

ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

ANDROID APLIKÁCIA NA SLEDOVANIE POLOHY
MOBILNÉHO ZARIADENIA

ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

**ANDROID APLIKÁCIA NA SLEDOVANIE POLOHY
MOBILNÉHO ZARIADENIA**

Kód: 28360220161474

BAKALÁRSKA PRÁCA

Študijný program:

Informatika – stará akreditácia

Katedra:

Katedra informatiky

Vedúci bakalárskej práce:

Ing. Michal Varga, PhD.

Žilina 2016

Andrej Šišila

Zadanie záverečnej práce

Tento list nahradiť za zadanie bakalárskej práce.

POĎAKOVANIE

Chcel by som sa poďakovať Ing. Michalovi Vargovi, PhD. za jeho trpezlivosť a podporu, doc. Ing. Norbertovi Adamkovi, PhD. za aktívne usmernenie počas celého vývoja práce a Bc. Jaroslavovi Janigovi za spoluprácu.

ABSTRAKT

Šišila, Andrej: *Android aplikácia na sledovanie polohy mobilného zariadenia*. [Bakalárska práca]. Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra informatiky. Vedúci: Ing. Michal Varga, PhD. Stupeň odbornej kvalifikácie: Bakalár, Študijný program: Informatika - stará akreditácia. Žilina: FRI ŽU, 2016. 49 s.

V našej bakalárskej práci sa zaoberáme vytvorením Android aplikácie na zisťovanie polohy rôznymi technológiami. Budeme opisovať jednotlivé technológie, ktoré je možné použiť na sledovanie mobilných zariadení a rozhodneme sa pre jednu, ktorá vyhovuje našim požiadavkám. Ďalej sa zaoberáme návrhom Android aplikácie, v ktorej hovoríme o komunikácii aplikácie s centrálnym úložiskom, jej objektovom návrhu a grafickom používateľskom rozhraní. Nakoniec budeme do našej aplikácie implementovať sledovanie prostredníctvom GPS podľa predchádzajúcej časti a overíme správnosť nami navrhnutého riešenia.

Kľúčové slová: Android, Bluetooth, GPS, mobilná aplikácia, wifi

ABSTRACT

Šišila, Andrej: *Android application to track position of mobile device*. [Bachelor Thesis]. University of Žilina. Faculty of Management Science and Informatics; Department of Informatics. Advisor: Ing. Michal Varga. Qualification degree: Bachelor. Study programme: Informatics. Žilina: FRI ŽU, 2016. 49 s.

In our bachelor thesis we are concerned with development of an Android application to track position of a mobile device via various technologies. We will be describing individual technologies, that can be used to track mobile devices and we will decide for one, that will fulfill our requirements. Next we will proceed to the design of our Android application, in which we are talking about communication between the application and the central storage, its object design and graphical user interface. In the end we will implement tracking via GPS into our application according to the previous part and we will verify accuracy of designed solution.

Keywords: Android, Bluetooth, GPS, mobile application, wifi

PREHLÁSENIE

Prehlasujem, že som túto prácu napísal samostatne a že som uviedol všetky použité pramene a literatúru, z ktorých som čerpal.

V Žiline, dňa **10.5.2016**

Andrej Šišila

OBSAH

Obsah	7
Zoznam ilustrácií	8
Zoznam tabuliek	9
Zoznam ukážok zdrojových kódov	9
Úvod.....	10
1. Ciele práce	11
2. Sledovacie technológie	12
2.1. GPS	12
2.2. Wi-Fi.....	13
2.3. Bluetooth.....	14
2.4. Porovnanie technológií	16
3. Návrh Android aplikácie.....	18
3.1. Komunikácia so serverom.....	20
3.1.1. Výber typu servera	21
3.1.2. Formát správ.....	24
3.1.3. Typy správ.....	26
3.1.3.1. Správy typu POST.....	26
3.1.3.2. Správy typu GET.....	28
3.2. Integrácia novej sledovacej technológie	28
3.2.1. Vytvorenie sledovacej triedy.....	28
3.2.2. Integrácia do služby	29
3.2.3. Integrácia do grafického rozhrania.....	31
3.2.4. Integrácia na server	33
4. Vytvorenie sledovacej triedy pre zbieranie dát pomocou GPS	36
4.1. Integrácia do služby	36
4.2. Vytvorenie triedy pre GPS sledovanie.....	37
4.3. Integrácia do grafického rozhrania	39
4.4. Integrácia na server	40
5. Overenie.....	42
6. Záver	45
7. Zoznam použitej literatúry.....	46

ZOZNAM ILUSTRÁCIÍ

Obrázok 1: Usporiadanie GPS satelitov okolo Zeme [7]	13
Obrázok 2: Rôzne veľkosti Bluetooth Beacon zariadení [23]	16
Obrázok 3: UML diagram aplikácie „PedTrack“	19
Obrázok 4: Znázornenie komunikácie medzi triedami	20
Obrázok 5: Priebeh "Three Way Handshake" procesu [26]	22
Obrázok 6: Entitno-relačný diagram databázy	24
Obrázok 7: Ukážka životného cyklu služby „Service“ (vľavo) a ukážka životného cyklu služby „Bound Service“ (vpravo) [31]	31
Obrázok 8: Sled metód životného cyklu aktivity[31]	33
Obrázok 9: Znázornenie komunikácie medzi Android aplikáciou a serverom	35
Obrázok 10: Úprava nastavení polohy v Android 4.4.4: a) Zapnutie polohy, b) Možnosti zisťovania polohy, c) Režim určovania polohy	37
Obrázok 11: Oznamovanie GPS polohy	38
Obrázok 12: Priebeh zapínania a vypínania GPS sledovania cez grafické rozhranie.....	40
Obrázok 13: Znázornenie naplňovania a zapuzdrovania tried.....	41
Obrázok 14: Databáza pred meraním	42
Obrázok 15: Databáza po meraní.....	43
Obrázok 16: Overenie súradníc pomocou Google Maps v zázname s hodnotou „GPS_DATA_ID“ 656	43
Obrázok 17: Záznam v databáze pod "GPS_DATA_ID" 556.....	44
Obrázok 18: Overenie súradníc pomocou Google Maps v zázname s hodnotou „GPS_DATA_ID“ 556	44

ZOZNAM TABULIEK

Tabuľka 1: Bluetooth triedy[20]	15
Tabuľka 2: Bluetooth verzie a im prislúchajúce maximálne rýchlosti[21]	15
Tabuľka 3: Porovnanie jednotlivých technológií	17

ZOZNAM UKÁŽOK ZDROJOVÝCH KÓDOV

Ukážka zdrojového kódu 1: Príklady jednotlivých formátov správ	25
Ukážka zdrojového kódu 2: Inicializácia zaškrtáčacieho políčka pre ovládanie GPS sledovania	39

ÚVOD

V dnešnej dobe je počítačová simulácia na vysokej úrovni. Na to, aby simulačný model dával správne výsledky, je potrebné získať validné vstupné dáta. Keď zároveň zoberieme do úvahy, že väčšina inteligentných mobilných zariadení je v súčasnosti založených na platforme Android, vytvorením takejto aplikácie máme možnosť získavať od používateľov veľké množstvo dát o pozíciách jednotlivých zariadení, ktoré môžeme neskôr spracovať. Tieto informácie ďalej použijeme v simulačnom nástroji pohybu a správania sa chodcov, ktorý je vyvíjaný na Fakulte riadenia a informatiky Žilinskej univerzity v Žiline.

Našou úlohou je vytvoriť Android aplikáciu, ktorá bude vedieť získať polohu mobilného zariadenia a zhromaždené dáta odoslať do centrálného úložiska. Pri jej vytváraní musíme dbať na používateľovu bezpečnosť, anonymitu a presnosť odosielaných dát. Odoslané dáta následne použijeme na simulačné a štatistické účely.

Takáto aplikácia dokáže uľahčiť zber dát o polohách jednotlivých zariadení, pričom dokáže zaručiť, že takto zozbierané dáta sú validné, pretože pochádzajú z reálneho sveta, čiže nie sú vymyslené.

Túto tému bakalárskej práce sme si vybrali hlavne kvôli tomu, že sme mohli prepojiť naše vedomosti z odboru sietí ako aj z programovania. Na jednej strane sme použili vedomosti zo sietí pri riešení problémov v komunikácii medzi klientom a serverom a pri porovnávaní jednotlivých technológií na zisťovanie polohy, na druhej strane, naše poznatky z programovania sme aplikovali pri implementácii jednotlivých častí Android aplikácie.

1. CIELE PRÁCE

Hlavným cieľom našej práce je vytvorenie Android aplikácie, ktorá bude slúžiť na získavanie informácií o polohe mobilného zariadenia a bude odosielať zozbierané informácie do centrálného úložiska. Vytvorenie takejto aplikácie si vyžadovalo splnenie nasledovných čiastkových cieľov:

- Najprv sme museli preskúmať technológie, pomocou ktorých je možné určiť polohu mobilného zariadenia, čo zahŕňalo preskúmať fungovania GPS, Wi-Fi a Bluetooth.
- Ďalej sme potrebovali navrhnúť a implementovať Android aplikáciu na získanie polohy mobilného zariadenia na základe dát o polohe získaných z predtým zvolenej technológie. Pre to sme si najprv museli zvoliť technológiu, z ktorej budú získavané dáta o polohe a následne zistiť spôsoby implementácie danej technológie na platforme Android.
- Potom bolo nutné vytvoriť mechanizmus, ktorý bude odosielať zozbierané anonymné dáta do centrálného úložiska, pričom sa museli zvoliť komunikačný protokol medzi mobilným zariadením (klientom) a serverom, vhodný formát odosielaných a prijímaných dát medzi klientom a serverom a implementovať odosielanie a príjem dát zo servera tak, aby aplikácia fungovala korektne.
- Nakoniec sme overovali funkčnosť vytvoreného riešenia tým, že sme nainštalovali aplikáciu na rôzne Android zariadenia. Kontrolovali sme, či sa dáta z Android aplikácie odosielajú do centrálného úložiska a či sú tieto dáta validné.

2. SLEDOVACIE TECHNOLOGIE

Mobilné zariadenie sa dá sledovať použitím rôznych technológií, ako napr. GPS, Wi-Fi a Bluetooth. Jednotlivé technológie sú opísané nižšie spolu s ich špecifickými vlastnosťami.

2.1. GPS

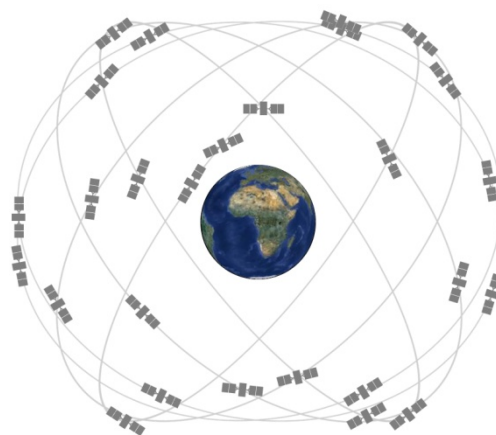
GPS (Global Positioning System) je vesmírny navigačný systém, ktorý poskytuje súradnice a časové informácie pri akomkoľvek počasi, kdekoľvek na alebo blízko Zeme [1]. GPS technológia je určená predovšetkým na sledovanie v teréne, ideálne, kde je čo najväčší výhľad na oblohu.

Dnešné GPS prijímače poskytujú horizontálnu presnosť lepšiu ako 3,5 metra, hoci viacero faktorov, ako napr. kvalita prijímača a atmosférické problémy môžu ovplyvniť presnosť. Bežné GPS prijímače sú presné v priemere do 15 metrov [2].

GPS prijímač musí byť zameraný aspoň štyrmi satelitmi, aby vypočítal súradnice v trojrozmernom priestore. Ak zameranie trvá prijímaču príliš dlho, môže pomôcť technológia A-GPS, ktorá dokáže urýchliť tento proces [3].

Poloha GPS prijímača sa vypočíta použitím trilaterácie. Princíp trilaterácie spočíva v tom, že GPS prijímač porovná čas, v ktorom bol signál vysielaný zo satelitu s časom, v ktorom bol prijatý. Časový rozdiel hovorí, ako ďaleko sa satelit nachádza. Ak sa takto zmerajú vzdialenosti od niekoľkých ďalších satelitov, prijímač dokáže zistiť používateľovu polohu [4].

Obvykle sa vo vesmíre nachádza viac ako 24 GPS satelitov pre zaistenie pokrytia, akonáhle je potrebné nejaký zo satelitov opraviť alebo vyradiť z prevádzky. Väčšie množstvo satelitov môže zvýšiť výkonnosť GPS, ale nepočíta sa s nimi v základnej konfigurácii [5]. Satelity sú organizované do šiestich rovnako od seba vzdialených rovín okolo Zeme, pričom na každej rovine sa nachádzajú štyri satelity. Táto 24-miestna konfigurácia zaisťuje používateľovi, že uvidí aspoň 4 satelity z každého miesta na Zemi (pozri Obrázok 1) [6].



Obrázok 1: Usporiadanie GPS satelitov okolo Zeme [7]

GPS systém je teda spoľahlivý nástroj na určenie polohy zariadenia v teréne, horšie je na tom s lokalizáciou vnútri budov, pretože GPS signál nie je taký silný na to, aby prešiel väčšinou pevných objektov. Prevažná časť dnešných mobilných telefónov už má v sebe integrovanú GPS anténu. Nevýhodou je, že akonáhle ju zapneme, batéria telefónu sa vplyvom zvýšených energetických nárokov začne vybíjať rýchlejšie.

2.2. Wi-Fi

Wi-Fi je technológia založená na štandarde 802.11, vyvinutý inštitútom IEEE, ktorá umožňuje elektronickým zariadeniam pripojiť sa na bezdrôtovú lokálnu sieť (WLAN) [8].

Zariadenia podporujúce Wi-Fi sa môžu pripojiť na internet prostredníctvom bezdrôtového prístupového bodu, zvaného hotspot. Väčšina Wi-Fi prístupových bodov vysiela na frekvenciách 2,4 (802.11b/g/n) alebo 5 GHz (802.11 a/n/ac) [9]. Každý prístupový bod odosiela v pravidelných intervaloch všetkým zariadeniam v dosahu názov bezdrôtovej siete, tzv. SSID (ak nie je skrytá), kanál, na ktorom vysiela a druh použitého zabezpečenia (otvorená, WEP, WPA a pod.). Wi-Fi hotspot má dosah približne 20 metrov vnútri budov a v závislosti na sile a druhu použitej antény, hrúbke a štruktúre stien sa pokrytie signálom môže meniť. Vo vonkajších podmienkach sa môže dosah signálu zvýšiť až na 2 km, tým, že použijeme smerové Wi-Fi antény [10], čím sa zväčšuje dosah signálu na úkor plošného pokrytia.

Wi-Fi predstavuje zvýšené bezpečnostné riziko než káblové pripojenia [11], akým je napr. Ethernet, pretože sa narušiteľ nepotrebuje fyzicky pripojiť k sieti, ale dokáže do nej

vniknúť na diaľku. Existujú rôzne techniky zabezpečenia, ako napr. WEP, WPA či WPA2, ktoré sú podrobnejšie uvedené v [12][13].

Keďže väčšina Wi-Fi prístupových bodov pracuje na frekvencii 2,4 GHz a tiež mnohé mobilné telefóny podporujú výlučne 2,4 GHz Wi-Fi pásmo (ale podpora 5 GHz rastie, pozri [14]), predstavuje to istým spôsobom nevýhodu, pretože pokiaľ viacero prístupových bodov vysiela na kanáloch s prekrývajúcimi sa frekvenciami, jednotlivé signály z rôznych prístupových bodov sa navzájom rušia a celková kvalita prijímu sa znižuje, keďže v rámci tohto frekvenčného pásma existujú iba tri kanály, ktoré sa navzájom neprekrývajú: 1, 6 a 11. Navyše aj iné technológie a elektrospotrebiče využívajú 2,4 GHz pásmo, ako napr. Bluetooth alebo mikrovlnné rúry, čo môže ďalej znížiť kvalitu prijímaného signálu [15]. Riešením by bolo použitie 5 GHz pásma, ktoré je oveľa menej zarušené [16].

Pokiaľ ide o zisťovanie pozície vnútri budov, Wi-Fi technológia je vhodným kandidátom na jej realizáciu, pretože je pomerne ľahko dostupná a jednoducho rozširovateľná. Zisťovanie polohy prebieha tak, že meriame silu signálu od prístupového bodu. Musíme však zobrať do úvahy, že aj pri použití Wi-Fi ako spôsobu pre interiérové zisťovanie polohy, existujú nepresnosti, ktoré sú spôsobené hlavne stenami a vzájomným rušením jednotlivých prístupových bodov.

Zapnutie Wi-Fi v mobilnom telefóne, podobne ako pri GPS, má tiež za následok zvýšenú spotrebu energie, v dôsledku čoho sa batéria telefónu vybíja rýchlejšie.

2.3. Bluetooth

Bluetooth je štandard na bezdrôtovú výmenu dát a komunikáciu zariadení na krátku vzdialenosť. Na prenos signálu sa používa 2,4 GHz pásmo, podobne ako pri Wi-Fi [17].

Na rozdiel od Wi-Fi, Bluetooth má omnoho kratší dosah signálu a nižšiu rýchlosť prenosu dát. Obvykle na to, aby dve zariadenia medzi sebou komunikovali, musia byť v dosahu signálu a musia byť spárované [18].

Bluetooth technológia sa rozlišuje nielen podľa verzie, ale aj podľa tried. Zatiaľ čo verzia určuje prenosovú rýchlosť, trieda udáva dosah prenášaného signálu. Vo väčšine aktuálnych mobilných zariadení je použitý modul vo verzii 3.0 alebo vyšší, ktorý patrí prevažne do triedy 2, ktorá má dosah približne 10 metrov [19].

Tabuľka 1:
Bluetooth triedy[20]

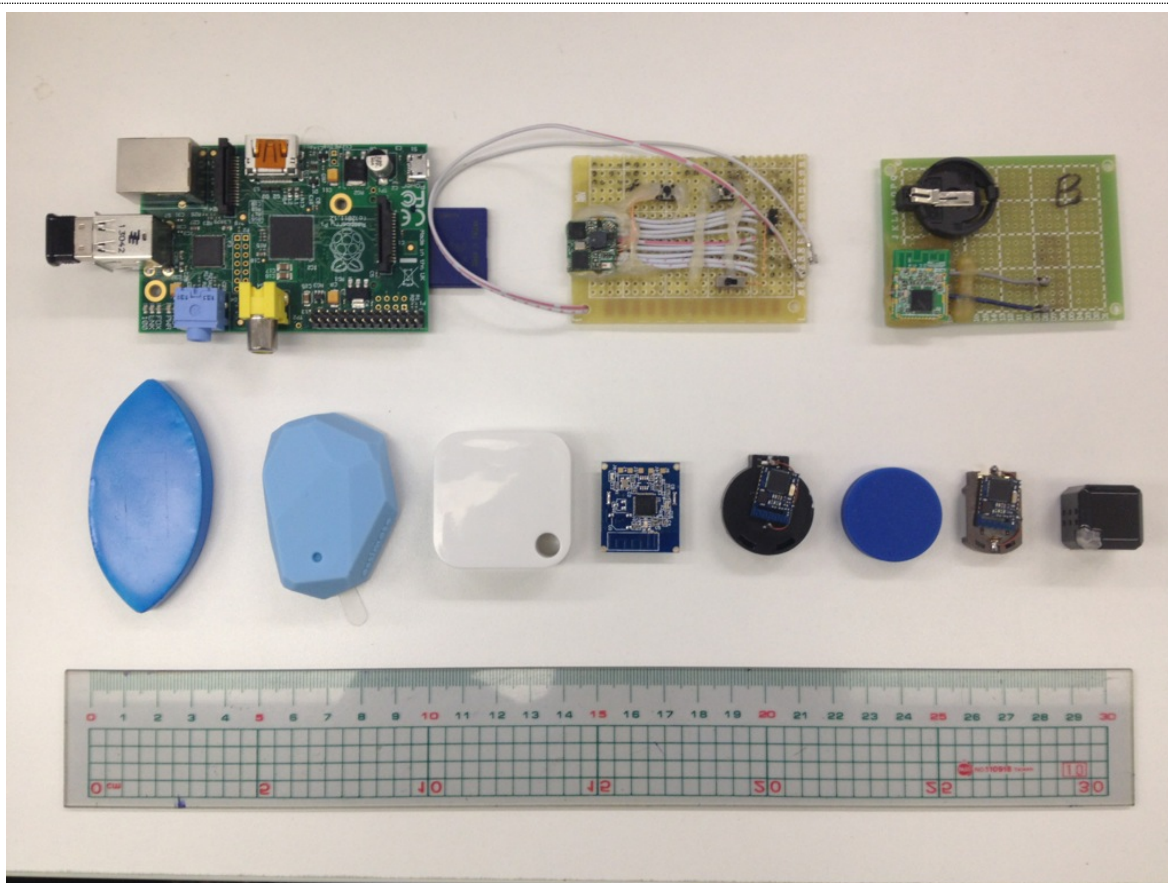
<i>Trieda</i>	<i>Max. sila vysielania</i>		<i>Približný dosah (m)</i>
	<i>(mW)</i>	<i>(dBm)</i>	
<i>1</i>	100	20	~100
<i>2</i>	2.5	4	~10
<i>3</i>	1	0	~1

Tabuľka 2:
Bluetooth verzie a im prislúchajúce maximálne rýchlosti[21]

<i>Verzia</i>	<i>Maximálna rýchlosť (Mbit/s)</i>
<i>1.2</i>	1
<i>2.0 + EDR</i>	3
<i>3.0 + HS</i>	24
<i>4.0</i>	24

Osobitnú kategóriu Bluetooth zariadení tvoria tzv. Bluetooth Beacon zariadenia. Bluetooth Beacon je špeciálne zariadenie, ktoré vysiela všetkým zariadeniam v dosahu správy ľubovoľného obsahu. Dosah jedného Bluetooth Beacon zariadenia na voľnom priestranstve je nanajvýš 50 m pre modely so zníženou spotrebou a až 100 m pre klasické modely [18].

Bluetooth Beacon zariadenia môžeme použiť nielen na prenos správ do mobilného zariadenia, ale aj na lokalizáciu iných zariadení v dosahu. Môžeme ich umiestniť kdekoľvek v budove vďaka ich malým rozmerom (pozri Obrázok 2) a sú lacnejšie v porovnaní s Wi-Fi prístupovým bodom. Lokalizácia zariadenia prebieha podobne ako pri Wi-Fi. Podrobnejší opis možností Bluetooth Beacon zariadení nájdeme v [22].



Obrázok 2: Rôzne veľkosti Bluetooth Beacon zariadení [23]

Podobne ako pri Wi-Fi a GPS technológiách, zapnutím Bluetooth na mobilnom zariadení má za následok zvýšenú energetickú spotrebu. Ďalšou nevýhodou je potreba vymieňať resp. dobíjať batérie v samotnom Bluetooth Beacon zariadení, pokiaľ ho nechceme mať napájaný adaptérom z elektrickej zásuvky.

2.4. Porovnanie technológií

Každá z týchto technológií má svoje výhody a nevýhody, ktoré teraz zhrnieme do prehľadnej tabuľky.

Z tabuľky vyplýva, že najpresnejšou technológiou na zisťovanie polohy vnútri budov je Bluetooth, zatiaľ čo najjednoduchším spôsobom, ako zistiť polohu vonku je GPS.

Keďže požadujeme, aby bolo zariadenie sledované v teréne, rozhodli sme sa do našej aplikácie implementovať sledovanie prostredníctvom GPS.

Tabuľka 3:
Porovnanie jednotlivých technológií

<i>Technológia</i>	<i>Použitie</i>	<i>Presnosť</i>	<i>Potrebné zariadenia</i>	<i>Súkromie / Ochrana osobných údajov</i>	<i>Náročnosť výpočtu</i>
<i>GPS</i>	vonku	5-15m	GPS prijímač	vysoká	nízka
<i>Wi-Fi</i>	vnútri	2-4m[24]	Wi-Fi príst. body	nižšia	vysoká
<i>Bluetooth</i>	vnútri	1-3m[25]	Bluetooth Beacon zariadenia	nižšia	vysoká

3. NÁVRH ANDROID APLIKÁCIE

Úlohou aplikácie je sledovať polohu zariadenia a odosielať ju na server. Aplikácia je kompatibilná s operačným systémom Android od verzie 4.0.x (API 14) až do verzie 5.1.1 (API 22). Vyvíjali sme ju pre v prostredí Android Studio, ktorý používa programovací jazyk Java.

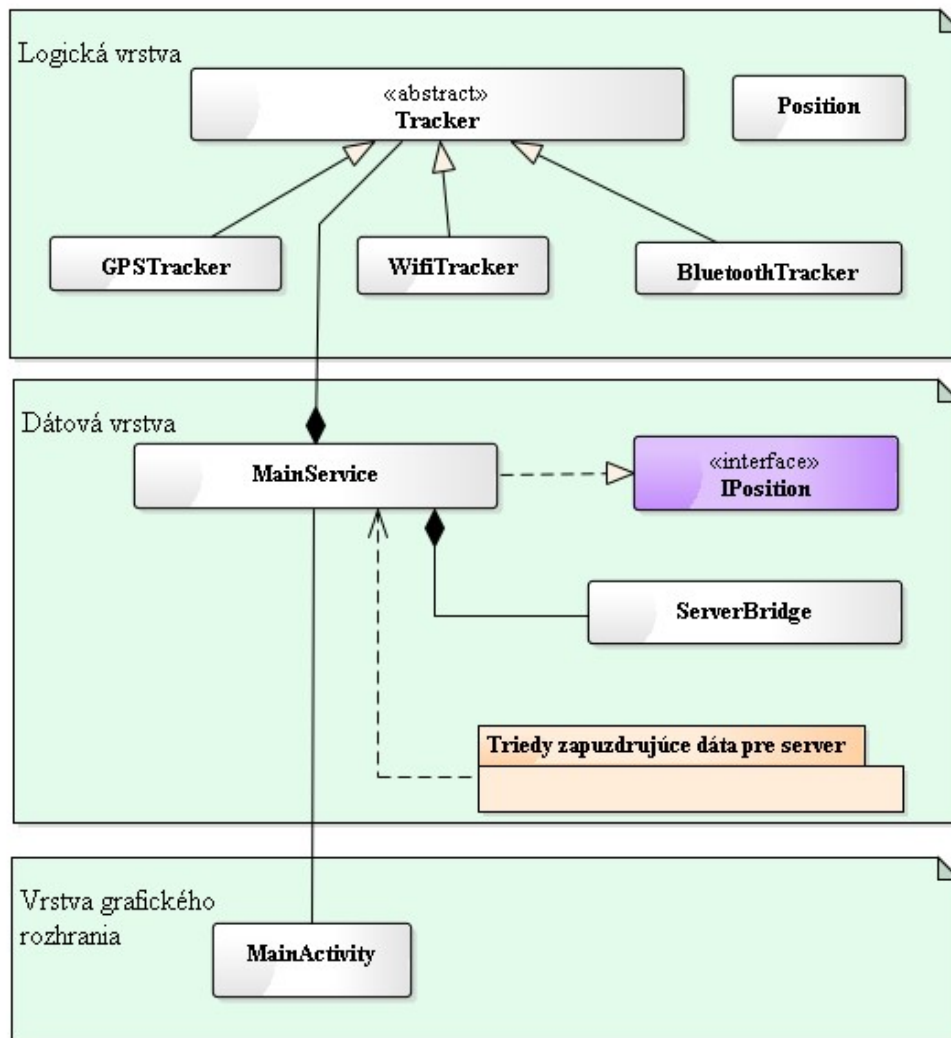
Aplikácia je navrhnutá tak, aby podporovala viacero spôsobov sledovania mobilného zariadenia napriek tomu, že sme sa rozhodli pre využitie GPS.

Aplikácia sa skladá z troch hlavných vrstiev: logickej, dátovej a grafickej. Do logickej vrstvy zahŕňame jednotlivé sledovacie triedy, ktoré zisťujú aktuálne súradnice o polohe zariadenia. Dátovú časť tvorí služba bežiaca na pozadí, trieda na odosielanie informácií na server a triedy, ktoré používa server na vykonávanie požiadaviek. Grafickú vrstvu tvorí trieda s grafickým rozhraním, ktorým používateľ riadi správanie aplikácie. Pre viac informácií prikladáme UML diagram aplikácie (pozri Obrázok 3).

Sledovacie triedy sú zodpovedné za sprostredkovanie informácií o polohe mobilného zariadenia. Tieto informácie zahŕňajú súradnice pozície, na ktorej sa nachádzame a identifikačné číslo sledovacej triedy, aby sme vedeli rozlíšiť, ktorá sledovacia trieda odoslala príslušné súradnice. Akonáhle sú súradnice o polohe zistené, posunú sa ďalej na spracovanie triede „MainService“. Trieda „MainService“ je služba bežiaca na pozadí, zodpovedná za spracovanie informácií odovzdané sledovacími triedami. Komunikácia medzi triedou „MainService“ a jednotlivými sledovacími triedami je zabezpečená rozhraním „IPosition“ a triedou „Tracker“, ktorá je abstraktným predkom všetkých sledovacích tried. Trieda „Tracker“ je zodpovedná za doručenie informácií o polohe rozhraniu „IPosition“. Pretože služba „MainService“ implementuje toto rozhranie, môže tieto informácie prijať. Sledovacie triedy nekomunikujú so službou „MainService“ priamo, pretože ak by tomu tak bolo, mohlo by sa stať, že jedna sledovacia trieda by mohla vypnúť sledovanie inej triedy a vo všeobecnosti by sme porušili princíp zapuzdrenia, čo je jeden zo základných konceptov objektovo orientovaného programovania. Týmto spôsobom sme izolovali jednotlivé sledovacie triedy a zaistili zvýšenú bezpečnosť v komunikácii medzi logickou a dátovou vrstvou.

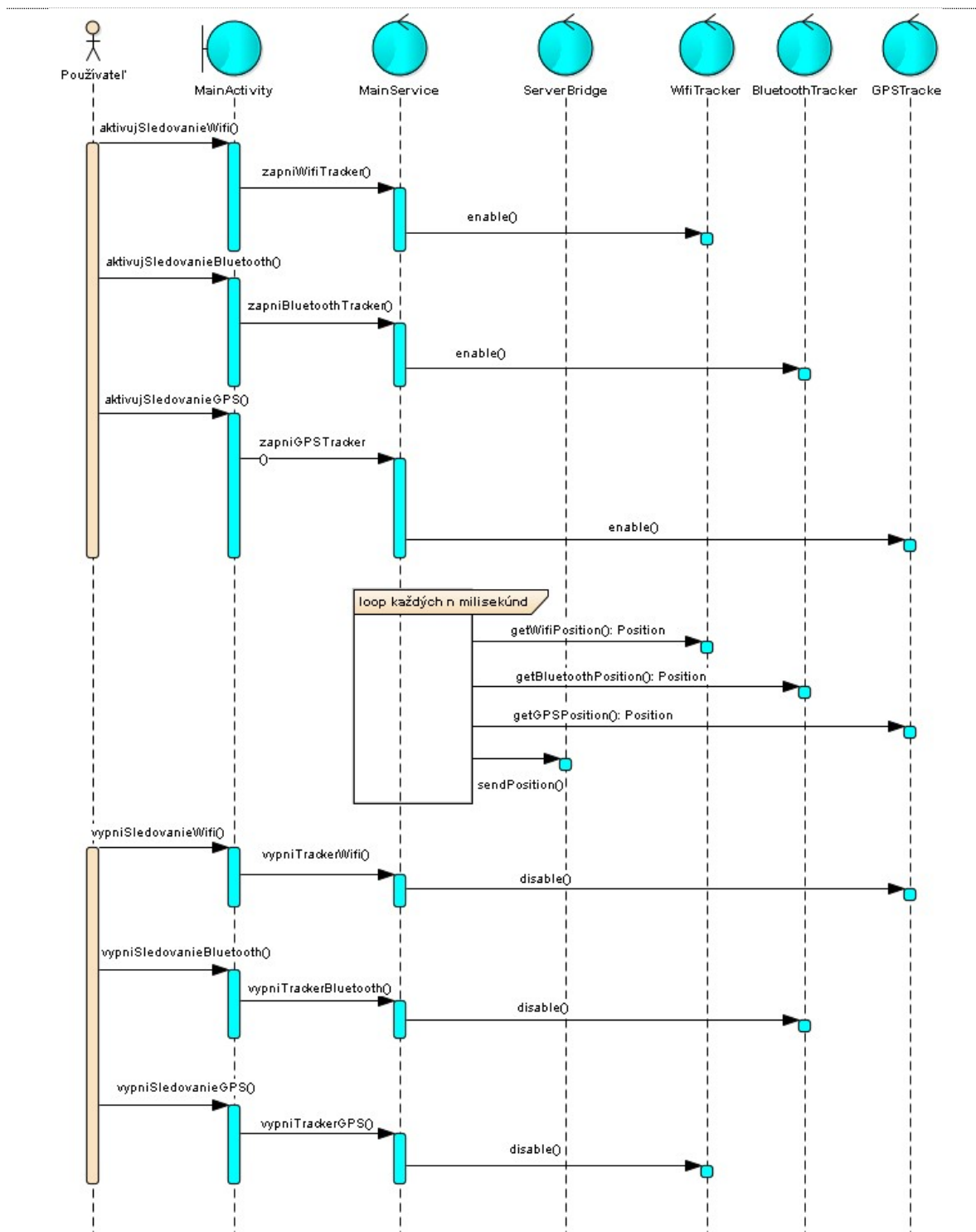
Trieda „MainService“ má všetky sledovacie triedy uložené v poli, ktorého prvky sú typu „Tracker“. Ku každej sledovacej triede potom pristupujeme na základe jej

identifikátora, čo je aj jej index v poli. Kvôli prehľadnosti sú identifikátory jednotlivých sledovacích tried uložené v triede „Tracker“ ako statické konštantné atribúty celočíselného typu.



Obrázok 3: UML diagram aplikácie „PedTrack“

Akonáhle služba „MainService“ prijme informáciu o polohe od danej sledovacej triedy, uloží si ju a odošle na server. O komunikáciu medzi Android aplikáciou a centrálnym úložiskom sa stará trieda „ServerBridge“. Prehľad komunikácie medzi triedami a ich životný cyklus môžeme vidieť na Obrázok 4. Trieda „MainService“ zbiera a odosiela polohy aktivovaných sledovacích tried v samostatnom vlákne v pravidelných intervaloch n milisekúnd. Je tomu tak preto, aby sa nezaťažovalo hlavné vlákno aplikácie, ktoré je zodpovedné za interakciu s užívateľom cez grafické rozhranie. Ďalej sa problematike komunikácie aplikácie so serverom budeme venovať v kapitole 3.1.



Obrázok 4: Znáznornenie komunikácie medzi triedami

3.1. Komunikácia so serverom

Aplikácia komunikuje so serverom, na ktorý odosiela polohy rôznych používateľov v danom čase. V nasledujúcich kapitolách opisujeme, aký typ servera používame, aké

správy posiela Android aplikácia na server, v akom sú formáte a aký komunikačný protokol používame na výmenu dát.

3.1.1. Výber typu servera

Najprv sme museli určiť, aký typ servera použijeme na ukladanie dát z Android aplikácie. Do úvahy pripadal buď súborový server, alebo databázový server.

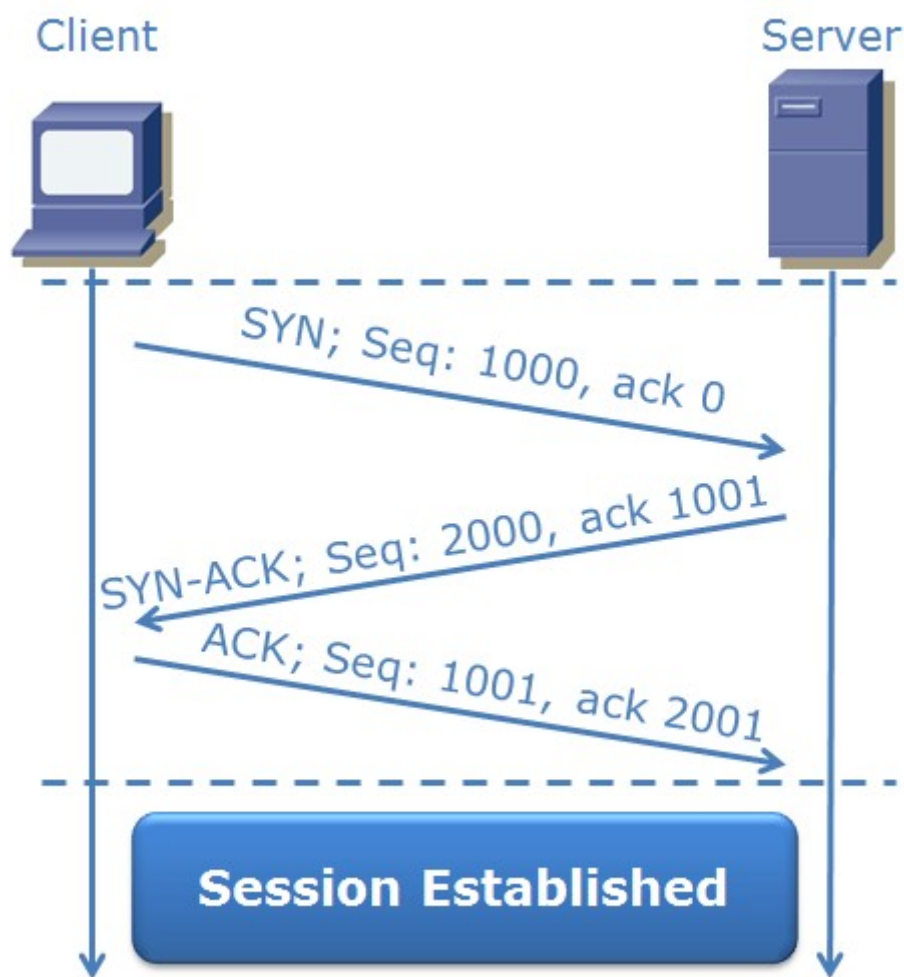
Súborový server je vhodný na zdieľanie súborov medzi viacerými používateľmi. Dáta na serveri sú uložené v adresároch a jednotlivým používateľom resp. skupinám používateľov vieme nastaviť prístupové práva ku adresárom alebo súborom.

Na prenos súborov môžeme použiť napr. FTP. FTP (File Transfer Protocol) je protokol aplikačnej vrstvy TCP/IP modelu resp. vrstvy 7 ISO/OSI modelu používaný na prenos súborov. Používa protokol TCP na transportnej vrstve (vrstve 4) pre inicializáciu, udržanie a ukončenie spojenia. Protokol TCP je spojovaný a spoľahlivý.

Spojovanosť znamená, že najprv medzi klientom a serverom musí vzniknúť spojenie. Na inicializáciu spojenia sa používa tzv. „Three Way Handshake“ proces, ako je znázornené na Obrázok 5. Najprv klient odošle serveru správu „SYN“ (Synchronize), ktorá hovorí, že sa chce so serverom spojiť. Server následne odpovie klientovi správou „ACK“ (Acknowledgement) a spätne odošle klientovi správu „SYN“. Klient prijíma tieto správy a potvrdí spojenie odoslaním správy „ACK“ serveru, čím je spojenie medzi klientom a serverom vytvorené [26].

Spoľahlivosť spočíva zase v tom, že po vytvorení spojenia sa prenesú všetky dáta (segmenty). Každý segment má svoje sekvenčné číslo, ktoré určuje, koľko dát sa prenieslo medzi klientom a serverom. Tie segmenty, ktoré sa nepreniesli, sa odosielajúca strana pokúša odoslať znova, až kým jej nepríde potvrdenie („ACK“ správa) od druhej strany, že prijala stratené segmenty [27].

Aby používateľ mohol pristupovať k súborom, autentifikuje sa svojim používateľským menom a heslom, ktoré sa na server odosielaajú v textovej podobe. Existujú aj zabezpečené varianty protokolu FTP napr. SFTP (SSH File Transfer Protocol), no najčastejšie je FTP spojenie šifrované pomocou TLS/SSL (Transport Layer Security/SecureSocketLayer), tiež známe ako FTPS.



Obrázok 5: Priebeh "Three Way Handshake" procesu [26]

Na druhej strane máme databázový server, ktorý je prispôbený na ukladanie veľkého množstva dát. V rámci rôznych druhov databázových systémov sme sa rozhodovali medzi NoSQL a SQL.

- **NoSQL (Not-only-SQL)** - druh databázového systému pozostávajúci z viacerých druhov špecializovaných databáz. Integrita dát je dynamická, takže každý záznam v databáze môže obsahovať rôzne množstvo informácií. Škálovateľnosť tohto druhu databázového systému je horizontálna, čo znamená, že stačí pridať ďalší server alebo internetovú inštanciu (obvykle sa jedná o virtuálny stroj), aby sme zvýšili kapacitu na ukladanie záznamov. Databáza následne automaticky rozmiestni dáta naprieč servermi podľa potreby [28].
- **SQL** - SQL je naproti tomu relačný typ databázového systému. Dáta sú uložené v tabuľkách, ktoré sú medzi sebou prepojené reláciami. Tie sú

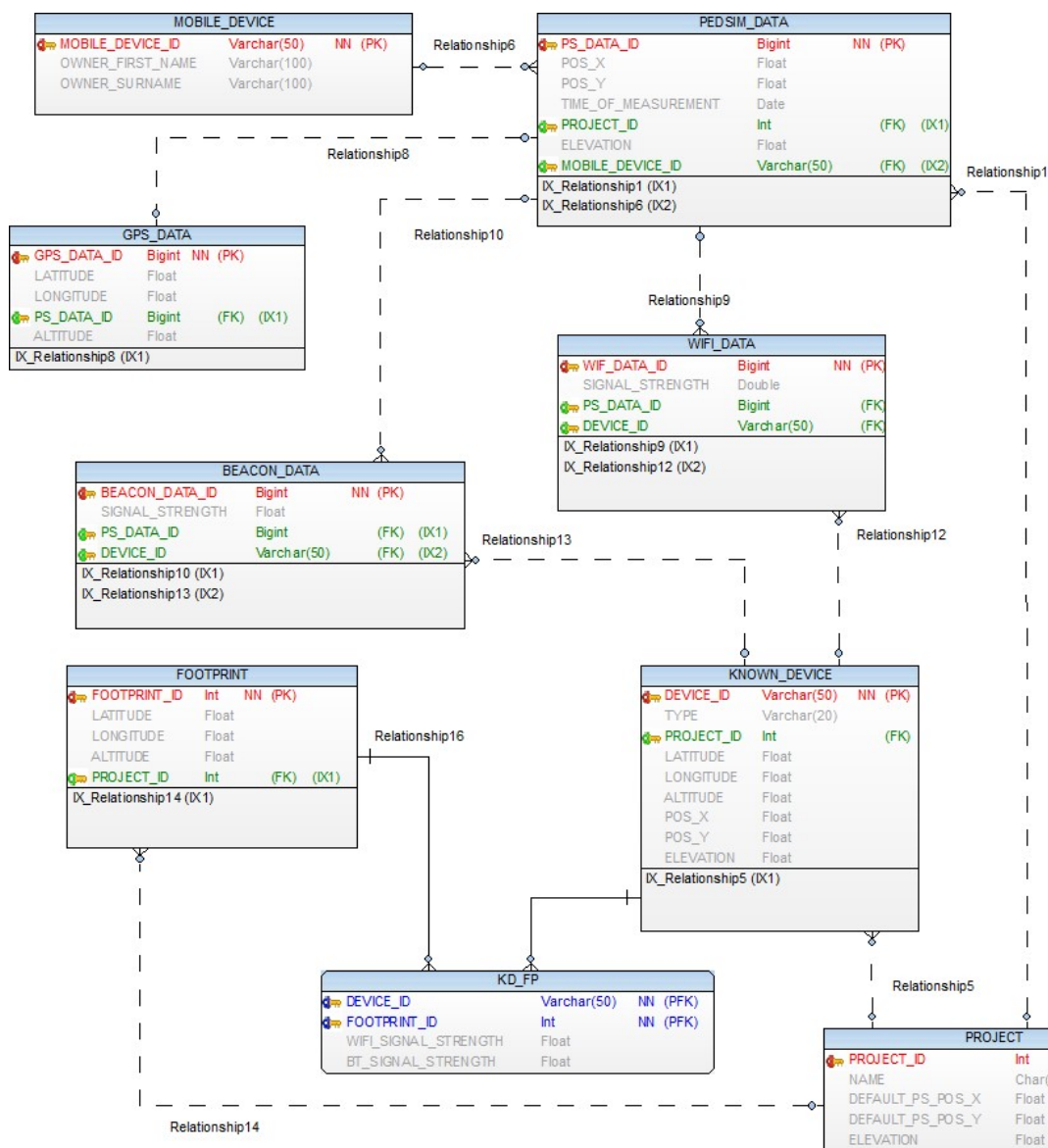
znázornené v entitno-relačnom diagrame. Integrita dát musí byť dodržiavaná a je prísne kontrolovaná, preto je nutné pri vkladaní nového záznamu (riadku) dodržiavať primárne kľúče, „NOT NULL“ atribúty a integritné obmedzenia pre danú tabuľku. Škálovateľnosť je ale oproti NoSQL horizontálna, čo znamená, že na zväčšenie kapacity je potrebný výkonnejší server. Databázu je možné rozšíriť na viaceré servery, ale nie je to triviálna operácia [28].

Od centrálného úložiska požadujeme, aby dokázal dáta rýchlo ukladať, vyhľadávať a triediť t. j. mal by v prípade potreby rýchlo vyhľadať používateľovu trasu podľa zadáných parametrov. Takéto nároky spĺňal databázový server s SQL databázovým systémom, ktorý je v našom prípade vhodnejšou voľbou. Konkrétne sme sa rozhodli používať databázový server MySQL[29].

Vytvorenie samotného servera nie je úlohou tejto práce. Pre názornosť však uvádzame entitno-relačný diagram na Obrázok 6.

V Android aplikácií nekomunikujeme s databázovým serverom priamo, ale prostredníctvom HTTP klienta a REST rozhrania. Podrobnosti komunikácii medzi Android aplikáciou a serverom bližšie vysvetlíme v kapitole 3.1.

Uvedomujeme si, že použitie protokolu HTTP na prenos údajov z Android aplikácie na server nie je optimálne z hľadiska používateľovej bezpečnosti. Vhodnejšou alternatívou na prenos dát medzi Android aplikáciou a serverom by bol protokol HTTPS, ktorý vie šifrovať komunikáciu použitím bezpečnostného certifikátu. Zabezpečenie spojenia medzi aplikáciou a serverom bude predmetom ďalšej práce na projekte.



Obrázok 6: Entitno-relačný diagram databázy

3.1.2. Formát správ

Ďalej sme sa museli dohodnúť na formáte, v ktorom budeme dáta na server posilať. Formát dát musel byť rovnaký aj v Android aplikácii, aj na serveri. Museli sme zohľadniť množstvo prenášaných dát a jednoduchosť serializácie resp. deserializácie jednotlivých formátov.

Uvažovali sme nad viacerými možnosťami: jednoduchým textom, XML a JSON formátom. Každý z týchto formátov má svoje výhody a nevýhody, ktoré teraz opíšeme.

- Jednoduchý text - predstavuje najefektívnejší spôsob, pretože vyžaduje najmenej režijných informácií t. j. znakov navyše, ktoré nenesú užitočnú

informáciu a slúžia iba na indikáciu, aby sme vedeli objekt spracovať. Je však náročnejší na ukladanie objektov, keďže si celú serializáciu a deserializáciu musíme doprogramovať sami.

- XML (eXtensible Markup Language) - značkovací jazyk, ktorý slúži na výmenu informácií cez internet. V rámci tohto formátu sú dáta umiestňované do značiek tzv. tagov, ktoré môžu mať ľubovoľný názov a indikujú, o akú informáciu sa jedná. Značky predstavujú réžiu, kedy sa do správy pridávajú dodatočné znaky bez výpovednej hodnoty. Na druhej strane sa XML dá omnoho ľahšie použiť na serializáciu objektov, vďaka existujúcim knižniciam pre Android.
- JSON (JavaScript Object Notation) - formát používaný na prenos dátových objektov v podobe „atribút-hodnota“. Bol vyvinutý za účelom nahradenia formátu XML. JSON je kompromisom medzi nenáročnosťou jednoduchého textu a ľahkosťou ukladania objektov z XML. Serializácia a deserializácia objektov do formátu JSON je, podobne ako pri XML, významne zjednodušená prítomnosťou knižníc pre Android. Hlavnými výhodami formátu JSON oproti XML je menšia veľkosť výsledných dát a ich lepšia čitateľnosť. Menšia veľkosť dát je spôsobená menším množstvom režijných znakov.

Porovnanie jednotlivých formátov je znázornené v Ukážka zdrojového kódu 1.

```
Jednoduchý text: 18.124;49.258;302.5
XML:
    <position>
        <latitude>18.124</latitude>
        <longitude>49.258</longitude>
        <altitude>302.5</altitude>
    </position>
JSON:
    {
        "latitude": "18.124",
        "longitude": "49.258",
        "altitude": "302.5"
    }
```

Ukážka zdrojového kódu 1: Príklady jednotlivých formátov správ

Nakoniec sme sa rozhodli použiť formát JSON, pretože je kompromisom, čo sa týka veľkosti prenášaných dát a pohodlnosti používania.

3.1.3. Typy správ

Vzhľadom na to, že používame protokol HTTP, rozdeľujeme správy do dvoch hlavných skupín: POST a GET. Správu vytvárame v Android aplikácii tak, že vytvoríme inštanciu triedy, transformujeme do JSON formátu a odošleme na server.

3.1.3.1. Správy typu POST

Správy typu POST slúžia na odosielanie dát na server. Medzi správy typu POST patria: RegisterDevice, PushRecord, AddFootprint, AddFootprints, AddKnownDevice a AddKnownDevices.

Spomenuté správy teraz popíšeme:

- RegisterDevice - Odosiela triedu „MobileDeviceRegistration“, ktorá zaregistruje mobilné zariadenie do databázy pod jedinečným identifikátorom, aby na server mohlo posielať dáta. Dáta odoslané na server nezaregistrovaným mobilným zariadením sú zahodené, pretože jeho identifikátor ešte nie je prítomný v databáze.

Mobilné zariadenia identifikujeme pomocou MAC adresy, čo je jedinečný identifikátor zariadenia v linkovej vrstve. Každý aktívny sieťový prvok musí mať svoju vlastnú unikátnu MAC adresu. V prípade, že by dve zariadenia mali rovnakú MAC adresu, spôsobovalo by to problémy nielen pri ARP požiadavkách, ale aj v duplicitu primárnych kľúčov v databáze. V našej aplikácii používame MAC adresu sieťovej karty Wi-Fi na jednoznačnú identifikáciu mobilného zariadenia.

- PushRecord - Odosiela triedu „MobileDeviceEntry“, ktorá obsahuje všetky objekty, ktoré sú zodpovedné za oznamovanie polohy t. j. „GPSEntry“, „WifiEntry“ a „BeaconEntry“.

Trieda „MobileDeviceEntry“ má nasledovné atribúty:

- „posX“, „posY“ a „elevation“ ukladajú finálne súradnice polohy vypočítané sledovacími triedami pre Wi-Fi a Bluetooth. Súradnice pre obidve spomenuté technológie sú preto len jedny, pretože budú spolupracovať pre získanie presnejšej polohy vnútri budov.
- „timeOfMeasurement“ ukladá čas, v ktorom boli súradnice zaznamenané.

-
- „mobileDeviceId“ hovorí, ktoré zariadenie odoslalo polohu. Je v ňom uložená MAC adresa Wi-Fi sieťovej karty mobilného zariadenia.
 - „projectId“ je identifikačné číslo projektu, ktoré určuje oblasť resp. budovu, v ktorej sa nachádzame.
 - „GPSEntry“, „WifiEntry“ a „BeaconEntry“, prostredníctvom ktorých vieme určiť súradnice podľa použitej technológie.

Trieda „GPSEntry“ má atribúty „latitude“, „longitude“ a „altitude“ na uloženie GPS súradníc severnej šírky, východnej dĺžky a nadmorskej výšky (v takomto poradí).

Trieda „WifiEntry“ a „BeaconEntry“ majú obe zhodne po dva atribúty, ktorými sú:

- „signalStrength“, ktorý meria silu signálu z Wi-Fi prístupového bodu alebo z Bluetooth Beacon zariadenia.
 - „knownDeviceId“, ktoré identifikuje buď Wi-Fi prístupový bod alebo Bluetooth Beacon zariadenie pomocou ich MAC adresy.
- AddFootprint - Pridáva stopu o pozícií do databázy. Stopa slúži na spresnenie informácií o polohe vnútri budov. Pomocou nej meriame silu signálu na konkrétnom mieste v budove z Wi-Fi alebo Bluetooth zariadení.

Jeden záznam sa ukladá do triedy „KdfpEntry“, ktorá má nasledovné atribúty:

- „knownDeviceId“ má rovnakú funkciu, ako rovnomenný atribút v triedach „WifiEntry“ alebo „BeaconEntry“.
- „wifiSignalStrength“ a „btSignalStrength“ udáva silu signálu konkrétneho Wifi alebo Bluetooth zariadenia.

Jednotlivé záznamy tvoria stopu, ktoré môžeme zhromažďovať do triedy „FootprintEntry“, keďže na jednom mieste môžeme prijímať signál z viacerých Wi-Fi alebo Bluetooth zariadení. Trieda obsahuje tieto atribúty:

- „projectId“ je identifikačné číslo projektu, ku ktorému Wi-Fi alebo Bluetooth zariadenie patrí resp. v ktorej budove sa nachádza.

-
- „longitude“, „latitude“ a „altitude“ sú súradnice Wi-Fi alebo Bluetooth zariadenia vnútri budovy.
 - „kdpfEntries“ je zoznam tried typu „KdfpEntry“ t. j. zoznam Wi-Fi a Bluetooth zariadení spolu s ich vlastnosťami, ktoré sme na konkrétnom mieste namerali.
 - AddFootprints - Podobne ako správa „AddFootprint“, len umožňuje poslať na server viacero stôp naraz.
 - AddKnownDevice - Pridáva jedno nové Wi-Fi alebo Bluetooth zariadenie.
 - AddKnownDevices - Podobne ako správa „AddKnownDevice“, len umožňuje poslať na server viacero Wi-Fi alebo Bluetooth zariadení naraz.

3.1.3.2. Správy typu GET

Správy typu GET slúžia na sťahovanie informácií zo servera. Medzi správy typu GET patria: KnownDevices, Projects, KnownDevices podľa parametra projectId, Footprints a Footprints podľa parametra projectId.

- KnownDevices - Vracia zoznam všetkých známych Wi-Fi a Bluetooth zariadení.
- Projects - Vracia zoznam všetkých projektov.
- KnownDevices podľa parametra projectId - Podobne, ako KnownDevices, len vracia zoznam zariadení patriace danému projektu.
- Footprints - Vracia zoznam všetkých stôp.
- Footprints podľa parametra projectId - Podobne, ako Footprints, len vracia zoznam stôp patriacich danému projektu.

3.2. Integrácia novej sledovacej technológie

Na to, aby sme mohli novú sledovaciu technológiu používať, musíme ju najprv zaregistrovať do už vytvoreného konceptu.

3.2.1. Vytvorenie sledovacej triedy

V prvom rade vytvoríme triedu s názvom „<názovTechnológie>Tracker“, pretože takúto konvenciu dodržiujeme v našej aplikácii (teda napr. GPSTracker, WifiTracker, atď.). Do hlavičky triedy pridáme, že vytvorená trieda má byť potomkom triedy „Tracker“. Ďalej

vytvoríme konštruktor, ktorý obsahuje jediný parameter „*paIPosition*“ typu *IPosition*, ktorý odovzdáme do konšuktora predka.

Teraz pomocou predka „*Tracker*“ vie jeho potomok komunikovať so zvyškom aplikácie resp. s triedou „*MainService*“. Spolupráca je zabezpečená tým, že ich predok t. j. trieda „*Tracker*“ má atribút „*aIPosition*“ typu „*IPosition*“. Hodnota spomenutého atribútu je inicializovaná vtedy, keď vytvárajú sa sledovacie triedy v triede „*MainService*“. Keďže trieda „*MainService*“ implementuje rozhranie „*IPosition*“, môže ho odovzdať do konštruktorov sledovacích tried. Potomok sa stará iba o získavanie dát, zatiaľ čo predok sa stará o komunikáciu v rámci aplikácie.

V prípade, že sledovacia trieda zistí súradnice polohy, musíme tieto súradnice odovzdať predkovi zavolaním jeho metódy „*updatePosition*“. Ako parametre vyplníme súradnice a identifikačné číslo sledovacej triedy. Identifikačné číslo ale ešte predtým zadeklarujeme ako konštantný statický celočíselný atribút v triede „*Tracker*“. Dodržiavame konvenciu, že identifikačné číslo prvej sledovacej triedy je „0“ a identifikačné číslo každej ďalšej sledovacej triedy inkrementujeme o jednotku, pretože toto číslo slúži aj ako index pre danú sledovaciu triedu v poli „*aTrackers*“ typu „*Tracker*“ v triede „*MainService*“.

3.2.2. Integrácia do služby

Novú sledovaciu triedu do služby zintegrujeme tak, že zmeníme veľkosť poľa „*aTrackers*“ v triede „*MainService*“, ktoré ukladá inštancie všetkých sledovacích tried. Na to, aby sme k jednotlivým sledovacím triedam mohli pristupovať (napr. zapnúť alebo vypnúť sledovanie), ich musíme pridať do tohto poľa. Následne v konšuktore triedy „*MainService*“ zavoláme metódu „*init*“, ktorá inicializuje pole „*aTrackers*“ inštanciami sledovacích tried, ktoré sa uložia pod takým indexom v poli, aké určuje ich identifikačné číslo, nachádzajúce sa v triede „*Trackers*“.

Ďalej v triede „*MainService*“, konkrétne v metóde „*startTracking*“ treba v periodicky opakujúcim sa cykle (vlákne) vytvoriť inštancie potrebných zapuzdrujúcich tried, ktoré následne naplníme potrebnými dátami z inštancie sledovacej triedy (pozri „loop každých *n* milisekúnd“ na Obrázok 4). Zapuzdrujúcim triedam sa budeme hlbšie venovať v podkapitole 3.2.4. Tu len spomenieme, že zapuzdrujúce triedy slúžia na výmenu informácií medzi Android aplikáciou a serverom.

Keďže sa spomínané vlákno opakuje v pravidelných krátkych časových okamihoch (menej ako 10 minút [30]), na jeho vytvorenie sme použili triedu „Handler“ a jeho metódou „postDelayed“ sme ho nastavili tak, aby sa opakovalo v pravidelnom intervale, pretože potrebujeme aktualizovať pozíciu mobilného zariadenia a následne ju odosielať na server. Vo vlákne sa najprv skontroluje, či je sledovacia trieda zapnutá. Ak áno, skontroluje sa, či atribút „aNotifPosition“ konkrétnej sledovacej triedy nie je prázdny. Pokiaľ prázdny nie je, potom sledovacia trieda aktualizovala hodnotu tohto atribútu a pokračujeme ďalej naplňaním zapuzdrujúcich tried.

Každý sledovacej triede prislúcha jedna zapuzdrúca trieda. Trieda „MobileDeviceEntry“ obsahuje všetky zapuzdrujúce triedy sledovacích tried. V aplikácií používame túto triedu na hlásenie informácií o polohách serveru. Ale ešte predtým, než ju odošleme serveru, ju musíme naplniť a skonvertovať do JSON formátu. Konverzií triedy „MobileDeviceEntry“ si povieme viac v podkapitole 3.2.4.

Komunikácia medzi triedami a ich životný cyklus je znázornený v sekvenčnom diagrame na Obrázok 4.

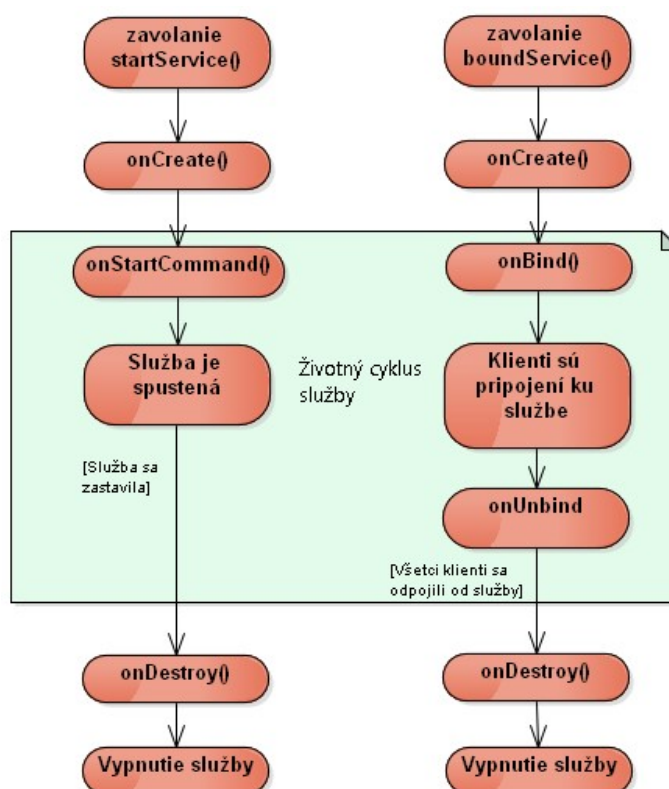
Pri komunikácii služby „MainService“ a aktivity „MainActivity“ využívame koncepty „BoundService“ a „Foreground Service“. „BoundService“ je implementácia triedy „Service“, ktorá umožňuje aktivite pripojiť sa ku službe. Komunikácia prebieha v rámci klient-server architektúry, pričom „serverom“ je v našej aplikácii trieda „MainService“ a „klientom“ aktivita „MainActivity“.

„Foreground Service“ je druh služby, pri ktorej je najmenej pravdepodobné, že ju Android ukončí. Tento druh služby sa hodí pre naše účely, pretože takáto služba môže byť spustená na pozadí veľmi dlhý čas (odhadom niekoľko dní). Špecifickou vlastnosťou takejto služby je nutnosť pridania notifikácie do notifikačnej lišty. Notifikácia sa v našej aplikácii nedá odstrániť z notifikačnej lišty, ale iba z aktivity.

Keď teda z aktivity „MainActivity“ zapneme sledovanie ľubovoľnou sledovacou triedou, najprv sa vytvorí inštancia služby „MainService“ metódou „onCreate“, a metóda „onStartCommand“ sa postará o zobrazenie notifikácie v notifikačnej lište. Potom sa aktivita zavolaním metódy „bindService“ pripojí ku službe a aktivuje sa sledovanie danou sledovacou triedou. V tomto momente beží aj služba aj aktivita. Keď zatvoríme aktivitu, odpojí sa od služby zavolaním metódy „unbindService“ v rámci „onPause“ metódy a uloží sa stav jej grafických prvkov. Služba beží stále na pozadí aj po ukončení aktivity. Ak opäť

otvoríme aplikáciu, aktivita sa znova pripojí ku existujúcej inštancii služby metódou „bindService“, čiže s ňou môže znova komunikovať. Keď sa rozhodneme, že službu z aktivity ukončíme, aktivita sa odpojí od služby a služba sa ukončí zavolaním svojej metódy „onDestroy“, čím uvoľní používané zdroje operačného systému.

Metódy „onCreate“, „onStartCommand“, „onBind“, „bindService“, „unbindService“ sú metódami životného cyklu rôznych druhov služieb a ich sled je znázornený na Obrázok 7.



Obrázok 7: Ukážka životného cyklu služby „Service“ (vľavo) a ukážka životného cyklu služby „Bound Service“ (vpravo) [31]

3.2.3. Integrácia do grafického rozhrania

Aby sme mohli správanie novej sledovacej triedy ovládať prostredníctvom grafického rozhrania, musíme najprv definovať, akým grafickým prvkom ju chceme ovládať (zaškrŕavacie políčko, prepínač a pod.) a následne definovať správanie v každom stave tohto grafického prvku.

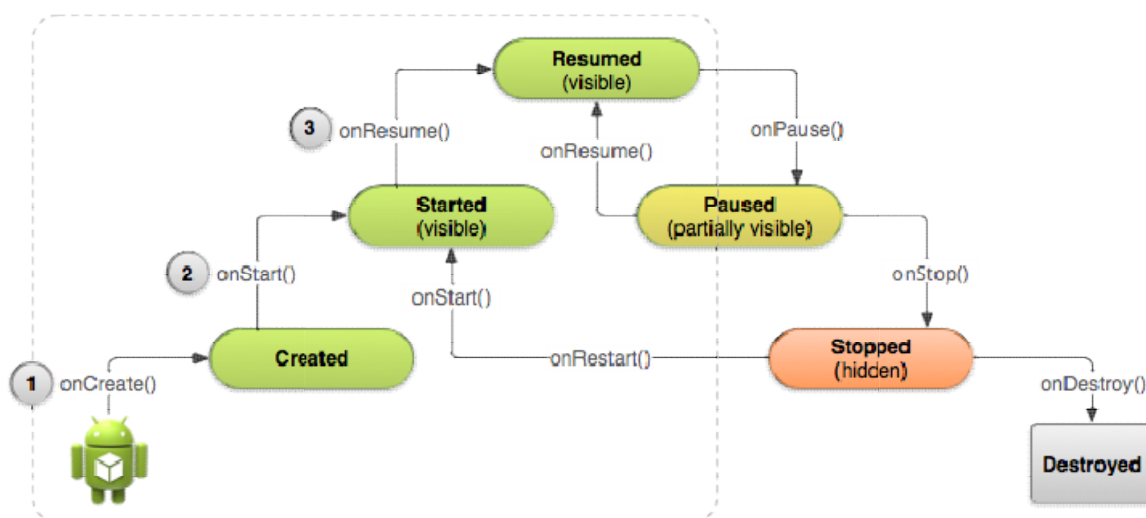
Na výber typu grafického prvku, ktorým používateľ bude ovládať stav novej sledovacej triedy, potrebujeme upraviť súbor „activity_main.xml“. Tento súbor definuje rozloženie jednotlivých grafických prvkov aktivity, ich súradnice, veľkosť zobrazovaného

písma atď. Každý grafický prvok má svoj identifikátor, pomocou ktorého ho vieme z aktivity ovládať. V našom prípade plní úlohu aktivity trieda „MainActivity“, ktorá zodpovedá za riadenie správania jednotlivých prvkov grafického rozhrania. Potom v triede „MainActivity“ tento grafický prvok uložíme do príslušného atribútu a prostredníctvom metód určíme správanie grafického prvku v jednotlivých stavoch (napr. kedy sa sledovacia trieda zapne alebo vypne).

Android aplikácia si vie uložiť stav grafického rozhrania aktivity, ako stav zaškrtnutých políčok, textových polí a iných grafických prvkov. Ukladanie grafického rozhrania sa uskutočňuje, akonáhle opustíme aktivitu, kedy Android zavolá metódu „onPause“. Metóda „onPause“ je jednou z metód životného cyklu aktivity. Zavolá sa v momente, keď napr. zamkneme obrazovku telefónu, máme prichádzajúci hovor alebo využívame multitasking na prepínanie medzi viacerými aplikáciami. Na ukladanie stavu grafických prvkov používame triedu „SharedPreferences“, ktorá vytvorí v adresári aplikácie XML súbor, do ktorého si uloží stav jednotlivých grafických prvkov. Grafické prvky sa do súboru ukladajú vo formáte „kľúč-hodnota“. Kľúčom môže byť ľubovoľný reťazec. Podmienkou je, aby každý grafický prvok mal svoj vlastný, unikátny kľúč, inak môžeme prepísať hodnotu iného grafického prvku.

Obnovovanie grafických prvkov v aktivite sa uskutočňuje pri opätovnom otvorení Android aplikácie, resp. keď sa používateľ prepne do našej aplikácie. Vtedy Android zavolá metódu „onResume“, čo je tiež metóda životného cyklu aktivity. V nej sa zo súboru uloženého triedou „SharedPreferences“ obnovia hodnoty do jednotlivých grafických prvkov a premenných podľa kľúča, pod ktorým sme hodnotu uložili.

Android poskytuje aj iné spôsoby ukladania a obnovenia stavu aktivity: „onSaveInstanceState“ a „onRestoreInstanceState“. Tieto metódy Android volá automaticky, podobne ako „onPause“ a „onResume“. Avšak nepoužívam tieto metódy, pretože ich Android v určitých situáciách nemusí zavolať, čo má za následok neuloženie stavu aktivity. Použitie metód „onPause“ a „onRestore“ spolu s ukladaním stavu cez triedu „SharedPreferences“ je spoľahlivejšie, pretože sa zavolajú vždy, keď opúšťame aplikáciu. Prehľad sledu metód životného cyklu aktivity a služby môžeme vidieť na Obrázok 8.



Obrázok 8: Sled metód životného cyklu aktivity[31]

3.2.4. Integrácia na server

Ako sme už spomenuli v závere kapitoly 3.1.1, Android aplikácia a databázový server spolu nekomunikujú priamo, ale prostredníctvom HTTP protokolu a REST rozhrania. REST (Representational State Transfer) je softvérový architektonický štýl dnešného webu. Je to abstrakcia, ktorá obsahuje množinu pravidiel na vytváranie protokolov. Jedným z protokolov, ktorý bol postavený na REST architektúre, je aj HTTP. HTTP je „Request-Response“ protokol: klient pošle na server požiadavku (Request) a server mu na ňu odpovie (Response) [32]. HTTP je protokol aplikačnej vrstvy, a využíva protokol TCP na transportnej vrstve, ktorému sme sa venovali v úvode kapitoly 3.1.1.

Na výmenu dát používame balíček tried, ktoré sú integrované tak v Android aplikácií, ako aj na serveri. Do týchto tried zapuzdrujeme rôzne dáta a vymieňame si správy medzi Android aplikáciou a serverom, ako sme uviedli v kapitole 0. Akékoľvek zmeny v týchto triedach v aplikácií (napr. názvy atribútov týchto tried) sa musia odzrkadliť aj na serveri a opačne. Inak hrozí, že bude vzájomná komunikácia medzi Android aplikáciou a serverom narušená.¹

Keďže je sledovacia trieda nová, treba ju integrovať na server, aby jej dáta vedel spracovať a uložiť do databázy. Integrácia spočíva vo vytvorení zapuzdrujúcej triedy na

¹ Hoci nemôžeme meniť názvy atribútov, typ atribútu zmeniť môžeme podľa potreby. Musíme ale dbať na dodržanie správneho formátu dát, na ktorom je potrebné dohodnúť sa s administrátorom servera.

serveri, ktorá bude ukladať informácie novej sledovacej triedy. Tú istú zapuzdrujúcu triedu treba duplikovať do Android aplikácie, aby ukladala informácie našej novej sledovacej triedy. V našom prípade sa jedná predovšetkým o triedu „MobileDeviceEntry“, pomocou ktorej odosielame informácie o polohách jednotlivých sledovacích tried na server. V tejto podkapitole sa budeme venovať jej naplneniu a konverzií do formátu JSON.

Akonáhle sme zistili súradnice pomocou ľubovoľnej sledovacej triedy, v periodickom cykle najprv vytvoríme inštanciu triedy „MobileDeviceEntry“, ktorej nastavíme tie atribúty, ktoré poznáme: nastavíme čas, kedy sa poloha namerala („setTimeOfMeasurement“), identifikačné číslo zariadenia („setMobileDeviceId“) a číslo projektu, ku ktorému sa meranie vzťahuje („setProjectId“). Zvyšné atribúty inicializujeme na prázdny reťazec resp. na nulu. Pri napĺňaní musíme dávať pozor, aby sme v každom atribúte v inštancií triedy „MobileDeviceEntry“ mali nejakú hodnotu, pretože server môže odmietnuť správu, v ktorej sú atribúty s hodnotou „null“.

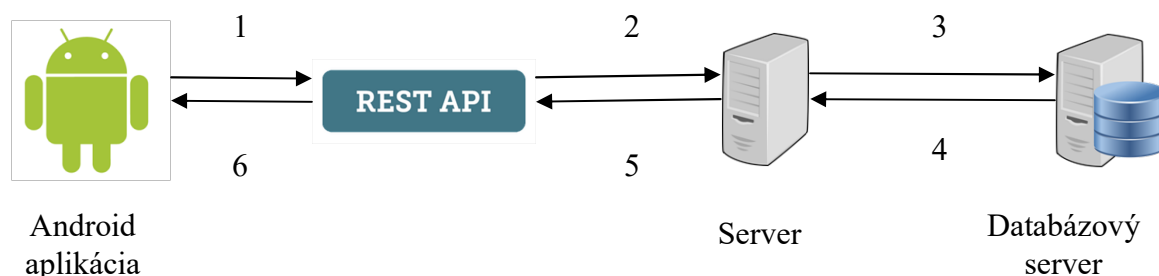
Inštanciu triedy „MobileDeviceEntry“ skonvertujeme v triede „MainService“ do formátu JSON pomocou knižnice „Gson“ a výsledok uložíme do premennej „jsonString“ typu „String“. Následne v tej istej triede vytvoríme inštanciu triedy „HashMap“. Záznamy do inštancie triedy „HashMap“ ukladáme vo formáte „kľúč-hodnota“, pričom obe sú typu „String“. Všetky kľúče sú deklarované v triede „ServerBridge“ ako verejné statické konštantné atribúty. Do vytvorenej inštancie triedy „HashMap“ pridáme pod kľúčom „ServerBridge.REQUEST_BODY“ reťazec „jsonString“, ktoré nesie telo prenášanej správy. Ďalej pod kľúčom „ServerBridge.URL“ pridáme adresu, na ktorú sa má správa doručiť. Všetky adresy sú deklarované v triede „ServerBridge“ ako verejné statické konštantné atribúty typu „String“, rovnako, ako kľúče. Adresu tvorí základ tzv. „URL_ROOT“, ku ktorému sa pripájajú rôzne prefixy závisiace na charaktere prenášanej správy (napr. „URL_POST_PUSH_RECORD“ pre zaslanie správy o pozíciách „PushRecord“). Nakoniec pridáme záznam s kľúčom „ServerBridge.METHOD_TYPE“, ktorý hovorí, či budeme cez HTTP odosielať na server správu typu GET („ServerBridge.GET“) alebo POST („ServerBridge.POST“). Podľa typu prenášanej správy sa potom vyberá algoritmus na odoslanie požiadavky na server (napr. „PushRecord“ je správa typu POST). Prehľad všetkých správ typu GET a POST uvádzame v kapitole 0. Takto pripravenú inštanciu triedy „HashMap“ odovzdáme triede „ServerBridge“.

Trieda „ServerBridge“ je zodpovedná za odoslanie požiadavky serveru. Na prenos správy na používame HTTP klienta. Komunikácia medzi triedou „MainService“ a triedou

„ServerBridge“ sa uskutočňuje pomocou vnorenej triedy „OdosliSpravuNaServer“, ktorá sídli v triede „ServerBridge“ a je potomkom triedy „AsyncTask“. „AsyncTask“ je tiež jednou z implementácií vlákna v operačnom systéme Android.

Inštanciu zapuzdrujúcej triedy prevedieme do JSON formátu a odošleme inštanciu triedy „ServerBridge“. Tá v samostatnom vlákne odošle HTTP klientom požiadavku na server. Avšak, ak by sme túto požiadavku odosielali v hlavnom vlákne aplikácie, aplikácia by sa stala nečinnou až dovtedy, pokým by sme zo servera nedostali odpoveď.

Odosielanie správ na server z Android aplikácie je znázornené na Obrázok 9. Po prevedení zapuzdrujúcej triedy do formátu JSON sa vytvorí a odošle HTTP požiadavka z Android aplikácie na REST rozhranie servera (1), ktorú server následne prijíme (2). Pokiaľ je požiadavka platná, odošle ju databázovému serveru (3). Databázový server buď zapíše/prečíta údaje do/z databázy a pošle ich naspäť serveru (4), ktorý znovu cez REST rozhranie odošle HTTP odpoveď Android aplikácii (5)(6). Android aplikácia prijíme odpoveď a spracuje ju podľa potreby.



Obrázok 9: Znázornenie komunikácie medzi Android aplikáciou a serverom

4. VYTVORENIE SLEDOVACEJ TRIEDY PRE ZBIERANIE DÁT POMOCOU GPS

Predchádzajúca kapitola sa venovala všeobecným konceptom, na ktorých je aplikácia postavená. V tejto časti sa zameriame na vytvorenie konkrétnej sledovacej technológie. Akékoľvek ďalšie rozšírenie aplikácie o novú sledovaciu technológiu je možné dosiahnuť veľmi jednoducho sledovaním tu uvedeného postupu. Z analýzy relevantných technológií (pozri kapitolu 2 alebo sumárne tabuľku Tabuľka 3) sa ako správna voľba pre naše účely javí GPS, preto sa v tejto kapitole budeme zaoberať integrovaním práve tejto technológie na sledovanie polohy mobilného zariadenia.

4.1. Integrácia do služby

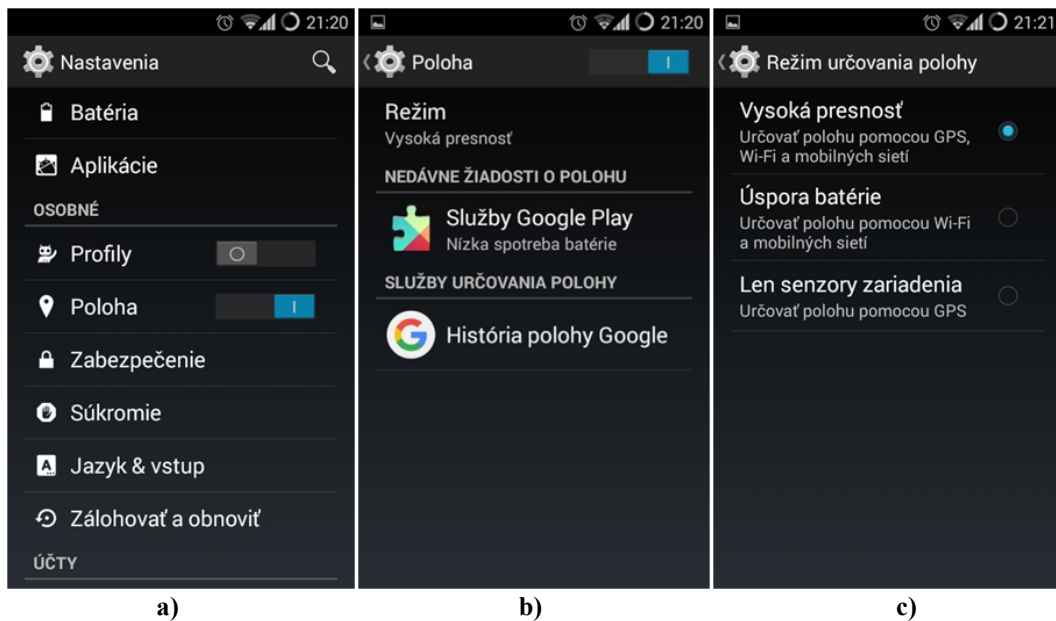
V triede „MainService“ v metóde „init“ pridáme záznam, v ktorom pridáme inštanciu triedy „GPSTracker“ do poľa „aTrackers“. Inštancia sa uloží pod takým indexom, aký sme určili príslušným identifikačným číslom v triede „Tracker“ (pozri záver kapitoly 3.2.1). Keďže je trieda „GPSTracker“ našou prvou sledovacou triedou, bude mať identifikačné číslo „0“.

Trieda „MainService“ vie informáciu o polohe z triedy „GPSTracker“ zachytiť v metóde „getGPSPosition“. Metóda „getGPSPosition“ vytvára anonymný objekt typu „Position“, ktorý dáva ako návratovú hodnotu.

Potom v periodicky opakujúcom sa cykle v metóde „startTracking“ získavame a odosielame GPS polohu. Keďže hodnota atribútu „aNotifPositionGPS“ je aktualizovaná v momente, keď sa zavolá metóda „onLocationChanged“, nemusíme sa dotazovať o polohu triedy „GPSTracker“, ale iba prečítame súradnice uložené v atribúte „aNotifPositionGPS“ a na základe nej vytvoríme záznam pre zapuzdrujúcu triedu. Následne sa vytvorí správa vo formáte JSON, ktorá sa odošle na server. Vytváraním tried, potrebných pre odosielanie informácií o GPS polohe sa budeme venovať bližšie v kapitole 4.4.

4.2. Vytvorenie triedy pre GPS sledovanie

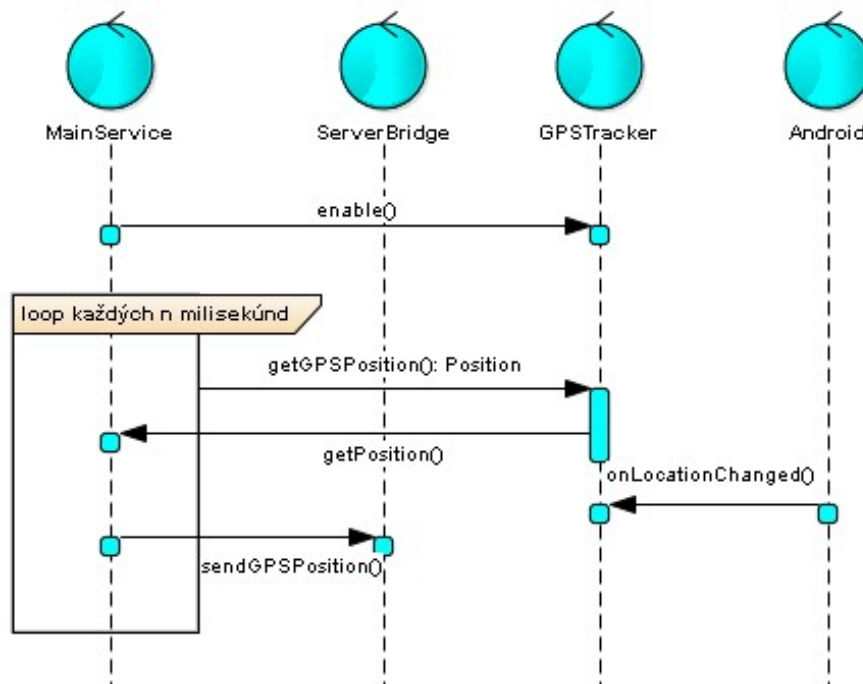
Za získavanie GPS súradníc je zodpovedná trieda „GPSTracker“. Na zisťovanie GPS súradníc používame predovšetkým signál z GPS satelitov. Na to, aby algoritmus mohol začať zisťovať polohu, musíme v Android nastaveniach zapnúť funkciu „Poloha“ (pozri Obrázok 10a) a „Režim“ nastaviť na „Vysokú presnosť“ (pozri Obrázok 10b a Obrázok 10c), čím umožníme Android zariadeniu použiť všetky dostupné metódy na zisťovanie GPS polohy.²



Obrázok 10: Úprava nastavení polohy v Android 4.4.4: a) Zapnutie polohy, b) Možnosti zisťovania polohy, c) Režim určovania polohy

Pokiaľ je „Poloha“ aktívna, algoritmus pokračuje ďalej tým, že si vypýta potrebné objekty od Androidu na zisťovanie polohy GPS a explicitne definujeme, že ju chceme získavať pomocou satelitov. Týmto sa zapne GPS sledovanie. Po jeho zapnutí začne Android automaticky volať metódu „onLocationChanged“, ktorá sa zavolá, ako to vyplýva z názvu metódy, iba vtedy, keď Android zistí, že sa zmenili súradnice GPS.

² Zapnutie používania GPS polohy sa môže v jednotlivých verziách operačného systému Android líšiť.



Obrázok 11: Oznamovanie GPS polohy

Metóda „onLocationChanged“ je v triede „GPSTracker“ implementovaná tak, že sa najprv skontroluje, či sa už zistili konkrétne súradnice. Ak áno, zmeníme atribút ukladajúci polohu „aNotifPositionGPS“ v triede „GPSTracker“. Trieda „MainService“ v pravidelných intervaloch získava hodnotu atribútu v metóde „startTracking“ tým, že zavolá metódu „getGPSPosition“, ktorý má ako návratovú hodnotu objekt typu „Position“ s vyplnenými súradnicami x, y, z a identifikačným číslom triedy, v prípade triedy „GPSTracker“ je to „0“. Jednotlivé súradnice získame prečítaním jednotlivých atribútov triedy „Position“, ktoré následne zabalíme do potrebných zapuzdrujúcich tried a odošleme triedou „ServerBridge“ na server.

následne sa oznámia súradnice polohy rozhraniu „IPosition“. Použijeme na to metódu „updatePosition“ z triedy „Tracker“. v ktorej ako parametre zadáme zistené súradnice a identifikačné číslo sledovacej triedy. V našej aplikácii má trieda „GPSTracker“ identifikačné číslo „0“. V metóde „updatePosition“ sa zavolá metóda „newPositionNotify“ patriaca rozhraniu „IPosition“, v ktorej vytvoríme anonymný objekt typu „Position“ s rovnakými parametrami, aké sme prijali v metóde „updatePosition“. Trieda „Position“ slúži iba na prenos informácií o polohe (súradníc x, y, z a identifikačného čísla sledovacej triedy, čiže „0“) z predka „Tracker“ do triedy „MainService“ cez rozhranie „IPosition“. Spomenutý postup znázorňujeme na Obrázok 11.

4.3. Integrácia do grafického rozhrania

Teraz potrebujeme integrovať GPS sledovanie do grafického rozhrania aplikácie, aby sme mohli zapnúť a vypnúť GPS sledovanie pomocou nami zvoleného grafického prvku. Za umiestnenie a vzhľad jednotlivých grafických prvkov aplikácie zodpovedá súbor „android_layout.xml“, ale ich správanie je definované v triede „MainActivity“.

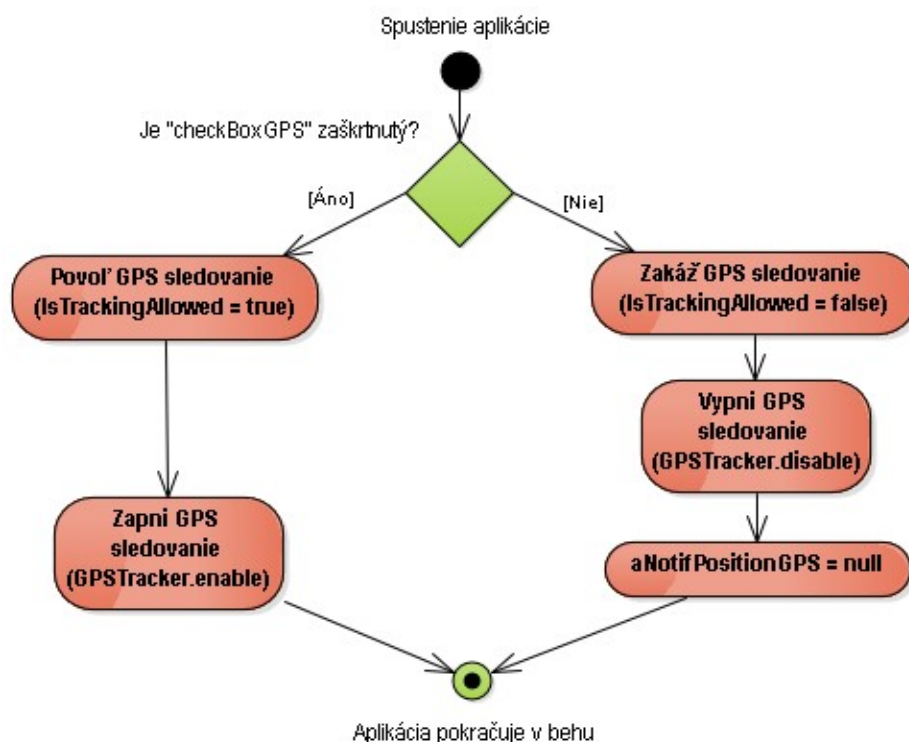
Najprv potrebujeme do súboru „android_layout.xml“ pridať grafický prvok, ktorým budeme ovládať GPS sledovanie. V našej aplikácii používame zaškrŕavacie políčka, preto si do tohto súboru zadefinujeme nové zaškrŕavacie políčko „CheckBox“ a nastavíme grafickému prvku unikátny identifikátor („checkBoxGPS“), pomocou ktorého s ním môžeme manipulovať.

Teraz v triede „MainActivity“ vytvoríme atribút „aGPSCheckbox“ typu „CheckBox“, prostredníctvom ktorého budeme neskôr zisťovať a meniť stav zaškrŕavacieho políčka „checkBoxGPS“. Ďalej v metóde „onCreate“, čo je metóda životného cyklu aktivity, inicializujeme tento atribút pomocou metódy „findViewById“, ktorá vyžaduje jeden parameter - identifikátor grafického prvku (pozri Ukážka zdrojového kódu 2).

```
aGPSCheckbox = (CheckBox) findViewById(R.id.checkBoxGPS);
```

Ukážka zdrojového kódu 2: Inicializácia zaškrŕavacieho políčka pre ovládanie GPS sledovania

Akonáhle je prvok inicializovaný, môžeme definovať, ako sa má správať v rôznych stavoch (napr. zaškrŕnutý/odškrŕnutý a pod.). To docielime tým, že ku grafickému prvku pripojíme „OnClickListener“. Metóda „onClick“ bude obsahovať, kvôli prehľadnosti, iba jednu metódu - „GPSCheckboxOperation“. V nej zadefinujeme činnosti prislúchajúce jednotlivým stavom zaškrŕavajúceho políčka „aGPSCheckbox“. Ak je „aGPSCheckbox“ zaškrŕnutý, nastavíme triede „GPSTracker“ atribút „isTrackingAllowed“ na „true“, čiže ňou môžeme zbierať súradnice. Následne zapneme sledovanie touto triedou metódou „enable“. Ku triede „GPSTracker“ pristupujeme cez pole „Trackers“ v triede „MainService“. V prípade, že zaškrŕavacie políčko odškrŕneme, atribút „isTrackingAllowed“ nastavíme na „false“, vypneme sledovanie triedou „GPSTracker“ a vymažeme hodnotu atribútu „aNotifPositionGPS“. Metóda „GPSCheckboxOperation“ je znázornená na Obrázok 12.



Obrázok 12: Priebeh zapínania a vypínania GPS sledovania cez grafické rozhranie

4.4. Integrácia na server

Ďalej potrebujeme použiť zapuzdrujúce triedy, ktoré budú niesť informácie o sledovacej triede „GPSTracker“. Zapuzdrujúce triedy slúžia na výmenu správ medzi Android aplikáciou a serverom (pozri kapitoly 0, 3.2.2 a 3.2.4). Na oznamovanie GPS polohy slúžia dve triedy: „GpsEntry“ a „MobileDeviceEntry“. Trieda „GpsEntry“ ukladá súradnice zistené triedou „GPSTracker“, čiže severnú šírku (latitude), východnú dĺžku (longitude) a nadmorskú výšku (altitude). Trieda „MobileDeviceEntry“ zase ukladá v príslušných atribútoch typu „Set“ zapuzdrujúce triedy pre všetky sledovacie technológie, vrátane triedy „GpsEntry“ a ďalšie dodatočné informácie, ako sme uviedli v kapitole 0 pri opisovaní správy „PushRecord“.

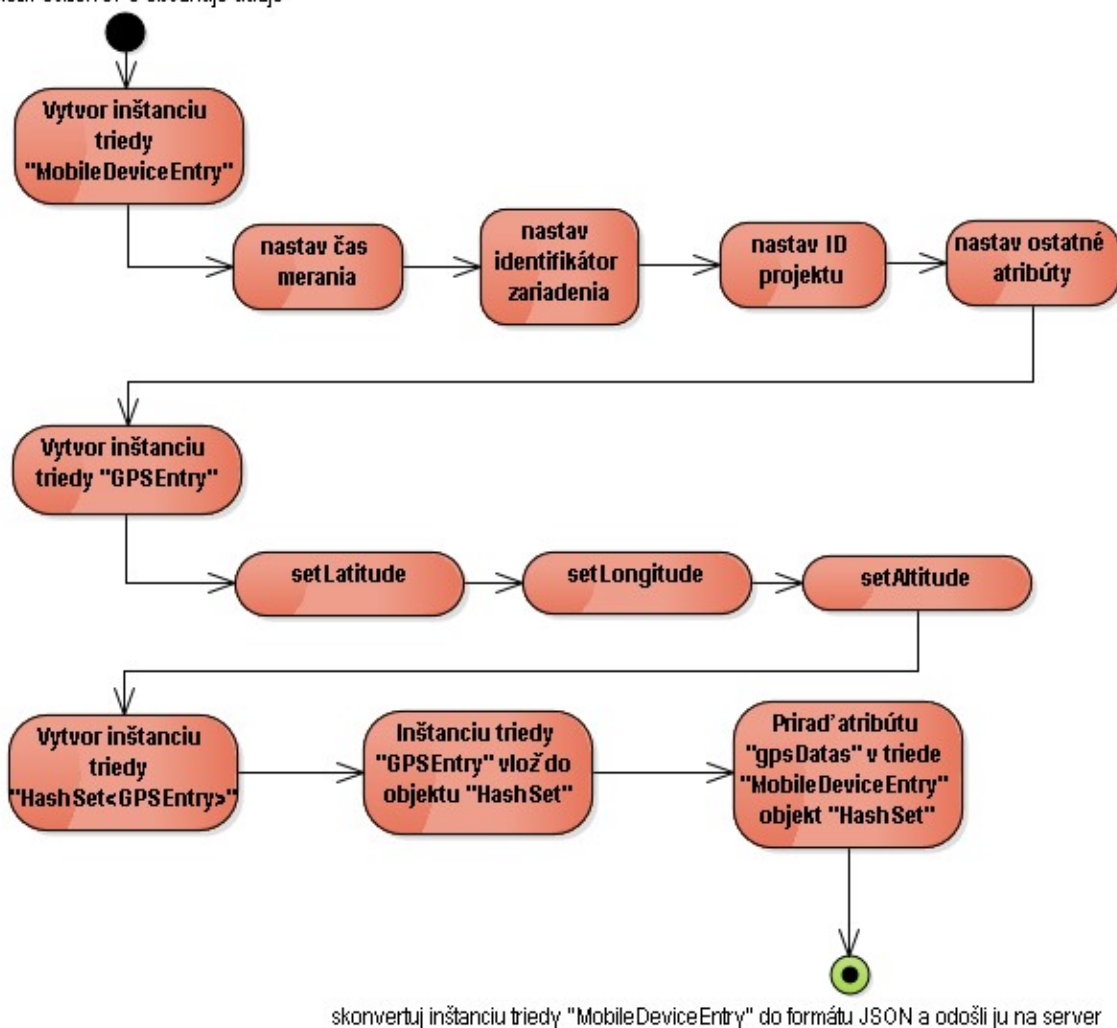
Ak chceme pridať novú sledovaciu triedu, potrebujeme k nej vytvoriť aj novú zapuzdrujúcu triedu s požadovanými atribútmi a zároveň do triedy „MobileDeviceEntry“ pridať atribút, ktorý bude rovnakého typu, ako zapuzdrujúca trieda, ktorú sme práve vytvorili.

Najprv vytvoríme inštanciu triedy „MobileDeviceEntry“, ako sme už spomenuli v kapitole 3.2.4. Ďalej vytvoríme inštanciu triedy „GpsEntry“, ktorej nastavíme jednotlivé súradnice podľa atribútu „aNotifPositionGPS“. Potom vytvoríme inštanciu triedy

„HashSet“ ukladajúcu typ „GpsEntry“ a do inštancie triedy „HashSet“ pridáme inštanciu triedy „GpsEntry“. Teraz vytvorenú inštanciu „HashSet“ nastavíme atribútu „gpsDatas“ v inšancií triedy „MobileDeviceEntry“. V tomto momente je inštancia triedy „MobileDeviceEntry“ pripravená na konverziu do formátu JSON. Konverzií do formátu JSON a odosielaní správy serveru sme sa venovali v kapitole 3.2.4. Priebeh naplňania a zapuzdrovania do tried v prípade GPS sledovania je znázornený na Obrázok 13.

Teraz je trieda „GPSTracker“ integrovaná do aplikácie a dá sa ovládať z grafického rozhrania.

aNotifPositionGPS obsahuje údaje

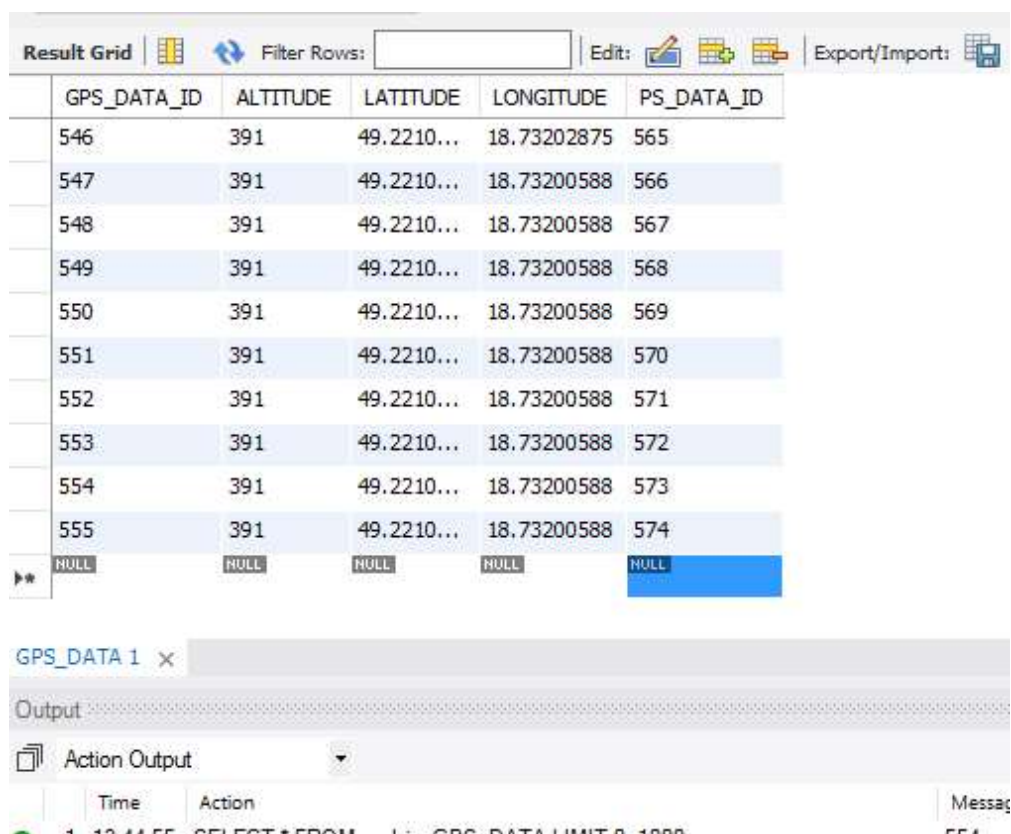


Obrázok 13: Znázornenie naplňania a zapuzdrovania tried

5. OVERENIE

Aplikáciu sme testovali v blízkosti Fakulty riadenia a informatiky na Žilinskej univerzite v Žiline. Aplikáciu sme nainštalovali na zariadenie „Samsung Galaxy S3 Mini“, ktoré obsahovalo operačný systém Android vo verzii 4.4.4. Obsah databázy budeme kontrolovať programom „MySQL Workbench“.

Najprv ukážeme stav databázy pred meraním o 13:55 9.5.2016.



The screenshot shows the MySQL Workbench interface. The top part displays a 'Result Grid' with a table containing GPS data. The table has columns: GPS_DATA_ID, ALTITUDE, LATITUDE, LONGITUDE, and PS_DATA_ID. The data rows show IDs from 546 to 555, all with an altitude of 391 and similar coordinates. The last row (ID 555) is highlighted in blue. Below the table, there are buttons for 'Filter Rows', 'Edit', and 'Export/Import'. The bottom part of the screenshot shows a window titled 'GPS_DATA 1' with an 'Output' section. The output shows a message: '1 13:55 SELECT FROM GPS_DATA LIMIT 1000'.

GPS_DATA_ID	ALTITUDE	LATITUDE	LONGITUDE	PS_DATA_ID
546	391	49.2210...	18.73202875	565
547	391	49.2210...	18.73200588	566
548	391	49.2210...	18.73200588	567
549	391	49.2210...	18.73200588	568
550	391	49.2210...	18.73200588	569
551	391	49.2210...	18.73200588	570
552	391	49.2210...	18.73200588	571
553	391	49.2210...	18.73200588	572
554	391	49.2210...	18.73200588	573
555	391	49.2210...	18.73200588	574
NULL	NULL	NULL	NULL	NULL

Obrázok 14: Databáza pred meraním

Hodnota „GPS_DATA_ID“ posledného záznamu je 555 a jeho súradnice vidíme na Obrázok 14

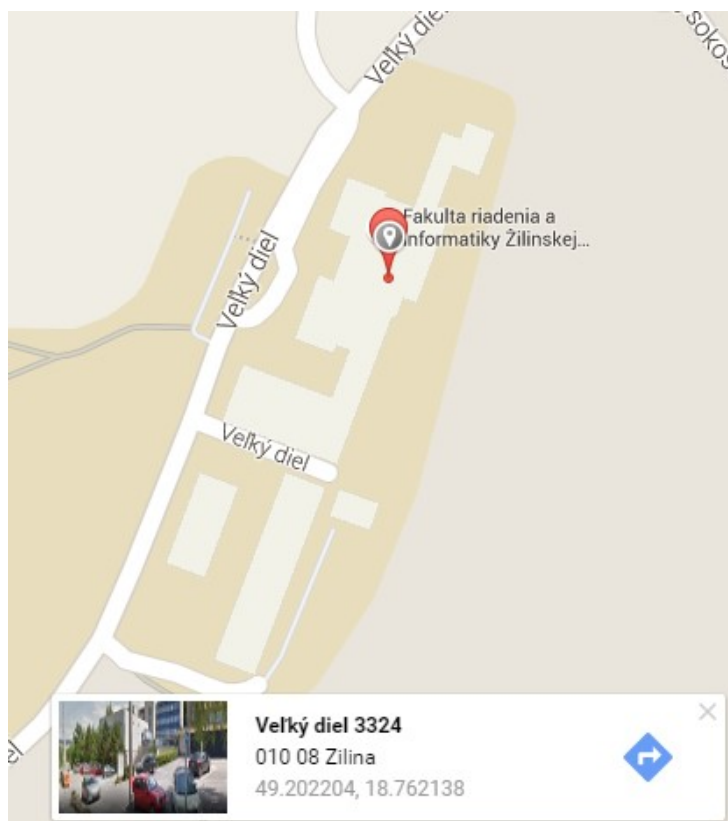
Potom sme išli do univerzitného mestečka, zamerali GPS signál a išli smerom ku fakulte. Zároveň musíme zapnúť aj mobilný internet, aby sa zistené GPS súradnice mohli okamžite odoslať na server.

Result Grid					
Filter Rows:		Edit: Export/Import: Wrap Cell Content			
	GPS_DATA_ID	ALTITUDE	LATITUDE	LONGITUDE	PS_DATA_ID
	646	411.7000000000...	49.20212089084089	18.761816024780273	665
	647	412.2000000000...	49.20209478121251	18.761879978701472	666
	648	412.2000000000...	49.20209478121251	18.761879978701472	667
	649	412.2000000000...	49.20209478121251	18.761879978701472	668
	650	412.5	49.20206691138446	18.761995062232018	669
	651	412.5	49.20206691138446	18.761995062232018	670
	652	412.5	49.20206691138446	18.761995062232018	671
	653	412.5	49.202060624957085	18.76206312328577	672
	654	412.5	49.202060624957085	18.76206312328577	673
	655	412.5	49.202060624957085	18.76206312328577	674
▶	656	412.7000000000...	49.20204247813672	18.76212707720697	675
★	NULL	NULL	NULL	NULL	NULL

GPS_DATA 1 x

Obrázok 15: Databáza po meraní

Ako vidíme na Obrázok 15, aplikácia naozaj odoslala informácie o GPS polohe na server. Posledný záznam má hodnotu „GPS_DATA_ID“ rovnú 656. Súradniciam polohy s „GPS_DATA_ID“ zodpovedá poloha Fakulty riadenia a informatiky, ako vidíme na oObrázok 16.



Obrázok 16: Overenie súradníc pomocou Google Maps v zázname s hodnotou „GPS_DATA_ID“ 656

Súradnice prvého záznamu merania („GPS_DATA_ID“ je 556) vidíme na Obrázok 17

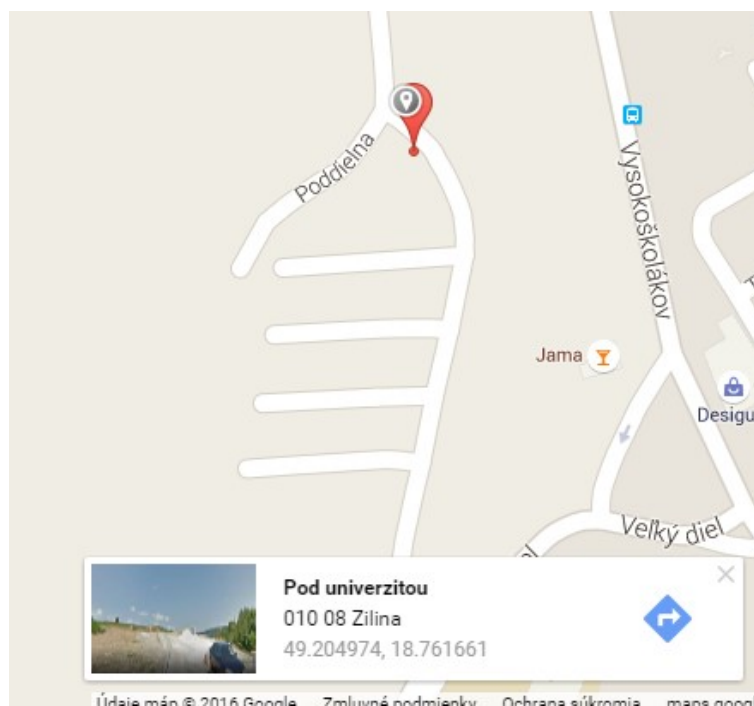
	GPS_DATA_ID	ALTITUDE	LATITUDE	LONGITUDE	PS_DATA_ID
	552	391	49.22108104	18.73200588	571
	553	391	49.22108104	18.73200588	572
	554	391	49.22108104	18.73200588	573
	555	391	49.22108104	18.73200588	574
▶	556	405.299999999...	49.20477074570954	18.76170370727775	575
	557	405.299999999...	49.20477074570954	18.76170370727775	576
	558	405.299999999...	49.20477074570954	18.76170370727775	577
	559	405.299999999...	49.20477074570954	18.76170370727775	578
	560	405.299999999...	49.20477074570954	18.76170370727775	579
	561	413.4	49.204753981903195	18.761693397536874	580
	562	413.4	49.204753981903195	18.761693397536874	581
	563	413.4	49.204753981903195	18.761693397536874	582
	564	413.4	49.204753981903195	18.761693397536874	583

GPS_DATA 1 x

Output

Obrázok 17: Záznam v databáze pod "GPS_DATA_ID" 556

Overenie súradníc prvého záznamu merania („GPS_DATA_ID“ je 556) vidíme na Obrázok 18.



Obrázok 18: Overenie súradníc pomocou Google Maps v zázname s hodnotou „GPS_DATA_ID“ 556

6. ZÁVER

Cieľom našej práce bolo vytvorenie Android aplikácie, ktorá slúži na získavanie informácií o polohe mobilného zariadenia a odosiela zozbierané informácie o polohe na server. Na to, aby sme takúto aplikáciu vytvorili sme museli preskúmať technológie GPS, Wi-Fi a Bluetooth. Porovnávaním sme vyhodnotili, že do našej aplikácie je najvhodnejšie použiť sledovanie polohy prostredníctvom GPS.

Pokračovali sme návrhom Android aplikácie. Aplikácia je navrhnutá tak, že podporuje sledovanie viacerými sledovacími technológiami. Aj napriek tomu, že sme implementovali iba jeden spôsob sledovania polohy, je možné jednoducho pridať sledovanie ďalšou sledovacou technológiou.

Ďalej sme vytvorili mechanizmus, ktorý odosiela anonymné dáta z aplikácie na server. Android aplikácia komunikuje so serverom prostredníctvom HTTP protokolu. Dáta sa na server odosielajú v pravidelných intervaloch a sú vo formáte JSON, ako to vyplýva z kapitoly 3.1.2.

Aplikáciu sme testovali v rôznych oblastiach a na rôznych verziách systému Android. Informácie o GPS polohe sme pomocou mobilného internetu odoslali na server. Následne sme skontrolovali, či záznamy o polohe zariadenia v databáze korešpondujú s jeho skutočnou polohou. Nakoniec sme usúdili, že dáta z Android aplikácie odosielané na server môžeme považovať za validné, ako ukazujeme v kapitole 5.

Vývoj našej aplikácie by mohol pokračovať napríklad v implementácii sledovacích algoritmov v budovách a interiéroch.

7. ZOZNAM POUŽITEJ LITERATÚRY

- [1] The Library of Congress. (2011, June) What is GPS? Everyday Mysteries. [Online]. <http://www.loc.gov/rr/scitech/mysteries/global.html>
- [2] National Coordination Office for Space-Based Positioning, Navigation, and Timing. (2014, Sep.) GPS.gov: GPS Accuracy. [Online]. <http://www.gps.gov/systems/gps/performance/accuracy/>
- [3] GPS Systems. (2014) A-GPS - What does it mean? | GPS Systems. [Online]. <http://gpssystems.net/agps/>
- [4] MiTAC Intl. (2011) What is trilateration? - Mio Technology. [Online]. <http://www.mio.com/technology-trilateration.htm>
- [5] Navigation, and Timing National Coordination Office for Space-Based Positioning. (2016, Mar.) GPS.gov: Space Segment. [Online]. <http://www.gps.gov/systems/gps/space/>
- [6] James R. CLYNCH. (2003) GPS Introduction. [Online]. <http://www.oc.nps.edu/oc2902w/gps/gpsoview.htm>
- [7] National Coordination Office for Space-Based Positioning, Navigation, and Timing. (2016, Mar.) constellation.jpg (1700×1450). [Online]. <http://www.gps.gov/multimedia/images/constellation.jpg>
- [8] CCM Benchmark Group. (2016, Mar.) What is WiFi and How Does it Work? [Online]. <http://ccm.net/faq/298-what-is-wifi-and-how-does-it-work>
- [9] About.com. (2015, Feb.) Wireless Standards: 802.11a, 802.11b/g/n and 802.11ac. [Online]. <http://compnetworking.about.com/cs/wireless80211/a/aa80211standard.htm>
- [10] Ltd. TP-LINK Technologies Co. (2016) 2.4GHz High Power Wireless Outdoor CPE TL-WA5210G - Welcome to TP-LINK. [Online]. <http://www.tp-link.us/products/details/TL-WA5210G.html>
- [11] Stefan VIEHBÖCK. (2011, Dec.) Brute forcing Wi-Fi Protected Setup. [Online]. https://sviehb.files.wordpress.com/2011/12/viehboeck_wps.pdf
- [12] Khamosh PATHAK. (2014, Dec.) Difference between WEP, WPA and WPA2 (Which is Secure). [Online]. <http://www.guidingtech.com/35711/difference-wep-wpa-2-secure/>
- [13] Ngan TENG YUEN. (2016, Jan.) WEP vs. WPA vs. WPA2 Comparison - How Do Wifi Wireless Get Hacked. [Online]. <http://www.geckoandfly.com/10380/wep-vs-wpa-vs-wpa2-comparison-table/>
- [14] MobileTechReview.com. (2016, Apr.) Android Phone Reviews - MobileTechReview. [Online]. <http://www.mobiletechreview.com/Android-Phone-Reviews.htm>
- [15] Apple Inc. (2016, Mar.) Potential sources of Wi-Fi and Bluetooth interference - Apple Support. [Online]. <https://support.apple.com/sk-sk/HT201542>
- [16] TP-LINK Technologies Co., Ltd. (2016) The differences between 2.4GHz and 5GHz Wireless - Welcome to TP-LINK. [Online]. <http://www.tp-link.com/en/faq->

- [17] Bluetooth SIG, Inc. (2016) Top Things to Know About Bluetooth Technology | Bluetooth Technology Website. [Online]. <https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics>
- [18] Nomi. (2016) Everything you need to know about Bluetooth Low-Energy Beacons - Nomi - In-store Analytics Platform and Sensors. [Online]. <http://www.nomi.com/resources/white-papers/bluetooth-low-energy-beacons/>
- [19] Matt BUCHANAN. (2009, Apr.) Giz Explains: Everything Bluetooth and Why Bluetooth 3.0 Is Better. [Online]. <http://gizmodo.com/5232940/giz-explains-everything-bluetooth-and-why-bluetooth-30-is-better>
- [20] Karen SIEGEL and Jeffrey SIEGEL. (2009) ActiveCaptain - Mobile Phones Series - Review of Bluetooth Headsets for Use on Boats. [Online]. <https://activecaptain.com/articles/mobilePhones/bluetoothHeadsets.php>
- [21] Sateesh VYSYARAJU. (2014, Aug.) Bluetooth. [Online]. <http://www.slideshare.net/sateeshvysyaraju/bluetooth-37715853>
- [22] Dev Chat. (2015, Dec.) Bluetooth Beacon Applications and Real World Developer - YouTube. [Online]. <https://www.youtube.com/watch?v=7DvnENKK6hU>
- [23] Future of Privacy Forum. (2014, Apr.) Contech Lab Ltd. | How small an iBeacon can be? [Online]. <http://www.contechlab.com/how-small-an-ibeacon-can-be/>
- [24] P. BAHL and V. N. PADMANABHAN, *RADAR: an in-building RF-based user location and tracking system,* in *Proceedings of 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00)*. Tel Aviv, Izrael, 2000.
- [25] Ramsey FARAGHER and Robert HARLE. (s. d.) An Analysis of the Accuracy of Bluetooth Low. [Online]. <http://www.cl.cam.ac.uk/~rmf25/papers/BLE.pdf>
- [26] HACKMAGEDDON. (2011, Oct.) TCP Split Handshake Attack Explained. [Online]. <http://www.hackmageddon.com/2011/04/17/tcp-split-handshake-attack-explained/>
- [27] Jeremy STRETCH. (2010, June) Understanding TCP Sequence and Acknowledgment Numbers - PacketLife.net. [Online]. <http://packetlife.net/blog/2010/jun/7/understanding-tcp-sequence-acknowledgment-numbers/>
- [28] Eileen MCNULTY. (2014, July) SQL vs. NoSQL- What You Need to Know - Dataconomy. [Online]. <http://dataconomy.com/sql-vs-nosql-need-know/>
- [29] Bill HOUGLUM. (2014, Feb.) Database System vs File System | Raima. [Online]. <http://raima.com/database-system-vs-file-system/>
- [30] Toni ALMEIDA. (2014, Apr.) android - How to run a method every X seconds - Stack Overflow. [Online]. <http://stackoverflow.com/questions/11434056/how-to-run-a-method-every-x-seconds>
- [31] Google, Inc. (s. d.) Starting an Activity | Android Developers. [Online]. <http://developer.android.com/training/basics/activity-lifecycle/starting.html>
- [32] Joe GREGORIO. (2008, Oct.) Intro to REST - YouTube. [Online]. <https://www.youtube.com/watch?v=YCcAE2SCQ6k>

[33] Google, Inc. (s. d.) Services | Android Developers. [Online].
<http://developer.android.com/guide/components/services.html#Foreground>