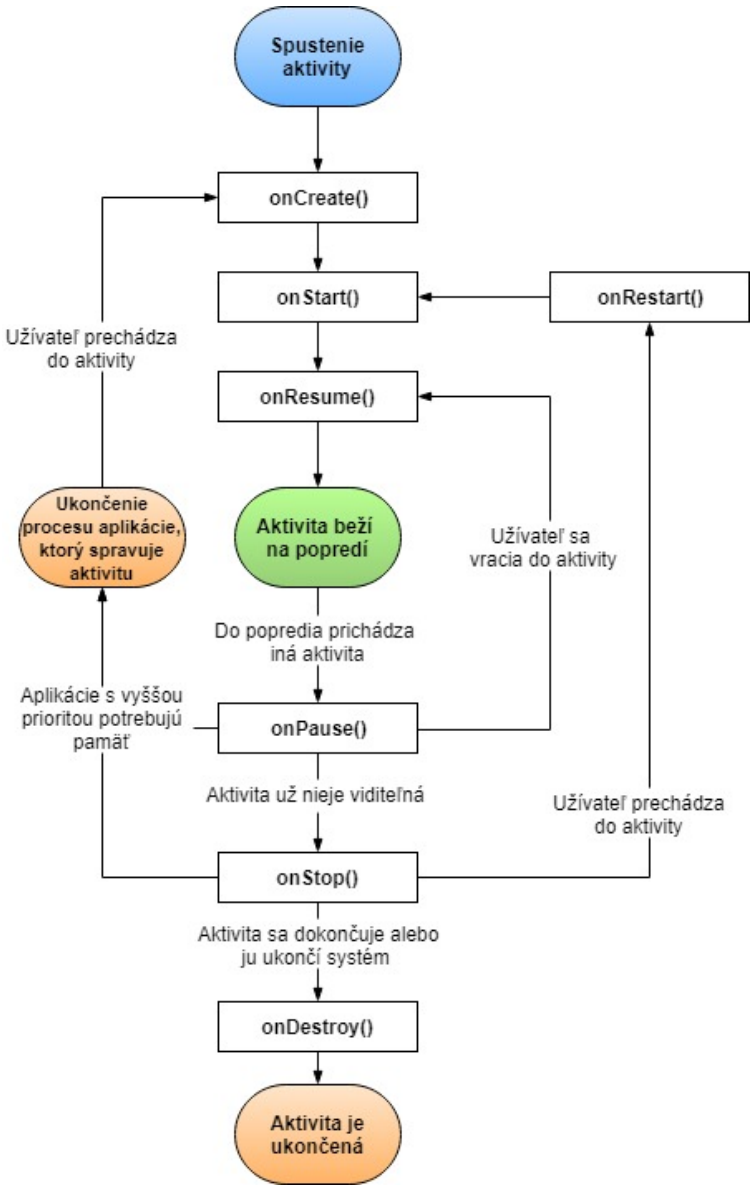
Trieda `LayoutManager`

`LayoutManager` je trieda, ktorá bola implementovaná na účel spravovania a manipulovania užívateľského rozhrania aplikácie. Všetky metódy, ktoré zapríčiňujú zmeny užívateľského rozhrania, sú vykonávané na vlákne užívateľského rozhrania. Metódy tejto triedy sú pri zmenách dát backendu volané metódami objektu typu `UiNotifier`. Táto trieda implementuje

metódy, ktoré nastavujú callbacky jednotlivých elementov, taktiež spravujú zobrazovanie, viditeľnosť či samotnú deštrukciu widgetov.

Reprezentovanie užívateľa a udalosti

Na reprezentovanie užívateľa a udalosti v užívateľskom rozhraní boli vytvorené triedy implementujúce rozhranie `ObjectOnMapUi`. Konkrétne sa jedná o triedy `UserOnMapUi` a `EventOnMapUi`. Jedným z atribútov týcho tried je odkaz na samotný objekt, ktorý reprezentujú. Jedná sa o objekt, ktorý je pod správou objektu triedy `ObjectManager`, a umožňuje prístup užívateľskému rozhraniu priamo k uloženým dátam. Tieto triedy ďalej implementujú metódy, ktoré sú potrebné na reprezentovanie objektu na mape alebo jeho karte profilu. Na vytváranie



Obr. 7.4: Životný cyklus aktivity.
zdroj [21], spracovanie: vlastné

obrázku samotného markera zobrazujúceho sa na mape slúži metóda `createPin()`, ktorá pomocou maskovania profilovej fotografie a vygenerovaného vzhľadu na základe vzťahov užívateľa k objektu vytvorí pomocou metód triedy `BitmapFactory` potrebný obrázok.

7.3 Komunikácia klienta a servera

Komunikácia klienta a servera je implementovaná pomocou socketov triedy `SSLSocket` na strane klienta a triedy `SSLServerSocket` na strane servera. Komunikácia prebieha pomocou spojenia TCP s využitím aplikačného protokolu definovaného v sekcii 6.4. Vzhľadom na fakt, že sa počas komunikácie posielajú citlivé dáta, ako je prihlasovacie meno, heslo a aktuálna pozícia užívateľa, tak som sa rozhodol implementovať zabezpečenie komunikácie pomocou technológie *SSL/TLSv1.2* s výmenou kľúčov a ich šifrovaním pomocou algoritmu *RSA*. Na generovanie súkromného a verejného kľúča, ktoré sú potrebné pre správne fungovanie komunikácie, som využil nástroj `keytool`, ktorý je súčasťou JDK 10. Samotné zahájenie komunikácie medzi klientom a serverom iniciuje klientska aplikácia volaním metódy `startHandshake()` triedy `SSLSocket`.

Obr. 7.5: Diagram tried frontendu mobilného klienta

Kapitola 8

Testovanie

Testovanie aplikácie a serveru prebiehalo počas vývoja aplikácie pomocou Android Emulátoru, ktorý umožňoval simulovanie behu aplikácie na rôzných zariadeniach a tým, že aplikácia na emulovanom zariadení bola priamo napojená na aplikačný server, tak sa ruka v ruke s aplikáciou testovala aj funkčnosť aplikačného serveru. Keď nastal čas, kedy bol produkt považovaný za dokončený, tak bolo nutné otestovať jeho funkčnosť a spoľahlivosť v reálnom prostredí.

8.1 Testovanie systému v reálnom prostredí

Aplikácia bola nasadená do reálneho prostredia pomocou trinástich osôb, kde dvanásť osôb vo vekovom rozmedzí 19 až 23 rokov skúšalo funkcionalitu aplikácie v teréne a jedna osoba kontrolovala aktuálny stav serveru.

Osoby, ktoré aplikáciu testovali v teréne, boli informované iba o tom, na čo je aplikácia určená. S užívateľským rozhraním a architektúrou aplikácie neboli dané osoby vôbec oboznámené. Tento prístup umožňoval testovanie užívateľského rozhrania zároveň s testovaním funkcionality aplikácie.

Úlohou účastníkov testovacej akcie bolo vyskúšať všetky funkčné prvky apikácie, sledovať ovládateľnosť aplikácie, intuitívnosť užívateľského rozhrania a prípadne si zapisovať iné pripomienky, respektíve problémy, ktoré sa počas testovania aplikácie naskytli. Zariadenia, ktoré slúžili na účel testovania, pokrývali API verzie operačného systému od verzie 19 až po verziu 25 vrátane. Prístroje, ktoré obsahujú API verziu 26 a 27, sa nám pre túto etapu testovania bohužial nepodarilo zaobstarať.

Dĺžka etapy testovania v reálnom prostredí trvala 48 hodín, počas ktorých užívatelia využívali služby danej aplikácie. Po ukončení tohto časového úseku testéri vyplnili na základe vlastných skúseností testovací dotazník.

Samotný dotazník pozostával z troch častí. Prvá časť obsahovala otázky mierené na funkčnosť aplikácie, druhá časť predstavovala zhodnotenie užívateľského rozhrania aplikácie a tretia textová časť umožňovala popis postrehov a poznámok testéra k aplikácii. V prvej a druhej časti dotazníka bolo možné odpovedať výberom jednej z piatich predpripravených odpovedí, pričom najpozitívnejšia odpoveď predstavovala hodnotu 5 a smerom k negatívnejším odpovediam sa hodnota odpovede zmenšovala až na hodnotu 1. Testovací dotazník je priložený v prílohe A.

8.2 Výsledky testovania systému

Pre potvrdenie naplnenia požiadavok pre daný systém boli vytvorené hypotézy:

1. Užívateľské rozhranie aplikácie je dostatočne intuitívne pre bežného užívateľa.
2. Aplikácia trpí malou mierou chybovosti.
3. Komunikácia medzi klientom a serverom je spoľahlivá.

Pre potvrdenie a vyvrátenie hypotéz boli použité zaznamenané hodnoty z vyplnených dotazníkov. Prvá časť dotazníka slúžila na potvrdenie hypotéz 2 a 3, a druhá časť dotazníka na potvrdenie hypotézy 1. Pre splnenie ľubovoľnej hypotézy bolo určené, že z otázok prisluchajúcich hypotéze je nutné získať aspoň 70% zo súčtu všetkých bodov týchto otázok.

Výsledky testovania sú naslednovné. Hypotéza „*Užívateľské rozhranie aplikácie je dostatočne intuitívne pre bežného užívateľa.*“ bola potvrdená na základe zisku 84% bodov z prisluchajúcich otázok. Druhá hypotéza „*Aplikácia trpí malou mierou chybovosti.*“ bola rovnako potvrdená hodnotou 92%. Posledná hypotéza „*Komunikácia medzi klientom a serverom je spoľahlivá.*“ bola taktiež potvrdená s percentuálnym ziskom 95%.

Z vyplnených dotazníkov boli ďalej nadobudnuté tieto poznatky:

1. Vysúvacie menu bolo v niektorých prípadoch objavené až po dlhšej dobe používania aplikácie.
2. Za veľkú výhodu aplikácie je považovaný spôsob zobrazovania udalostí a užívateľov na mape a zaostrenie na ich pozíciu.
3. Užívatelia by uvítali nastavenia farieb vytváranych udalostí.
4. Bolo by dobré, ak by aplikácia umožňovala užívateľovi prispôsobiť farby grafických elementov aplikácie.
5. Viacerým užívateľom sa nepáči farebné spracovanie karty chatu.
6. Testéri potvrdili, že užívateľské rozhranie je jednoduché na používanie a umožňuje jednoduchý prístup k väčšine funkcií aplikácie.
7. Osobám, ktoré aplikáciu používali, sa páčilo triedenie užívateľov a udalostí do troch špecifických skupín v pravom vysúvacom menu.
8. Respondentom sa páčilo, že pri zrušení udalosti prišlo každému uchádzačovi upozornenie o tejto zmene.

8.3 Možnosti vylepšenia a rozšírnenia aplikácie

Medzi možnosti vylepšenia je možné zaradiť všetky negatívne poznatky z kapitoly 8. Príkladom je prispôsobenie farebnej tématiky užívateľom a zlepšenie spracovania karty chatu. Ďalej by bolo možné rozšíriť aplikáciu do viacerých ľudských jazykov. Aplikácia je na toto rozšírenie predpripravená, a to tak, že všetky reťazce, ktoré sa vypisujú na užívateľskom rozhraní, sú uložené v osobitnom súbore, na ktorý kódová časť aplikácie referuje. To znamená, že zmenou jedného súboru alebo pridaním ďalších podobných súborov umožní jednoduché

rozšírenie aplikácie do viacerých jazykov. Taktiež by mohla byť aplikácia implementovaná pre iné mobilné operačné systém ako Android a to najmä pre operačný systém iOS, čím by sa viditeľne rozšírila potencionálna základňa užívateľov.

Kapitola 9

Záver

Cieľom tejto práce bolo vytvorenie mobilnej aplikácie pre platformu Android, ktorá slúži na vytváranie športových udalostí, vyhľadávanie vhodných užívateľov v okolí, pozývanie iných užívateľov na udalosť, potvrdzovanie účasti užívateľa, vzájomnú komunikáciu účastníkov alebo iných osôb a konečné zhodnotenie udalosti a jednotlivých užívateľov. Pre správne fungovanie mobilnej aplikácie bol však navrhnutý celý systém, ktorý pozostáva zo samotnej mobilnej aplikácie, relačnej databázy slúžicej na uchovávanie dát, a aplikačného servera.

Pre mobilnú aplikáciu bolo navrhnuté užívateľské rozhranie, ktoré umožňuje interakciu užívateľa a sprístupňuje všetky služby aplikácie. Taktiež bolo nutné implementovať backend, ktorý sa stará o komunikáciu s aplikačným serverom a spracovanie prijatých dát.

Aplikačný server zabezpečuje pomocou komunikácie s relačnou databázou sprístupnenie dát prisluchajúcim danému užívateľovi. Server vyhľadáva vhodných užívateľov v okolí obsluhovaného užívateľa, vhodné udalosti, ktoré sa taktiež konajú v jeho blízkosti, udalosti, ktorých sa užívateľ účastni alebo ktoré vytvoril. Okrem toho, server posiela a spracováva všetky potrebné informácie, ktoré sú nevyhnutné pre zabezpečenie služieb mobilnej aplikácie, akými sú posielanie notifikácií o zmenách jednotlivých objektov a prijímanie správ od iných užívateľov. Veľmi dôležité je podotknúť, že aplikačný server spracováva informácie tak, aby boli na stranu klienta doručené iba potrebné informácie, čím minimalizuje spotrebu internetových dát na strane mobilného klienta.

Za účelom komunikácie aplikačného serveru a mobilného klienta bol navrhnutý aplikačný protokol, ktorý pokrýva všetky požiadavky na funkcionalitu aplikácie a pri správnej implementácii redukuje prenos dát na minimum.

Nad rámec zadania bolo implementované zabezpečenie komunikácie medzi klientom a serverom pomocou SSL/TLSv1.2 s algoritmom RSA pre autentifikáciu.

Celý vývoj systému bol rozdelený do niekoľkých častí. Na úplnom začiatku sa vykonala analýza požiadavok pre daný systém, kde bola vymedzená cieľová skupina užívateľov, pre ktorých je aplikácia smerovaná, a požiadavky na funkcionalitu systému. Následne bol vytvorený návrh systému vrátane užívateľského rozhrania aplikácie. Za touto etapou nasledovala samotná implementácia systému a jeho testovanie. Pomocou testovania sa nadobudli poznatky o výhodách a nevýhodách aplikácie, ktoré môžu slúžiť pre rozšírenia do budúcnosti.

Literatúra

[1] AMAL, R. W.: *Learning Android Google Maps*. Packt Publishing, 2015, ISBN 9781849698863.

[2] BURNETTE, E.: *Hello, Android: Introducing Google's Mobile Development Platform*. Pragmatic Bookshelf, Štvrté vydanie, Máj 2015, ISBN 978-1680500370.

[3] CALVERT, K. L.; DONAHOO, M. J.: *TCP/IP sockets in Java: practical guide for programmers*. Morgan Kaufmann Publishers, druhé vydanie, 2002, ISBN 978-1558606852.

[4] CHURCHER, C.: *Beginning database design*. Springer-Verlag New York, 2007, ISBN 978-1590597699.

[5] CLYNCH, J. R.: Earth Coordinates. School of Civil and Environmental Engineering (CVEN) at UNSW AUSTRALIA, Február 2006, [Online; navštívené 8.5.2018].
URL https://web.archive.org/web/20150418092513/http://www.sage.unsw.edu.au/snap/gps/clynch_pdfs/coorddef.pdf

[6] COMER, D. E.: *Internetworking with TCP/IP Volume One*. Pearson, 6 vydanie, 2013, ISBN 978-0-13-608530-0.

[7] Defense Mapping Agency: Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems. DMA, Fairfax, Virginia, USA, September 1991, [Online; navštívené 8.5.2018].
URL http://www.dtic.mil/dtic/tr/fulltext/u2/a280358.pdf

[8] DESCARTES, R.: *Discourse on Method, Optics, Geometry, and Meteorology*. Hackett Publishing, 2001, ISBN 0-87220-567-3.

[9] ELMASRI, R.; NAVATHE, S.: *Fundamentals of Database Systems*. Pearson, 7 vydanie, 2015, ISBN 9780133970777.

[10] Google Inc.; Open Handset Alliance n.d.: Android Debug Bridge (adb). [Online; navštívené 8.5.2018].
URL https://developer.android.com/studio/command-line/adb

[11] Google Inc.; Open Handset Alliance n.d.: Android Lollipop. [Online; navštívené 8.5.2017].
URL https://developer.android.com/about/versions/lollipop

[12] Google Inc.; Open Handset Alliance n.d.: Android Studio Release Notes. [Online; navštívené 8.5.2018].
URL https://developer.android.com/studio/releases/

[13] Google Inc.; Open Handset Alliance n.d.: Distribution Dashboards. [Online; navštívené 22.12.2017].
URL https://developer.android.com/about/dashboards/

[14] Google Inc.; Open Handset Alliance n.d.: Emulator Release Notes. [Online; navštívené 8.5.2018].
URL https://developer.android.com/studio/run/emulator

[15] Google Inc.; Open Handset Alliance n.d.: Introduction to Activities. [Online; navštívené 8.5.2018].
URL https://developer.android.com/guide/components/activities/intro-activities

[16] Google Inc.; Open Handset Alliance n.d.: Platform Architecture. [Online; navštívené 8.5.2018].
URL https://developer.android.com/guide/platform/

[17] Google Inc.; Open Handset Alliance n.d.: Processes and threads overview. [Online; navštívené 8.5.2018].
URL https://developer.android.com/guide/components/processes-and-threads

[18] Google Inc.; Open Handset Alliance n.d.: SDK Platform Tools Release Notes. [Online; navštívené 8.5.2018].
URL https://developer.android.com/studio/releases/platform-tools

[19] Google Inc.; Open Handset Alliance n.d.: SDK Tools Release Notes. [Online; navštívené 8.5.2018].
URL https://developer.android.com/studio/releases/sdk-tools

[20] Google Inc.; Open Handset Alliance n.d.: Systrace. [Online; navštívené 8.5.2018].
URL https://developer.android.com/studio/command-line/systrace

[21] Google Inc.; Open Handset Alliance n.d.: Understand the Activity Lifecycle. [Online; navštívené 8.5.2018].
URL https://developer.android.com/guide/components/activities/activity-lifecycle

[22] Google Inc.; Open Handset Alliance n.d.: View. [Online; navštívené 8.5.2018].
URL https://developer.android.com/reference/android/view/View

[23] Google Inc.; Open Handset Alliance n.d.: Zipalign. [Online; navštívené 8.5.2018].
URL https://developer.android.com/studio/command-line/zipalign

[24] International Civil Aviation Organization: World Geodetic System–1984 (WGS-84) Manual. ICAO, Montreal, Quebec, Canada, 2002, [Online; navštívené 8.5.2018].
URL https://gis.icao.int/eganp/webpdf/REF08-Doc9674.pdf

[25] MALING, D. H.: *Coordinate Systems and Map Projections*. Pergamon, Január 1992, ISBN 9780080372334.

[26] OSTRANDER, J.: *Android UI Fundamentals Develop and Design*. Peachpit Press, 2012, ISBN 0-321-81458-4.

[27] RAE, A.; POPESCU, A.; BOUCHARD, H.; aj.: Mining the web for points of interest. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '12*, NewYork, NewYork, USA: ACM Press, 2012, ISBN 9781450314725, s. 711–720, doi:10.1145/2348283.2348379, Dostupné z: http://dl.acm.org/citation.cfm?doid=2348283.2348379.

[28] Statista: Global mobile OS market share in sales to end users. [Online; navštívené 8.5.2017].
URL https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[29] SURHONE, L. M.; TENNOE, M. T.; Henssonow, S. F.: *Spherical Law of Cosines*. Betascript Publishing, August 2010, ISBN 978-613-1-19139-8.

[30] World Health Organization: Obesity and overweight. [Online; navštívené 8.5.2017].
URL http://www.who.int/mediacentre/factsheets/fs311/en

[31] YE, M.; YIN, P.; LEE, W.-C.; aj.: Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, NewYork, NewYork, USA: ACM Press, 2011, ISBN 9781450307574, s. 325–334, doi:10.1145/2009916.2009962, Dostupné z: http://portal.acm.org/citation.cfm?doid=2009916.2009962.

[32] YUAN, Q.; CONG, G.; MA, Z.; aj.: Time-aware point-of-interest recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '13*, NewYork, NewYork, USA: ACM Press, 2013, ISBN 9781450320344, s. 363–372, doi:10.1145/2484028.2484030, Dostupné z: http://dl.acm.org/citation.cfm?doid=2484028.2484030.

Príloha A

DVD

Priložené DVD obsahuje zdrojové súbory tohto textu a systému, ktorý bol implementovaný pre účel tejto práce.

Príloha B

Testovací dotazník

Tento dotazník bol využitý ako súčasť testovania aplikácie.

Testovací dotazník

Odpoveď označte zakrúžkovaním

Časť prvá – funkčnosť a funkcionalita

1. Pri používaní aplikácia javila znaky neočakavaného správania.

Ani raz 1-krát 2 až 3-krát 3 až 5-krát 6-krát a viac

2. Počas používania aplikácie nastalo zlyhanie služby. (Príznaky: zobrazila sa chybová hláška alebo po vykonaní zmeny sa reálne zmeny neuskutočnili.)

Ani raz 1-krát 2 až 3-krát 3 až 5-krát 6-krát a viac

3. Aplikácia sa počas doby testovania nečakane ukončila.

Ani raz 1-krát 2 až 3-krát 3 až 5-krát 6-krát a viac

4. Aplikáciu bolo nutné reštartovať z dôvodu zlyhania sieťového pripojenia.

Ani raz 1-krát 2 až 3-krát 3 až 5-krát 6-krát a viac

Časť druhá – užívateľské rozhranie

1. Dochádzalo k situáciám, kedy ste ako užívateľ nevedeli nájsť požadovanú funkcionalitu? (Napríklad ako pozvať iného užívateľa na udalosť)

Vôbec Výnimočne Občas Veľmi často Takmer stále

2. Dochádzalo k situáciám, kedy sa po kliknutí na ikonu vykonala nečakaná akcia? (Napríklad kliknutím na tlačidlo, ktoré ste si mysleli, že otvorí kartu užívateľa, vykonalo akciu, ktorú by ste kliknutim na dané tlačidlo neočakávali.)

Vôbec Výnimočne Občas Veľmi často Takmer stále

3. Známkou ohodnoťte rozloženie grafických elementov obrazovky, ktorá sa Vám zobrazí hneď po úspešnom prihlásení.

Vynikajúce Veľmi dobré Dobré Dostatočné Nedostatočné

4. Známkou ohodnoťte rozloženie grafických elementov obrazovky, ktorá znázorňuje profil udalosti alebo užívateľa.

Vynikajúce Veľmi dobré Dobré Dostatočné Nedostatočné

5. Známkou ohodnoťte rozloženie grafických elementov pravého vysúvacieho menu aplikácie.

Vynikajúce Veľmi dobré Dobré Dostatočné Nedostatočné

6. Nastávali situácie, kedy bolo užívateľské rozhranie nepoužiteľné?
(Napríklad jednotlivé prvky nereagovali na gestá alebo mali vysokú odozvu, pohyb kamery na mape sa sekal... .)

Vôbec Výnimočne Občas Veľmi často Takmer stále

Časť tretia

Do nasledujúcej časti doplňte poznámky (výhody, nedostatky, nápady na vylepšenie), ktoré neboli pokryté otázkami v predchadzajúcich častiach dotazníka.