Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Analýza pohybu lidí v prostoru

Bc. Martin Pelant

Vedoucí práce: Ing. Jiří Hunka

3. května 2015

Poděkování

Rád bych na tomto místě poděkoval svému vedoucímu Ing. Jiřímu Hunkovi za přátelské vedení této práce. Dále bych chtěl poděkovat Ing. Lucii Hanákové za výpomoc s algoritmem pro výpočet polohy. V neposlední řadě děkuji všem, kteří se podíleli na testování aplikace, a tím přispěli k jejímu vylepšení.

Prohlášení

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

V Praze dne 3. května 2015 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2015 Martin Pelant. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Martin Pelant. *Analýza pohybu lidí v prostoru: Diplomová práce.* Praha: ČVUT v Praze, Fakulta informačních technologií, 2015.

Abstract

This thesis is about development of a system for monitoring people in a closed area. The system uses iBeacon technology to determine a precise location. It's developed in Java for devices with Android OS using Google App Engine as the server side platform. The system allows for configuring the environment, monitoring and analysis of the results in the form of a heatmap.

Keywords iBeacon, Bluetooth, heatmap, people monitoring, Android, Google App Engine

Abstrakt

Tato práce se zabývá vývojem systému pro sledování pohybu lidí v uzavřeném prostoru. Pro určení polohy se používá technologie iBeacon. Systém je naprogramován v jazyce Java pro mobilní zařízení s Android OS s podporou serverové části na platformě Google App Engine. Výsledný systém umožňuje nakonfigurovat prostředí pro monitorování, samotné monitorování a analýzu výsledků za pomoci heatmapy.

Klíčová slova iBeacon, Bluetooth, heatmapa, monitorování v uzavřeném prostoru, Android, Google App Engine

Obsah

Úvod **17**

1 Analýza **19**
1.1 Možnosti řešení 19
1.2 Funkční požadavky 20
1.3 Nefunkční požadavky 21
1.4 Existující řešení 23

2 Představení technologií **25**
2.1 Bluetooth 25
2.2 iBeacon 27
2.3 Android OS 29
2.4 Historie verzí Android 30
2.5 Podpora iBeacon u Android OS 31
2.6 App Engine 31
2.7 Architektura Android OS 32
2.8 Android Software Development Kit 35

3 Návrh **39**
3.1 Seznam obrazovek 39
3.2 API 41
3.3 Databáze 46

4 Implementace **49**
4.1 Specifika platformy 49
4.2 Architektura aplikace 56
4.3 Použité technologie 70

5 Testování **77**
5.1 iBeacon signál 78
5.2 Spotřeba energie 78
5.3 Stress test 80
5.4 Usability test 81

Závěr **89**

Literatura **91**

A Seznam použitých zkratek **95**

B Návrhy obrazovek **97**

C Výsledky měření vzdálenosti iBeaconů **107**

D Obsah přiloženého DVD **111**

Seznam obrázků

1.1 Infsoft heatmapa návštěvnosti 23

2.1 Příklady implementace Bluetooth 4.0 26
2.2 Bluetooth Link Layer Packet 28
2.3 Zastoupení verzí Android 32
2.4 Architektura AppEngine 33
2.5 Architektura operačního systému Android 34
2.6 Android Build Process 36
2.7 Android Studio 37

3.1 Eventuální konzistence 46
3.2 Databázové schéma backendu 47

4.1 Životní cyklus aktivity 51
4.2 Životní cyklus fragmentu 52
4.3 Diagram komponent 57
4.4 Zobrazení heatmapy 65
4.5 Použití větví v GITu 74

5.1 Nastavení projektu 87
5.2 Heatmapa 88

Seznam tabulek

5.1 Měření vzdálenosti iBeaconu 78
5.2 Měření spotřeby energie Nexus 5 79
5.3 Měření spotřeby energie HTC One mini 2 79
5.4 Post-test dotazník 86

Úvod

Analýza pohybu lidí v prostoru hraje velmi důležitou roli v marketingu. Obchodníci tak například mohou zjistit, zda rozmístění jejich zboží přitahuje návštěvníky k dané nabídce podle očekávání. Majitelé obchodních center mohou upravit nájem v závislosti na počtu návštěvníků v daných prostorech. Ředitel muzea může zhodnotit, které exponáty jsou nejvíce populární a zda je vhodné upravit trasy prohlídek. Podobně organizátoři festivalu mohou upravit rozmístění stánků podle frekvence návštěvníků.

Cílem této práce je realizace systému, který takovou analýzu umožní. Systém se bude skládat z administrátorské aplikace pro Android OS určené k nastavení projektů a jejich analýze, serverové části, na které se budou synchronizovat veškeré informace, a knihovny pro Android OS, která se použije v klientských aplikacích. Aplikace s touto knihovnou pak bude sledovat a zaznamenávat pohyb zařízení, na kterém je nainstalována, a získané záznamy bude anonymně posílat na server. Zároveň tato knihovna umožní aplikaci, která ji implementuje, poskytnout uživateli informaci o jeho aktuální poloze uvnitř budovy.

Pro zjištění polohy uvnitř budov se použije technologie iBeacon, která funguje na bázi Bluetooth Low Energy a která s velmi malými energetickými nároky umožňuje určit relativně přesně polohu zařízení.

Nejprve se provede analýza problému, ve které se specifikují požadavky, prozkoumají se již existující řešení, následně se rozeberou technologie klíčové pro realizaci tohoto systému a provede se návrh řešení. Poté se popíše samotná implementace včetně použitých optimalizací. Hotové řešení se řádně otestuje a opraví se případné chyby.

KAPITOLA 1

Analýza

1.1 Možnosti řešení

Analyzovat pohyb lidí v uzavřeném prostoru se dá následujícími způsoby:

- Pasivní zaznamenávání polohy pomocí speciálního HW detekujícího sílu signálu GSM a WiFi mobilních zařízení návštěvníků.
  - Výhody
    - není potřeba mít žádný SW nainstalovaný na mobilních zařízeních návštěvníků
    - nespotřebovávají se tedy na mobilních zařízeních žádné prostředky navíc
    - měří tudíž všechny mobilní zařízení - i tzv. „hloupé telefony“
  - Nevýhody
    - pasivní měření je méně přesné než aktivní měření
    - nemožnost poskytnutí návštěvníkům informaci o přesné poloze uvnitř objektu
    - vysoké pořizovací náklady
- Mystery Shopping – oslovení náhodného návštěvníka, aby provedl specifickou úlohu a poté vyplnil dotazník se zpětnou vazbou
  - Výhody
    - získání detailní zpětné vazby
  - Nevýhody
    - časově náročné
    - je možné provést pouze na vzorku návštěvníků

        * je potřeba mít na místě zaměstnance, aby oslovovali návštěvníky

- Aktivní zaznamenávání polohy
  - Výhody
    * přesnější měření než při volbě pasivního měření polohy
    * možnost poskytnout návštěvníkům informaci o jejich přesné poloze uvnitř objektu, implementovat do SW navigaci popř. jinou funkčnost závislou na poloze (zobrazení dotazníku při vstupu/opuštění určitého sektoru)
    * nižší pořizovací náklady než u výše zmíněných řešení
  - Nevýhody
    * je potřeba mít nainstalovaný SW na mobilních zařízeních návštěvníků
    * spotřebovává energii v mobilních zařízeních
  - Možnosti
    * Pomocí WiFi
      · náročnější na baterii zařízení [10]
      · méně přesné než v případě Bluetooth
    * Pomocí Bluetooth
      · méně náročné na baterii
      · přesnější
      · v závislosti na platformě je nutné mít zapnuté Bluetooth v zařízení

Pro diplomovou práci byl zvolen s ohledem na pořizovací náklady systém aktivního zaznamenávání polohy za pomocí krabiček iBeacon používajících technologii Bluetooth. Cena iBeaconů se pohybuje již od $6 za kus [25]. Při realizaci systému se však použijí ověřené iBeacony Kontakt, jejichž cena při zakoupeném množství 10 kusů je $25 za kus [24].

1.2 Funkční požadavky

Bude vytvořena administrátorská část (backend, mobilní aplikace pro správu projektů) a klientská část formou knihovny pro OS Android.

- Část administrátorská
  - Seznam projektů – Uživatel bude mít možnost vytvoření nového projektu, zobrazit již existující projekty a pokud má právo tak i editovat nebo mazat stávající projekty.

  - Detail projektu – V aplikaci půjde vložit a na mapě umístit plánek objektu. Na tento plánek pak bude možné umisťovat iBeacony, které zařízení vidí ve svém okolí, popř. upravovat pozici nebo mazat již existující iBeacony.
  - Seznam spolupracovníků – Projekty bude možné sdílet s více uživateli a nastavovat jim práva na pouze zobrazení projektů / heatmap, možnost editace plánků a iBeaconů popř. možnost editace seznamu spolupracovníků.
  - Heatmapa projektu – Jakmile se provede klientskou částí měření, bude možné si u projektu zobrazit heatmapu návštěvníků objektu. Bude možné si definovat časový úsek, ze kterého se má heatmapa vygenerovat.

- Část klientská
  - Zaznamenávání polohy – Jakmile se zapne Bluetooth v zařízení, začne aplikace implementující klientskou knihovnu scanovat okolí, zaznamenávat polohu a synchronizovat nasbíraná data se serverem.
  - Zobrazení polohy – Klientská aplikace bude mít možnost se napojit na službu v knihovně a zobrazit aktuální polohu zařízení získanou pomocí rozmístěných iBeacon krabiček v objektu.
  - Spuštění / vypnutí scanování – Aplikace implementující naší knihovnu bude mít možnost manuálně zapnout / vypnout scanování a vyvolat aktualizaci specifikace projektu.

1.3 Nefunkční požadavky

1.3.1 Uživatelské rozhraní

- Část administrátorská
  - Intuitivní uživatelské rozhraní v souladu s cílovou platformou s použitím standardních přístupů dané platformy.
  - Uživatelské rozhraní bude realizováno v anglickém jazyce.
- Část klientská nebude mít žádné uživatelské rozhraní. Případné UI bude mít na starosti aplikace implementující naši knihovnu.

1.3.2 Podporované platformy

- Aplikace je určena pro mobilní telefony se systémem Android.
- Vzhledem k dostupnosti Bluetooth LE API budou podporovány zařízení s verzí operačního systému 4.3 a novější, které disponují Bluetooth LE modulem.

- U administrátorské části bude rozložení ovládacích prvku a celkový návrh uživatelského rozhraní optimalizován pro rozlišení obrazovky 480x320, MDPI a vyšší.
- Jako backend bude použit Appengine s modulem Google Cloud Endpoints.

1.3.3 Získávání dat a dostupnost

Mobilní aplikace budou komunikovat se serverem pomocí REST API.

- Část administrátorská
  - K ověření uživatelů bude použita technologie OAuth 2.0 a bude vyžadován Google účet.
  - Seznam a detail projektů spolu se seznamem iBeaconů bude pro rychlejší odezvu uložen v lokální databázi a bude se synchronizovat se serverem.
  - Seznam spolupracovníků se bude vždy načítat ze serveru.
  - Histogram pro vygenerování heatmapy se připraví na serveru a na zařízení dojde za použití Google Maps Api pouze k vykreslení heatmapy na plánek objektu z připraveného histogramu.
- Část klientská
  - Pro optimalizaci spotřeby energie na klientských zařízeních se budou záznamy o poloze ukládat do lokální databáze a synchronizovat se serverem se budou po větších částech.
  - K identifikaci klientského zařízení bude použit SHA1 hash IMEI. Pokud není IMEI dostupný, bude při prvním spuštění aplikace s klientskou knihovnou vygenerován náhodný klíč, který se poté bude používat jako identifikátor.

1.3.4 Bluetooth scanování

- Část administrátorská
  - Scanování viditelných iBeaconů bude optimalizováno pro co nejrychlejší odezvu.
- Část klientská
  - Scanování iBeaconů bude optimalizováno pro co nejmenší spotřebu energie při spuštění na pozadí a pro co nejrychlejší detekci polohy v případě zobrazování polohy uživateli.

Obrázek 1.1: Infsoft heatmapa návštěvnosti

1.4 Existující řešení

Před začátkem práce bylo potřeba zjistit, zda existují již hotová řešení, jaké mají výhody a naopak čemu se při vývoji vyvarovat. Průzkum byl zaměřen jak na komplexní řešení zabývající se monitorováním lidí v prostoru, tak na samotné řešení mikrolokace na platformě Android za použití technologie Bluetooth LE. V současné době neexistuje žádné otevřená knihovna pro práci s iBeacon na zařízeních s Android OS, která by byla vhodná na použití do našeho systému a byla zvolena možnost vlastní implementace tohoto protokolu.

1.4.1 Infsoft indoor location analysis

http://www.infsoft.com/Products/Indoor-Location-Analytics

Komplexní řešení umožňující detekci a lokalizaci mobilních zařízení uvnitř i vně budov. Toto řešení však využívá k detekci infrastrukturu speciálního HW, které detekuje sílu mobilního signálu. Tím pádem jsou náklady na zprovoznění příliš vysoké a menší firmy si je nemohou dovolit. Mezi výhody patří bezesporu nezávislost na OS v mobilních zařízeních a přívětivě zpracované uživatelské rozhraní pro zobrazování výsledků měření. Umožňuje například zobrazení trendu počtu návštěvníků v průběhu dne, týdne nebo měsíce a tím zjistit špičku návštěvnosti.

1.4.2 Path intelligence

http://www.pathintelligence.com/technology

Opět řešení zabývající se výzkumem pohybu návštěvníků používající externí detektory, ovšem zaměřené primárně na obchodní domy. Funguje na bázi týdenních reportů obsahujících klíčové informace typu nejnavštěvovanější hodiny, frekvence návštěv nebo hodiny obchodníka. Dále je možné si najmout konzultanta, který pomůže maximalizovat příjmy, optimalizovat marketing a zlepšit požitek z nakupování.

1.4.3 Retail Sensing

http://www.retailsensing.com/video-people-counting.html

Toto řešení je zaměřené na počítání lidí procházejících určitým místem. K měření se využívá kamera umístěná nad měřeným místem namířená kolmo směrem dolů. Systém pak vyhodnocuje lidi procházející daným místem. Toto řešení má výhodu v tom, že měření je velmi přesné (dle autorů 98%) a v tom, že jsou započítány i osoby, které u sebe nemají zapnutý mobilní telefon. Nicméně velká nevýhoda spočívá v tom, že je možné tímto způsobem měřit lidi pouze v určitých místech, kde je nainstalována kamera.

1.4.4 Estimote

https://github.com/Estimote/Android-SDK

Výrobce iBeaconů s vlastní knihovnou pro vývojáře. Tato knihovna ovšem není open source a funguje pouze s iBeacony od Estimote. Toto řešení je navíc velmi zaměřené na spouštění událostí při vstupu regionu definovaném iBeacony, nikoli na získávání přesné polohy za použití iBeaconů. Kvůli tomuto důvodu a co nejlepší kontrole nad scanováním s možnosti optimalizovat spotřebu energie a nezávislosti na výrobcích iBeaconů bylo použití Estimote knihovny zavrženo.

1.4.5 AltBeacon

http://altbeacon.org

Otevřený standard pro proximity beacony. Jelikož iBeacon je uzavřený protokol, který má kompletně pod kontrolou Apple, vznikl nový standard AltBeacons. K otevřenému protokolu vzniklo i SDK, nicméně podpora zařízení je ve srovnání s protokolem iBeacon velmi mizivá. Z tohoto důvodu byl tento protokol zavržen.

KAPITOLA 2

Představení technologií

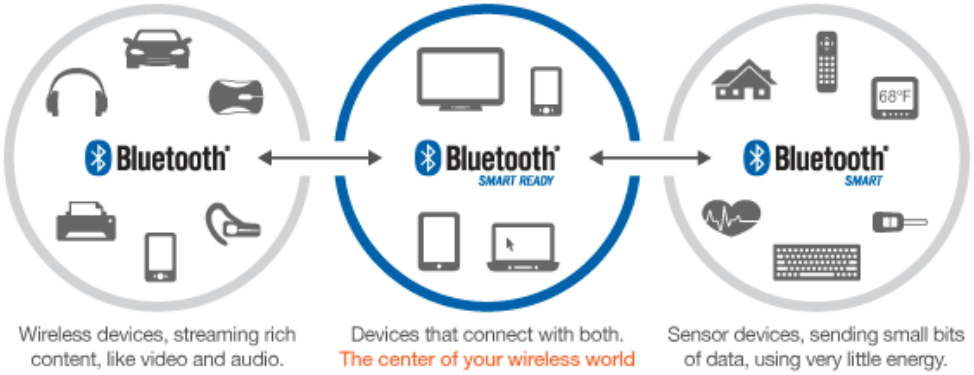
2.1 Bluetooth

Bluetooth je bezdrátová technologie umožňující komunikaci zařízení na krátkou vzdálenost. K tomu se používá pásmo na frekvenci 2.4 - 2.485 GHz. Technologie byla vyvinuta firmou Ericsson v roce 1994 jako bezdrátová náhrada za sériové rozhraní RS-232. Následně byla standardizována jako IEEE 802.15.1, avšak tento standard se již nevyvíjí. Na vývoj Bluetooth ovšem dohlíží skupina Bluetooth Special Interest Group (SIG), jejímiž členy je přes 20 000 společností po celém světě. Založena byla společnostmi Ericsson, IBM, Intel, Toshiba a Nokia a později se přidalo mnoho dalších společností. Všechna zařízení používající tuto technologii musí být certifikována.

Technologie funguje na principu master-slave kde 1 zařízení v roli serveru může komunikovat s až 7 zařízeními v roli klienta (limitováno 3 bitovou MAC adresou [4]), čímž se vytvoří lokální síť piconet. Každé zařízení zároveň může hrát roli serveru v jednom piconetu a roli klienta v piconetu druhém. Tím je možné propojit více piconetů do jedné společné sítě, která se nazývá scatternet.

V současnosti rozlišujeme tyto nejpoužívanější verze:[33]

- 1.2
  - Rychlost: základ (1Mb/s)
- 2.0 + EDR
  - Enhanced data rate (volitelná funkce)
  - Rychlost: základ + EDR (volitelné) = max 3 Mb/s
- 3.0 + HS
  - High speed



Obrázek 2.1: Příklady implementace Bluetooth 4.0
Zdroj: http://www.bluetooth.com/Pages/Bluetooth-Brand.aspx

  - Umožňuje rychlost přenosu až 24 Mb/s, avšak toho dosáhne vytvořením separátního 802.11 spojení, přes které se data touto rychlostí přenesou.
  - Rychlost: základ + EDR (volitelné) + HS (volitelné) = max 24 Mb/s

- 4.0
  - Přidána podpora pro Bluetooth Low Energy protokol pro nízkoenergetickou komunikaci se senzory.
  - 2 možnosti implementace:
    * Bluetooth Smart Ready: Jsou to převážně mobilní zařízení, které mohou komunikovat s Bluetooth Smart senzory. Umožňují tedy jak připojení přes BLE k senzorům tak přes BT k ostatním Bluetooth zařízením.
    * Bluetooth Smart: senzory, umožňující připojení pouze přes BLE
  - Rychlost: základ + EDR (volitelné) + HS (volitelné) = max 24 Mb/s , BLE – max 305 kb/s

2.1.1 Bluetooth Low Energy

Jak již bylo zmíněno výše, Bluetooth Low Energy bylo navrženo pro komunikaci s nízkou spotřebou energie. Při porovnání s WiFi, které při vysílání spotřebovává 210mW, potřebuje BLE pro vysílání jen zlomek energie – 0,147mW. Díky tomu si senzory používající tuto technologii vystačí pouze s klasickou knoflíkovou baterií, která dokáže napájet senzor v některých případech až rok.[29]

2.2 iBeacon

Technologie iBeacon umožňuje mikrolokaci a interakci mobilního zařízení ve fyzickém světě. To dosud bylo možné například pomocí technologií NFC (Near Field Communication), které umožňovalo komunikaci maximálně do vzdálenosti 10 cm nebo pomocí QR kódu, který ovšem vyžaduje přímou viditelnost mezi kódem a zařízením. iBeacon naproti tomu funguje právě na bázi Bluetooth Low Energy, která je součástí specifikace Bluetooth verze 4.0. Tím pádem je vzdálenost komunikace omezená pouze sílou signálu Bluetooth, což je ve většině případů okolo 10 metrů. Ve specifikaci však není určen maximální dosah a tak může být za použití silného vysílače dosah klidně přes 100 metrů.[7]

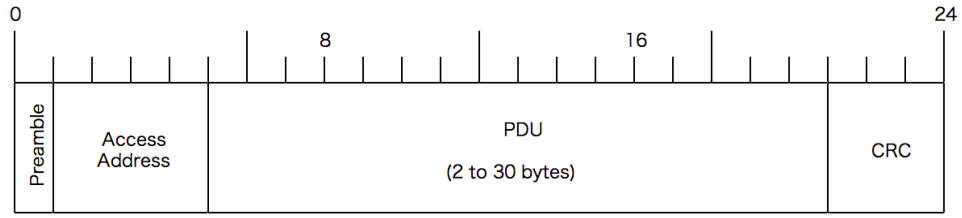
Technologie byla představena společností Apple v roce 2013 a jako první byla implementována do zařízení s iOS 7. Sensory podporující tento protokol fungují v módu TX-only a periodicky (nejčastěji v intervalech 100ms - 1s) vysílají do svého okolí formou tzv. Advertisingu následující informace sloužící k identifikaci senzoru: [26]

- proximity UUID: unikátní ID výrobce
- major: číselné ID úrovně 1
- minor: číselné ID úrovně 2
- Tx power: síla vysílaného signálu; slouží k určení vzdálenosti zařízení od senzoru

Na MAC adresu se nemůžeme spolehnout, jelikož Apple ji může u některých zařízení náhodně měnit. Musíme tedy použít výše uvedené údaje. Ty lze u většiny senzorů libovolně nastavit podle potřeby, čímž můžeme vytvořit tzv Regiony. Region se definuje kombinací UUID, Major a Minor hodnot. Jako příklad může mít aplikace pro obchodní řetězec region definovaný pouze pomocí UUID, který monitoruje uživatele ve všech obchodech. Když uživatel vstoupí do nějakého obchodu, můžou se použít Major a Minor hodnoty pro rozpoznání daného obchodu nebo definování konkrétnějších regionů, jako jsou třeba sekce v obchodě.

BLE v režimu Advertising vysílá tzv. Bluetooth Link Layer Packet, který může mít velikost 80-376 bajtů a má následující strukturu:

- Preamble: Slouží pro správu protokolu. Advertising packety mají vždy hodnotu `10101010b`.
- Access Address: Pro Advertising packety je adresa vždy `0x8E89BED6` (`10001110100010011011111011010110b`).
- Protocol Data Unit (PDU): Jsou 2 typy PDU, 1 pro Advertising packety a druhý pro datové packety.



Obrázek 2.2: Struktura Bluetooth Link Layer Packetu
Zdroj: http://j2abro.blogspot.cz/2014/06/understanding-bluetooth-advertising.html

- CRC: 3 bajty kontrolního součtu PDU.

Konkrétní packet s iBeacon Advertisingem může vypadat následovně: [9]

```
d6 be 89 8e # Access address for advertising data (this is always
        the same fixed value)
40 # Advertising Channel PDU Header byte 0. Contains:
        (type = 0), (tx add = 1), (rx add = 0)
24 # Advertising Channel PDU Header byte 1. Contains:
        (length = total bytes of the advertising payload + 6 bytes
        for the BLE mac address.)
05 a2 17 6e 3d 71 # Bluetooth Mac address (note this is a
        spoofed address)
02 01 1a 1a ff 4c 00 02 15 e2 c5 6d b5 df fb 48 d2 b0 60
d0 f5 a7 10 96 e0 00 00 00 00 c5 # Bluetooth advertisement
52 ab 8d 38 a5 # checksum
```

Samotné iBeacon data jsou zakódované v Bluetooth advertisement.

```
02 # Number of bytes that follow in first AD structure
01 # Flags AD type
1A # Flags value 0x1A = 000011010
bit 0 (OFF) LE Limited Discoverable Mode
bit 1 (ON) LE General Discoverable Mode
bit 2 (OFF) BR/EDR Not Supported
bit 3 (ON) Simultaneous LE and BR/EDR to Same Device Capable
      (controller)
bit 4 (ON) Simultaneous LE and BR/EDR to Same Device Capable
      (Host)
1A # Number of bytes that follow in second (and last) AD structure
FF # Manufacturer specific data AD type
4C 00 # Company identifier code (0x004C == Apple)
02 # Byte 0 of iBeacon advertisement indicator
15 # Byte 1 of iBeacon advertisement indicator
e2 c5 6d b5 df fb 48 d2 b0 60 d0 f5 a7 10 96 e0 # iBeacon uuid
00 00 # major
00 00 # minor
c5 # The 2nd complement of the calibrated Tx Power
```

Tento packet má UUID `E2C56DB5-DFFB-48D2-B060-D0F5A71096E0`, Major 0 a minor 0.

2.3 Android OS

Android je rozsáhlá mobilní platforma založena na otevřeném Linuxovém jádru. Zahrnuje v sobě operační systém, middleware a klíčové mobilní aplikace. Je optimalizována pro dotekové mobilní zařízení. Její unikátnost však spočívá v tom, že ji aktivně vyvíjí Open Handset Alliance, zároveň ji ale zdarma poskytuje výrobcům zařízení a operátorů pod open source licencí. Systém si tak může kdokoli libovolně upravit a použít na jakémkoli zařízení, aniž by musel platit licenční poplatky. [28]

Open Handset Alliance je skupina hardwarových, softwarových a telekomunikačních společností, které přispívají k vývoji Androidu. Tuto skupinu založil Google v roce 2007, 2 roky poté co odkoupil společnost Android Inc. vedenou Andy Rubinem, který zůstal v čele Android týmu až do roku 2013. [6] Nyní v čele stojí Sundar Pichai, který je zároveň na vedoucí pozici oddělení, starající se o vývoj webového prohlížeče Chrome. [5]

Systém byl vytvořen od samotných základů s cílem využít všech možností, jaké mobilní zařízení může nabídnout. Například mobilní aplikace pro tento systém může použít jakoukoli systémovou funkčnost jako je volání, posílání SMS či použití fotoaparátu. Toto umožňuje vývojářům vytvářet bohatší a konzistentní aplikace. Tomu napomáhá i fakt, že systém nerozlišuje mezi systémovými aplikacemi a aplikacemi třetích stran. Všechny aplikace si jsou rovny. Uživatelé tak mohou používat například vlastní klávesnici, vlastní aplikaci na telefonování nebo vlastní fotoaparát. [27]

Právě jako OS pro fotoaparáty byl tento systém v úplných začátcích zamýšlen. V roce 2004, kdy Andy Rubin hledal investory, ukázal na přednášce vizi fotoaparátu připojeného k domácímu počítači, který byl připojený k „Android datacentru“. Nicméně Andy si uvědomil, že trh s mobilními fotoaparáty je příliš malý, načež změnil business plán a 5 měsíců poté prohlásil Android jako open source platformu pro mobilní zařízení. V té době ještě neexistoval iPhone. Microsoft a Symbian byli jediní konkurenti. [23]

Tento mladý systém se těší stále větší popularitě a v současné době pohání stovky milionů mobilních zařízení ve více než 190 zemích světa a tím se stává nejpoužívanějším mobilním operačním systémem na světě. Každý den se prodají 3 miliony nových zařízení poháněných tímto systémem. [32]

2.4 Historie verzí Android

První verze Android 1.0 byla uvolněna v říjnu 2008 v jediném telefonu T-Mobile G1.

Postupem času Google vydává nové aktualizace a aktuálně již existuje 22 iterací tohoto systému.

Pro zjednodušení lze však rozlišovat 4 základní verze: Android 1.x, 2.x, 4.x a 5.x. Android 3.x je vynechán záměrně a to z důvodu, že tato verze byla uvolněna na velmi malé množství tabletů jako první verze systému připravená na tablety a nyní existuje pouze na zanedbatelném množství zařízení, které nedostaly aktualizaci na novější verzi. (viz 2.3)

Android 4.0 a novější je připravený běžet jak na tabletech, tak na telefonech, což přináší radikální změny v API. Referenční telefony již nemají hardwarová tlačítka, dříve schované menu se přesunulo na viditelnou pozici v horní části obrazovky, vývojářům přibyl nový způsob, jak zobrazit informace na obrazovce pomocí tzv. Fragmentů a uživatelům se otevřela nová možnost navigace pomocí tzv. ActionBaru. Dále přibylo mnoho nových API pro usnadnění práce prakticky s každou komponentou. Kvůli zpětné kompatibilitě nových funkcí Google vydal tzv. Support library, která umožňuje použít některou nově přidanou funkčnost na Android verzi 2.0 a novější.

Pro nás nejdůležitější verzí je však Android 4.3, kde přibyla podpora pro Bluetooth Low Energy. Všechny zařízení, které se prodávají s již nainstalovaným Android OS 4.3 a novějším, tak umožňují scanování iBeaconů.

Podpora pro Bluetooth Low Energy byla nadále vylepšena s verzí Android 5.0, která přinesla nové API pro scanování BLE zařízení s možností nastavit prioritu (úsporné scanování, popř. rychlá odezva) a také použít filtry pro zařízení, které nás při scanování zajímají. Tato nová API přispívají k menší energetické spotřebě při spuštěném scanování na pozadí nebo naopak k rychlejší odezvě, pokud uživatel přepne aplikaci do popředí.

2.5 Podpora iBeacon u Android OS

Android technologii iBeacon přímo nepodporuje, nicméně je možné tuto podporu naimplementovat. Jak je napsáno v předchozí sekci, od Andorid verze 5.0 je možné nastavit filtry pro LE scanování pouze iBeacon zařízení, které potřebujeme. Nelze sice přímo definovat proximity UUID, major ani minor, lze však nastavit filtr pro `Manufacture data`, kde jsou všechny tyto údaje uloženy (metoda `setManufacturerData(manufacturerId, manufacturerData, manufacturerDataMask)`). Jako parametr `ManufacturerId` se použije `0x004C`, což je hodnota pro Apple. Dále se musí definovat maska:

```
00 00
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF (UUID filtr)
FF FF (major, pokud chceme, jinak 00 00)
FF FF (minor, pokud chceme, jinak 00 00)
```

a příslušný filtr pro námi požadovaný region:

```
00 00
e2 c5 6d b5 df fb 48 d2 b0 60 d0 f5 a7 10 96 e0 (UUID filtr)
00 01 (major)
00 02 (minor)
```

Tento filtr nám vyfiltruje pouze iBeacon s UUID `E2C56DB5-DFFB-48D2-B060-D0F5A71096E0`, major 1 a minor 1.

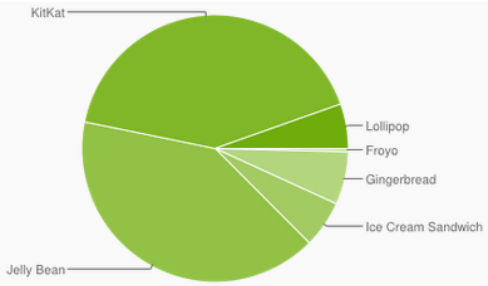
U zařízení s Android OS < 5.0 takto nastavovat filtry nelze a tudíž je potřeba procházet všechny nascanované zařízení a profiltrovat je až v naší aplikaci dodatečně.

2.6 App Engine

Google App Engine je služba PaaS pro vývoj a běh webových aplikací na serverech Google. Aplikace běží na datacentrech ve vlastním sandboxu a podle potřeby je možné aplikace automaticky škálovat v případě zvětšujícího se počtu requestů. Platforma byla představena v dubnu 2008 a v současnosti nabízí podporu pro celou řadu programovacích jazyků, mezi kterými jsou například Java, Python, Go nebo PHP. [2]

Pro App Engine existuje řada rozšíření. Mezi ně patří i Google Cloud Endpoints. Jedná se o sadu nástrojů a knihoven umožňující generování API spolu s knihovnou pro klientskou aplikaci, která s ním zajišťuje komunikaci. Součástí Google Cloud Endpoints je i plná podpora technologie OAuth 2.0.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.7%
4.1.x	Jelly Bean	16	16.5%
4.2.x		17	18.6%
4.3		18	5.6%
4.4	KitKat	19	41.4%
5.0	Lollipop	21	5.0%
5.1		22	0.4%



Obrázek 2.3: Aktuální zastoupení verzí na trhu. Měřeno na základě přihlášených zařízení do obchodu Google Play během období 14 dnů.
Zdroj: http://developer.android.com/about/dashboards/index.html

Klientské knihovny jsou dostupné pro platformy Android, iOS i JavaScript. [12]

2.7 Architektura Android OS

Architektura operačního systému Android je rozdělena do 5 vrstev. Každá vrstva má svůj účel a nemusí být přímo oddělena od ostatních vrstev. 2.5

2.7.1 Linuxové jádro

Android využívá Linuxové jádro ve verzi 3.4 pro základní systémové služby jako je bezpečnost, správa paměti, správa procesů, síťové prostředky nebo ovladače. Kernel slouží jako abstraktní vrstva mezi hardwarem a softwarovou vrstvou. [28]

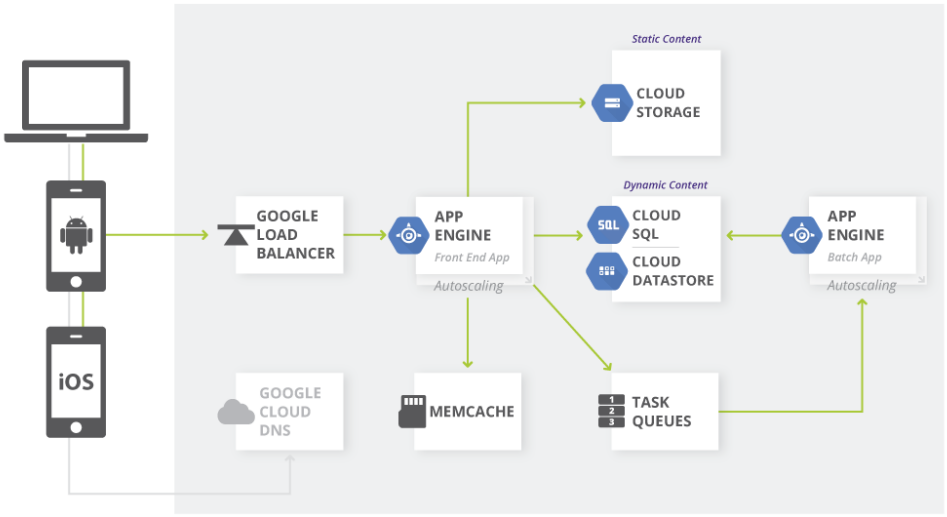
2.7.2 Knihovny

Systém obsahuje sadu C/C++ knihoven, které jsou využívané komponentami systému. Jsou také dostupné vývojářům skrz Android Application Framework. [28]

2.7.3 Android runtime

Android obsahuje sadu knihoven, které poskytují většinu funkčnosti dostupné v základních knihovnách jazyka Java.

Každá aplikace běží ve vlastním procesu, ve vlastní instanci Dalvik virtuálního stroje. Dalvik byl vyvinut tak, aby umožnil zařízení spouštět efektivně



Obrázek 2.4: Architektura AppEngine
Zdroj: https://cloud.google.com/solutions/architecture/webapp

vícero VM. Dalvik VM spouští soubory ve formátu Dalvik Executable (.dex), který je optimalizovaný pro minimální paměťovou náročnost. VM je založený na registrech a spouští třídy zkompilované pomocí kompilátoru jazyka Java, které byly převedené do .dex formátu pomocí poskytovaného `dx` nástroje.

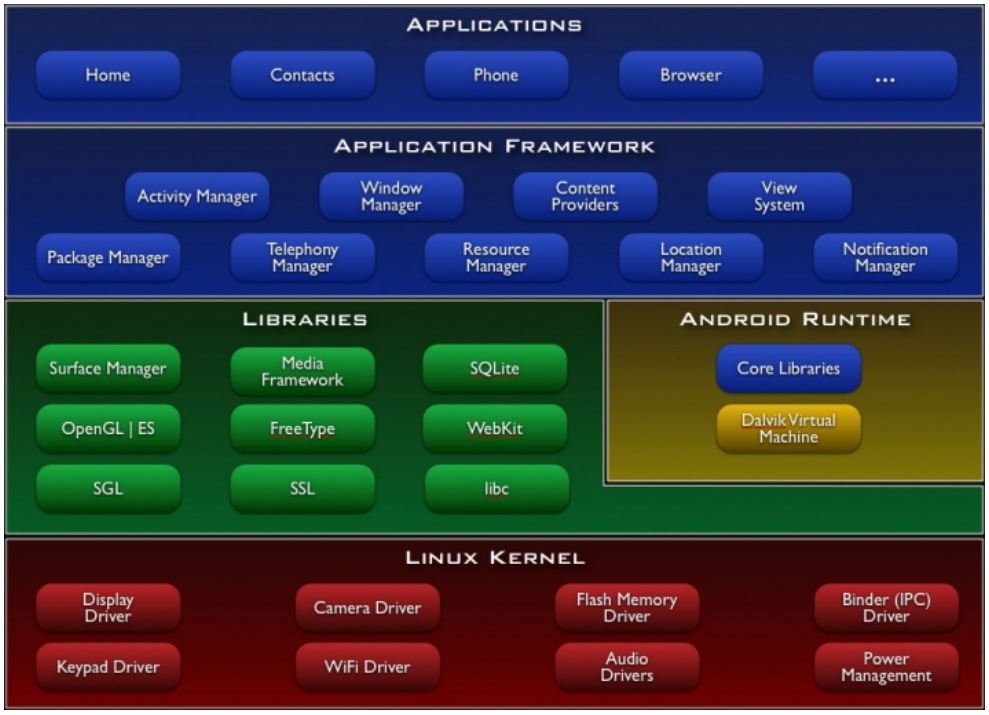
Dalvik VM využívá základních vlastností Linuxového jádra, jako je správa paměti nebo práce s vlákny. [28]

Od Android verze 5.0 Google nahradil Dalvik za vlastní ART (Android Runtime). ART na rozdíl od Dalviku, který používá Just in Time kompilátor, kompiluje kód již při instalaci aplikace. Tím se zvýší rychlost spouštění aplikací a zlepší se spotřeba energie zařízení. Nicméně toto řešení má i negativní vliv na dobu instalace aplikace, jelikož se právě při instalaci aplikace musí kompilovat. [1]

2.7.4 Application Framework

Díky své otevřenosti umožňuje Android vývojářům budovat kvalitní aplikace. Mohou využívat hardware na zařízení, přistupovat k lokačním službám, spouštět služby na pozadí, přidávat notifikace do notifikační lišty a mnoho dalšího.

Vývojáři mají plný přístup ke stejnému API, jako mají systémové aplikace. Architektura je navržena tak, aby bylo možné jednoduché znovupoužití jed-



Obrázek 2.5: Diagram znázorňující základní komponenty OS Android. Zdroj: http://developer.android.com/about/versions/index.html

notlivých komponent – aplikace můžou zpřístupnit svoje funkce jiným aplikacím (za předpokladu, že mají příslušná práva, které musí uživatel nejdřív odsouhlasit). Stejný mechanismus umožňuje tvorbu a použití vlastních komponent systému.

Mezi základní komponenty patří například Activity manager, který se stará o životní cyklus aplikací a poskytuje orientaci v zásobníku s aplikacemi nebo Resource manager, který umožňuje aplikacím přistupovat ke zdrojům, jako jsou texty, grafika nebo přidané soubory aplikace. [28]

2.7.5 Aplikace

Poslední vrstvu tvoří samotné aplikace, které využívají uživatelé. Tyto aplikace mohou být předinstalované (e-mailový klient, internetový prohlížeč) nebo si je uživatel může stáhnout z obchodu Google play. Po povolení instalace aplikací z neznámých zdrojů je také možné danou aplikace stáhnout jako .apk soubor a ten pak nainstalovat. Tento způsob se ovšem pro běžné uživatele nedoporučuje, protože Google nemá nad neznámými zdroji žádnou kontrolu a může se tak do zařízení jednoduše dostat škodlivý kód. Poslední možností, jak do zařízení dostat aplikaci je pomocí USB kabelu a nástroje

ADB, ovšem i tuto možnost je nutné kvůli bezpečnosti povolit v nastavení systému. [28]

2.8 Android Software Development Kit

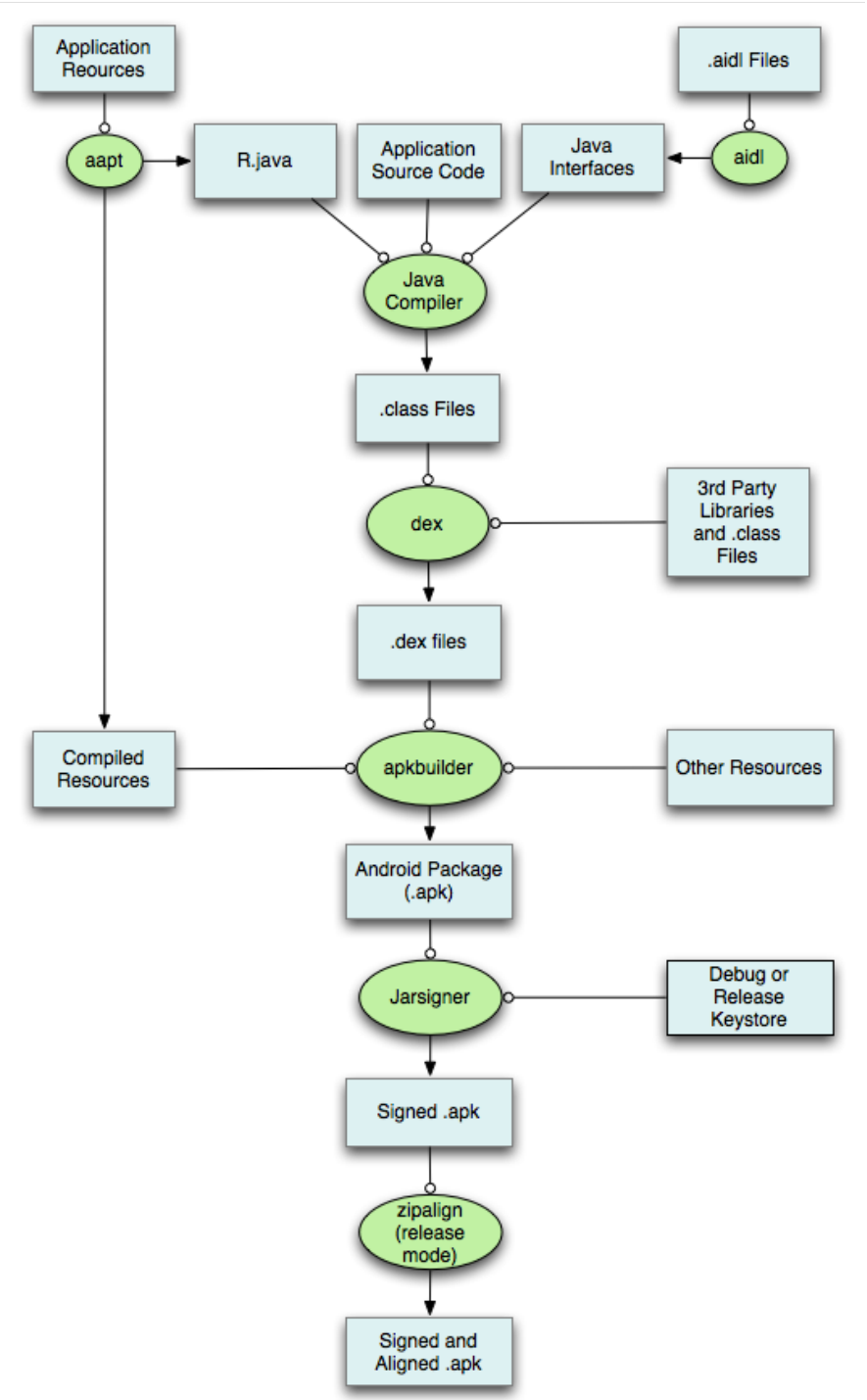
Android SDK poskytuje vývojáři API knihovny a vývojářské nástroje potřebné k vytváření, testování a spuštění aplikací pro Android. Skládá se z Android Studio IDE, Android SDK nástrojů, Android emulátoru a vybrané verze Android OS včetně zdrojových kódů. [17]

2.8.1 Android Build System

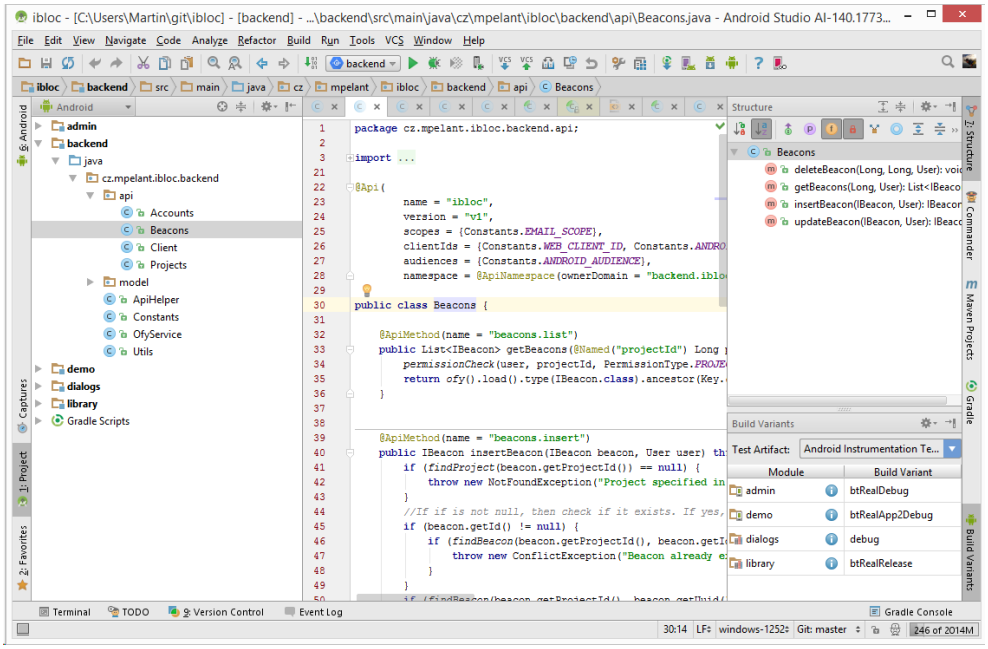
Pro build aplikací se používá Gradle Build System s Android pluginem. Ten nahradil dříve používaný ANT s cílem zjednodušit konfiguraci aplikace, zvýšit znovupoužitelnost build kódu, umožnit více variant jedné aplikace a lepší integrace do IDE. [13] Mezi další výhody patří i plná podpora Maven repositáře, přes které jsou distribuovány knihovny nebo pokročilý Manifest merger. Jelikož aplikace můžou využívat knihovny, které mají nadefinované vlastní komponenty a jiné vlastnosti v Android Manifest souboru, hraje Manifest merger velmi důležitou roli při jejich buildění. Ve výchozím stavu jsou všechny komponenty z knihoven přidány do výsledného Manifestu, nicméně je možné toto chování ovlivnit a například upravit atributy u některých komponent nebo je kompletně vynechat. Odpadá tak nutnost ručně vkládat komponenty ze všech knihoven do hlavní aplikace, jako tomu bylo za použití předchozího build systému ANT.[14]

2.8.2 Androd Studio

Plnou podporu výše zmíněného Gradle build systému využívá oficiální vývojové prostředí Android studio, což je upravená verze populární IntelliJ IDEA obohacená o funkce specifické pro Android vývoj. Nahrazuje Eclipse jako Googlem oficiálně podporované IDE pro vývoj mobilních aplikací pro platformu Android. Kromě mobilních aplikací podporuje Android Studio i vytváření serverových aplikací na platformě App Engine. Je tedy možné mít v rámci jednoho projektu kompletní systém – modul mobilní aplikace i backend na App Engine. Toto velmi usnadňuje vývoj, jelikož při použití Google Cloud Endpoints se změny v API na backendu okamžitě projeví v mobilní aplikaci díky napojení na automatické generování klientské knihovny.



Obrázek 2.6: Diagram procesu buildu aplikace pro Android.
Zdroj: https://developer.android.com/sdk/installing/studio-build.html



Obrázek 2.7: Příklad multi modulového projektu v Android Studio

KAPITOLA 3

Návrh

3.1 Seznam obrazovek

Zde budou popsány všechny obrazovky Administrátorské aplikace.

3.1.1 Seznam projektů

Jedná se o výchozí obrazovku po spuštění aplikace. Pokud jde o první spuštění, zobrazí se systémový dialog, kde si uživatel vybere, se kterým Google účtem se chce přihlásit. Poté se zobrazí seznam všech projektů, do kterých má uživatel přístup.

Kliknutím na projekt se uživatel dostane k jeho heatmapě 3.1.4. Dlouhým stiskem na projekt je možné si vybrat z následujících možností:

- Nastavení projektu: přechod na obrazovku 3.1.2, pokud má uživatel právo na editaci projektu, jinak je volba neaktivní
- Přejmenování projektu: zobrazení dialogu, ve kterém může uživatel změnit jméno projektu, pokud má právo na editaci projektu, jinak je volba neaktivní
- Smazání projektu: zobrazení potvrzovacího dialogu, ve kterém může uživatel smazat projekt, pokud má právo na editaci projektu, jinak je volba neaktivní

Z této obrazovky je také možné výběrem z menu vytvořit nový projekt.

3.1.2 Nastavení projektu

Na této obrazovce je možné přidávat, odebírat nebo upravovat umístění iBeaconů. Dále je také možné nastavit pozici plánku objektu. Plánek společne

s iBeacony ve formě pinů jsou zobrazeny na mapách Google a vše je možné pomocí standardních multidotekových gest přibližovat nebo oddalovat a také natáčet a naklánět. Na základě již umístěných iBeaconů se uživateli zobrazuje aktuální vypočítaná poloha (pokud má zařízení zapnuté Bluetooth a je v dosahu zvolených iBeaconů).

3.1.3 Přidání iBeacon

Pro přidávání iBeaconů se na telefonech zobrazí samotná obrazovka, kde se zobrazí seznam viditelných iBeaconů v okolí zařízení. V seznamu je vidět UUID, major i minor iBeaconu a zároveň se zobrazí přibližná vzdálenost zařízení od daného iBeaconu, pomocí které je seznam také seřazený. iBeacony, které jsou již do projektu přidány, jsou označeny, a po kliku na ně se uživateli zobrazí, kde jsou umístěny na mapě. Po vybrání nového iBeaconu je uživatel dotázán na volitelné pojmenování a poté se přidá pin doprostřed aktuálně zobrazené mapy s možností ho přetáhnout na požadované místo. U tabletů je tato obrazovka součástí nastavení projektu.

3.1.4 Heatmapa projektu

Na této obrazovce může uživatel prohlížet tepelnou mapu daného projektu znázorňující počet zařízení na daném místě za daný časový úsek. Heatmapa se zobrazuje na mapách Google, na kterých je umístěn plánek objektu. Je možné zvolit požadovaný časový úsek, ze kterého se mají zobrazovat data. Po kliknutí na mapu se zobrazí počet zařízení na daném místě.

3.1.5 Nastavení spolupracovníků

Pokud bude mít uživatel oprávnění na správu spolupracovníků, umožní mu tato obrazovka přidávat nové, mazat nebo upravovat stávající spolupracovníky a jejich oprávnění. Uživatel může mít tato oprávnění:

- Zobrazení projektu: možnost zobrazit si heatmapu a umístění iBeaconů
- Změna projektu: možnost editovat vlastnosti projektu včetně umístění iBeaconů a plánu objektu
- Správa spolupracovníků: možnost editovat spolupracovníky a jejich práva
- Vlastník projektu: speciální oprávnění znázorňující vlastníka projektu. Projekt může mít pouze jednoho vlastníka. Změnu vlastníka projektu může provést pouze samotný vlastník. Tím pádem nikdo, kromě samotného vlastníka, nemůže vlastníkovi z projektu odepřít přístup.

3.2 API

Pro komunikaci mezi mobilním zařízením a backendem je použita platforma Google Cloud Endpoints, což umožňuje zaobalit REST příkazy vygenerovanou knihovnou, s jejíž pomocí je možné navázat spojení s patřičnou metodou na serveru. Samotná komunikace pak probíhá pomocí REST protokolu, kde obsah je přenášen formátem JSON. Popisován zde tedy bude návrh ve formátu, v jakém se definuje na Google Cloud Endpoints. Jako autentikaci v administrační části byl použit protokol OAuth 2.0, který Google Cloud Endpoints plně podporuje. Uživatel se tedy bude autentikovat svým Google účtem, který má nastavený v mobilním zařízení. Jelikož pro účel tohoto projektu stačí znát pouze e-mailovou adresu, není potřeba se uživatelů tázat na žádné další oprávnění a systém Android za pomocí Google Play Services pouze vybídne uživatele ke zvolení, s jakým účtem se chce do aplikace přihlásit (pokud jich má více).

Pro řešení konfliktů byla zvolena metoda last update. Při každé změně entity na serveru se vygeneruje a uloží časové razítko, které se spolu s obsahem přenáší do klientů. Ti při změně dané entity pošlou zpět na server časové razítko, které obdrželi ze serveru při poslední aktualizaci a pokud je na serveru daná entita s novějším časovým razítkem, změna se neprovede. Toto chování tedy znemožní provádění změn, pokud zařízení nemá uložený aktuální obsah a tudíž se nemůže stát, že by uživatel nevědomky přepsal novější obsah, aniž by o něm věděl. Jelikož jsou vždy prováděné změny rozděleny na malé části (každá iBeacona se např. ukládá samostatně) a změny se posílají na server ihned, jakmile se provedou, je toto chování vyhovující.

3.2.1 API administrace

Administrační aplikace bude přistupovat k následujícímu API:

- projects.list – Vrátí seznam projektů, ke kterým má daný uživatel oprávnění.
  - Vstupní parametry:
    - User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401.
  - Vrácená hodnota:
    - List<Project> – Seznam projektů, ke kterým má uživatel oprávnění včetně informace o daném oprávnění.
- projects.insert – Vytvoří nový projekt a nastaví danému uživateli všechna oprávnění.
  - Vstupní parametry:
    - Project – Projekt, který se má vytvořit.

        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401.
    - Vrácená hodnota:
        * Project – Vytvořený projekt včetně nového ID.

- projects.update – Upraví projekt se zvoleným ID. Pokud projekt se zvoleným ID neexistuje, vrátí se chyba 404.
    - Vstupní parametry:
        * Project – Projekt, který se má upravit.
        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401. Pokud daný uživatel nemá oprávnění na úpravu projektu, vrátí se chyba 403.
    - Vrácená hodnota:
        * Project – Upravený projekt.

- projects.delete – Smaže projekt se zvoleným ID. Pokud projekt se zvoleným ID neexistuje, vrátí se chyba 404.
    - Vstupní parametry:
        * id – ID projektu, který se má smazat.
        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401. Pokud daný uživatel nemá oprávnění na úpravu projektu, vrátí se chyba 403.

- beacons.list – Vrátí seznam iBeaconů pro daný projekt.
    - Vstupní parametry:
        * projectId – ID projektu, pro který se mají vybrat iBeacony.
        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401. Pokud daný uživatel nemá oprávnění na prohlížení projektu, vrátí se chyba 403.
    - Vrácená hodnota:
        * List<IBeacon> – Seznam iBeaconů pro daný projekt.

- beacons.insert – Vytvoří nový iBeacon a přiřadí ho k patřičnému projektu.
    - Vstupní parametry:

        * IBeacon – IBeacon, který se má vytvořit. Pokud neexistuje projekt definovaný v tomto objetku jako projectId, vrátí se chyba 404.
        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401. Pokud daný uživatel nemá oprávnění na úpravu projektu, vrátí se chyba 403.
    - Vrácená hodnota:
        * IBeacon – Vytvořený iBeacon včetně nového ID.

- beacons.update – Upraví iBeacon projekt se zvoleným ID. Pokud iBeacon se zvoleným ID neexistuje nebo pokud projekt definovaný v IBeacon objektu jako projectId neexistuje, vrátí se chyba 404.
    - Vstupní parametry:
        * IBeacon – IBeacon, který se má upravit.
        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401. Pokud daný uživatel nemá oprávnění na úpravu projektu definovaném v IBeacon objektu, vrátí se chyba 403.
    - Vrácená hodnota:
        * IBeacon – Upravený iBeacon.

- beacons.delete – Smaže iBeacon se zvoleným ID.
    - Vstupní parametry:
        * id – ID iBeaconu, který se má smazat. Pokud iBeacon se zvoleným ID neexistuje, vrátí se chyba 404.
        * projectId – ID projektu, do kterého spadá daný iBeacon. Pokud projekt se zvoleným ID neexistuje, vrátí se chyba 404.
        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401. Pokud daný uživatel nemá oprávnění na úpravu projektu, vrátí se chyba 403.

- projects.collab.list – Vrátí seznam spolupracovníků k danému projektu včetně jejich oprávnění.
    - Vstupní parametry:
        * projectId – ID projektu, ke kterému se mají vrátit spolupracovníci. Pokud projekt neexistuje vrátí se chyba 404.

        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401. Pokud daný uživatel nemá oprávnění na správu spolupracovníků, vrátí se chyba 403.
    - Vrácená hodnota:
        * List<Permission> – Seznam spolupracovníků projektu včetně jejich oprávnění.

- projects.collab.set – Nastaví oprávnění pro daného uživatele k danému projektu.
    - Vstupní parametry:
        * projectId – ID projektu, ke kterému se nastavuje oprávnění. Pokud projekt neexistuje vrátí se chyba 404.
        * e-mail – Email uživatele, pro kterého se nastavuje oprávnění. Pokud uživatel neexistuje vrátí se chyba 404.
        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401. Pokud daný uživatel nemá oprávnění na správu uživatelů pro daný projekt nebo pokud je zvolený uživatel vlastníkem projektu, vrátí se chyba 403.

- projects.collab.remove – Odstraní spolupracovníka z daného projektu.
    - Vstupní parametry:
        * projectId – ID projektu, ze kterého se má odstranit spolupracovník. Pokud projekt se zvoleným ID neexistuje, vrátí se chyba 404.
        * e-mail – Email spolupracovníka, který se má z daného projektu odstranit. Pokud uživatel s daným e-mailem neexistuje, vrátí se chyba 404.
        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401. Pokud daný uživatel nemá oprávnění na správu uživatelů pro daný projekt nebo pokud je zvolený uživatel vlastníkem projektu, vrátí se chyba 403.

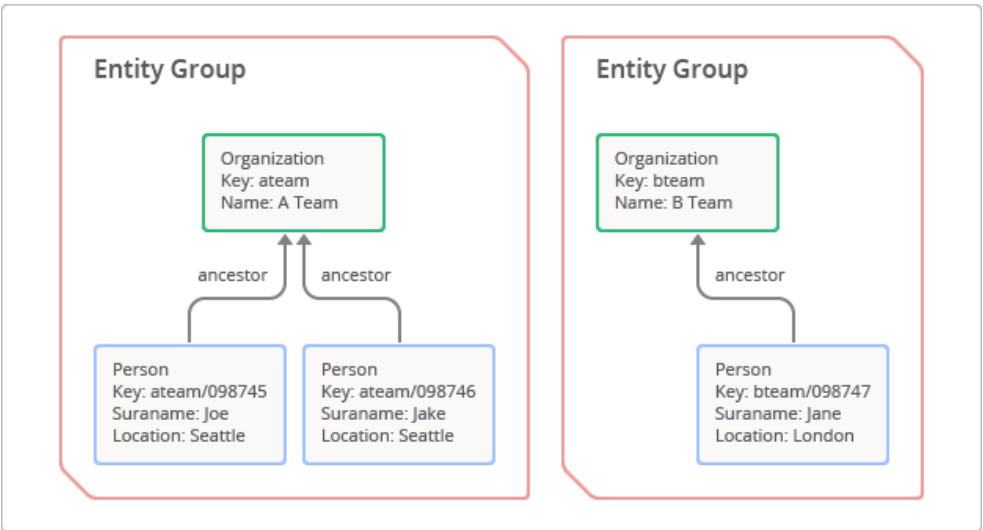
- projects.histogram – Vrátí histogram k danému projektu z daného časového úseku pro generování heatmapy.
    - Vstupní parametry:
        * projectId – ID projektu, ke kterému se má vrátit histogram. Pokud projekt se zvoleným ID neexistuje, vrátí se chyba 404.

        * from – Časové razítko začátku intervalu, pro který se má vrátit histogram.
        * to – Časové razítko konce intervalu, pro který se má vrátit histogram.
        * User – Objekt identifikující uživatele za použití OAuth 2.0. Pokud uživatel není veden v databázi, vrátí se chyba 401. Pokud daný uživatel nemá oprávnění na zobrazení daného projektu, vrátí se chyba 403.
    - Vrácená hodnota:
        * ApiHistogram – Objekt histogramu s informacemi o velikosti a umístění a dvourozměrné pole s hodnotami histogramu.

3.2.2 API klientské části

Knihovna starající se o měření polohy bude mít přístup k následujícímu API:

- client.beacons.list – Vrátí seznam iBeaconů s informacemi o jejich poloze.
    - Vstupní parametry:
        * projectId – ID projektu, ke kterému bude knihovna provádět měření. Pokud projekt se zvoleným ID neexistuje, vrátí se chyba 404.
    - Vrácená hodnota:
        * List<ApiBeaconPublic> – Seznam iBeaconů obsahující UUID, major, minor a polohu.
- client.project.get – Vrátí informace o daném projektu.
    - Vstupní parametry:
        * projectId – ID projektu, ke kterému bude knihovna provádět měření. Pokud projekt se zvoleným ID neexistuje, vrátí se chyba 404.
    - Vrácená hodnota:
        * ApiProjectPublic – Veřejné informace o projektu (od kdy do kdy má probíhat měření, plánek objektu a jeho umístění)
- client.records.insert – Uložení naměřených hodnot na server.
    - Vstupní parametry:
        * projectId – ID projektu, ke kterému bude knihovna provádět měření. Pokud projekt se zvoleným ID neexistuje, vrátí se chyba 404.
        * ApiRecords – Seznam naměřených hodnot (informace o zařízení, času a vypočítané poloze)

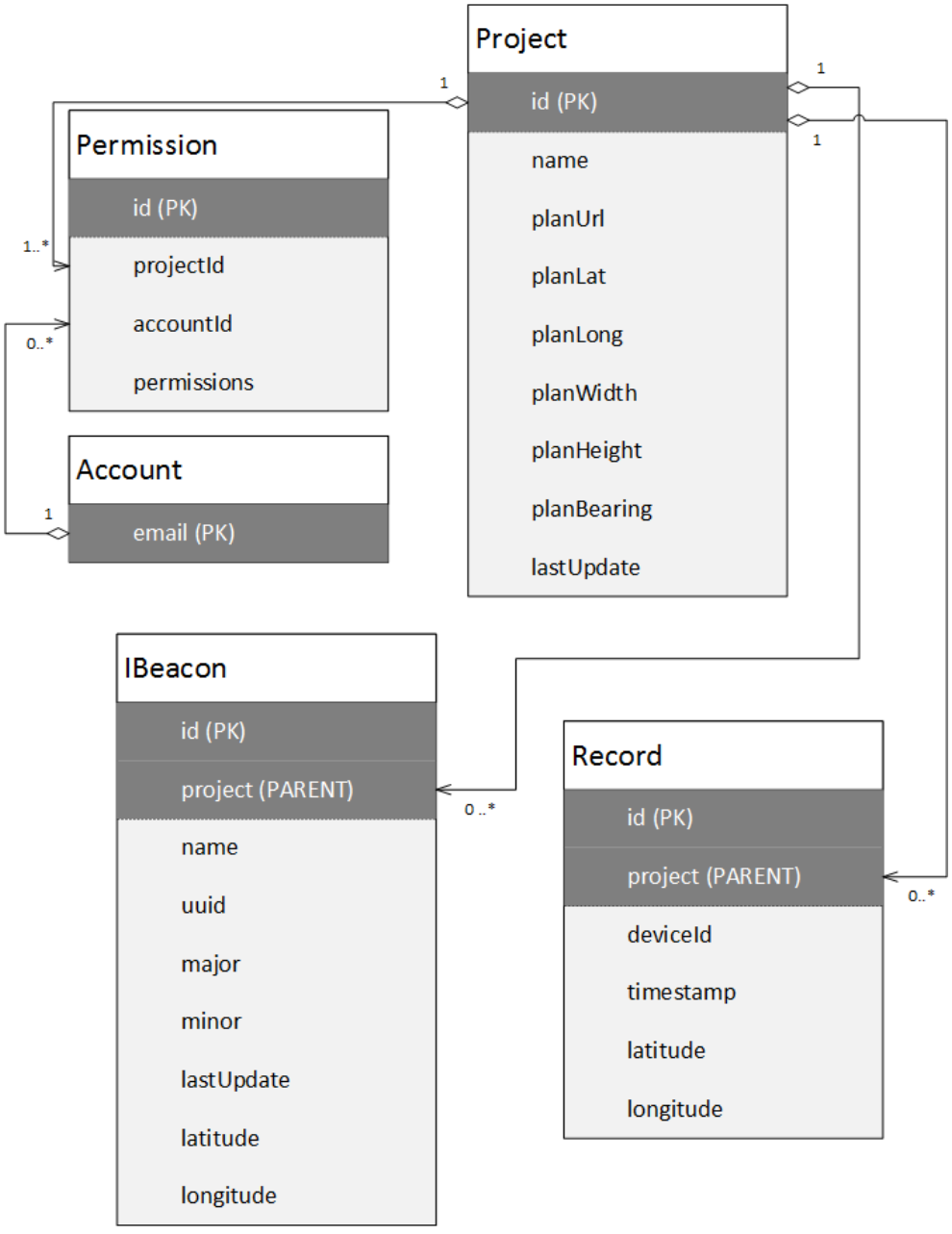


Obrázek 3.1: Eventuální konzistence V každé skupině, které jsou na obrázku znázorněny jako tým A a B, je zachována silná konzistence, transakcionalita (podpora principu ACID), fyzické umístění dat blízko sebe v lexikografickém pořadí podle klíčů, což má za následek minimální I/O při dotazování na předky. Zdroj: https://cloud.google.com/developers/articles/balancing-strong-and-eventual-consistency-with-google-cloud-datastore/

3.3 Databáze

Google App Engine pracuje s databází `DataStore` která je typu NoSQL. To mimo jiné znamená, že data mohou být rozdělena mezi více servery popř. datacentry. Tento přístup má výhodu v prakticky neomezené škálovatelnosti, avšak za cenu eventuální konzistence. Eventuální konzistence je teoretická garance toho, že pokud se nebudou provádět žádné další změny u dané entity, všechny dotazy budou časem vracet poslední zapsané hodnoty. Konzistenci lze ovlivnit hierarchickým návrhem databáze. Všechny entity se společným předkem mají garantovanou silnou konzistenci. Entity s jinými předky však mohou být umístěny v jiných datacentrech a tudíž mají garantovanou pouze eventuální konzistenci. Toto chování znázorňuje obrázek 3.1.

Na backendu bylo zvoleno hierarchické schéma databáze podle projektů. Jelikož projekty jsou na sobě nezávislé, není mezi nima vyžadována silná konzistence. Naopak záznamy měření a seznamy iBeaconů tuto konzistenci vyžadují. Proto je u těchto entit zvolen jako rodič právě Projekt. Databázové schéma je znázorněno na obrázku 3.2.



Obrázek 3.2: Databázové schéma backendu

KAPITOLA 4

Implementace

Tato kapitola popisuje vlastní tvorbu aplikace. Vzhledem k tomu, že zde není kvůli objemnosti vhodné popisovat celou implementaci, budou zde popsány klíčové části aplikace.

4.1 Specifika platformy

Pro pochopení architektury aplikace je potřeba znát aspoň základy Android frameworku a pochopit životní cyklus komponent. Proto jsou v následujících pár odstavcích tyto základy stručně popsané.

4.1.0.1 Application

Při spuštění aplikace se vytvoří instance třídy `Application`, která je společná pro všechny komponenty ve stejném procesu. K instanci je pak možné přistupovat ze všech komponent pomocí `getApplicationContext()`. Vývojář může použít vlastní implementaci přidáním odkazu na svoji třídu do `application` tagu v Android Manifestu: `android:name=".App"`. Následně se jako první při spuštění aplikace vytvoří instance právě této třídy, ve které se můžou inicializovat data potřebná napříč komponentami aplikace v metodě `onCreate()`. V případě komponent běžících v jiném procesu se vytvoří instance této třídy pro každý proces zvlášť.

4.1.0.2 Activity

Aktivita je komponenta aplikace, se kterou může uživatel interagovat a na kterou se vykresluje UI. Aplikace se většinou skládá z více aktivit, přičemž jedna je typicky hlavní „MainActivity“, která se zobrazí, když uživatel spustí aplikaci z launcheru. Když se spustí nová aktivita, stávající se pozastaví a

zůstane v paměti, dokud systém nepotřebuje uvolnit prostředky. Zároveň se uloží daná aktivita a její nastavení do zásobníku aktivit, který se zachová i když systém prostředky uvolní a aktivity ukončí. Při každém přechodu mezi stavy životního cyklu se volá příslušná metoda, přičemž jakmile je aktivita ve stavu pozastavení (zavoláno `onPause()`), systém ji může kdykoli ukončit kvůli potřebě uvolnit prostředky a další metody již nemusí nastat. Životní cyklus je znázorněn na obrázku 4.1. [16]

Je taky dobré vědět, že při změně konfigurace (otočení displeje, vysunutí HW klávesnice, změny jazyku...) se aktivita restartuje. Je tedy na vývojáři, aby si toto ohlídal a potřebná data předal ze staré do nové instance.

4.1.0.3 Fragment

Fragment reprezentuje chování uživatelského rozhraní aktivity nebo její části. Na jednu aktivitu je možné umístit více fragmentů, zároveň je možné jeden fragment znovu-použít ve více aktivitách. Fragment je možné považovat za modulární sekci aktivity, která má vlastní životní cyklus (viz 4.2), obdržuje vlastní události uživatelského vstupu, a která může být přidána nebo odstraněna za běhu aktivity.

Fragment musí být vložen v aktivitě a jeho životní cyklus je ovlivněn životním cyklem hostující aktivity. Pokud je například aktivita pozastavena, jsou pozastaveny všechny její fragmenty, pokud je ukončena, jsou i všechny její fragmenty s jednou výjimkou. Pokud dojde ke změně konfigurace a aktivita se restartuje, je možné nastavit fragmentu, aby se zachoval a předal do nové instance aktivity bez jeho restartování. Toto se prování zavoláním `setRetainInstance(true)`.

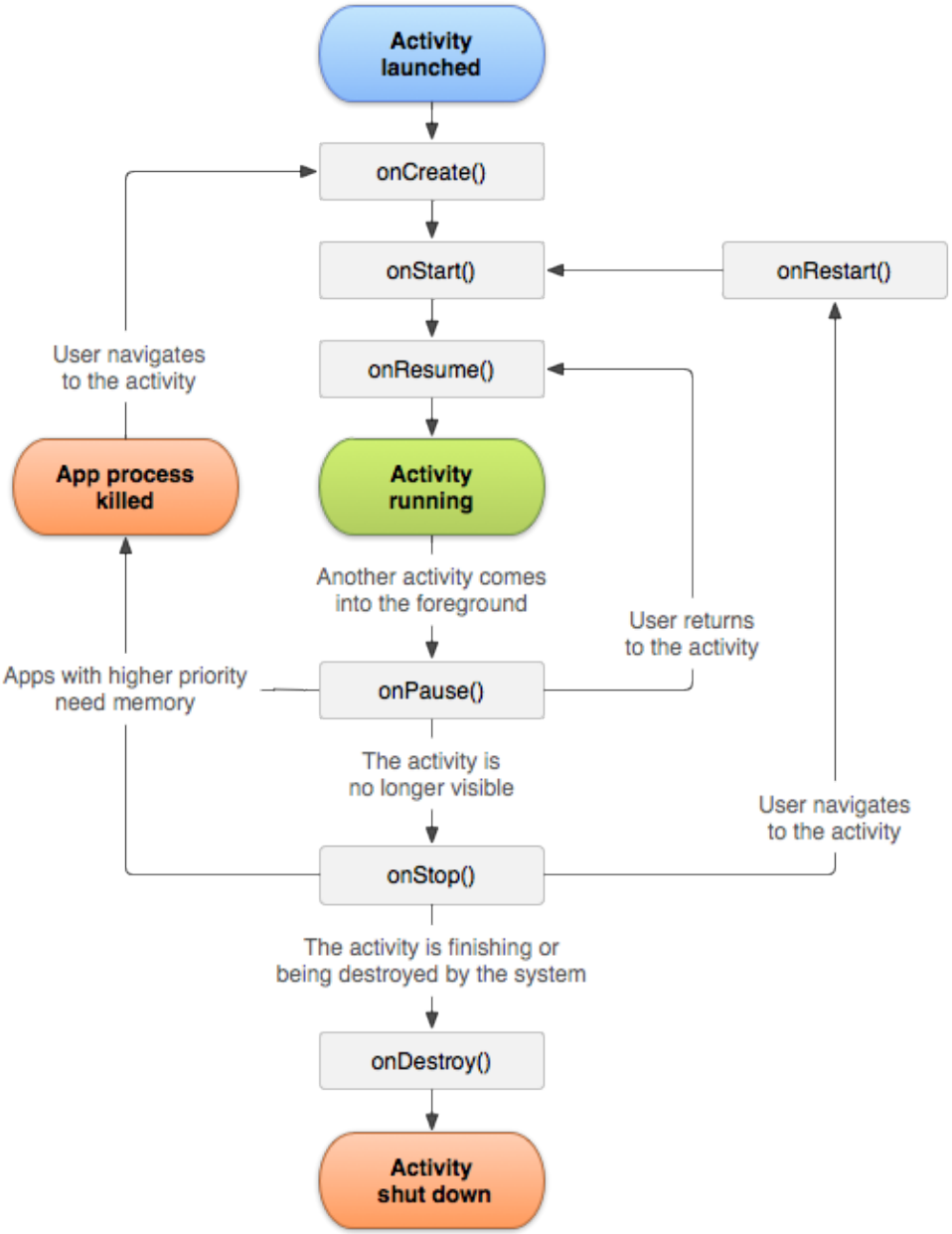
Jakmile však aktivita běží (je ve stavu mezi `onResume()` a `onPause()`), je možné manipulovat s jejími fragmenty nezávisle. Změny fragmentů je možné uchovávat v podobném zásobníku jako zásobník aktivit, který podobně jako zásobník aktivit zůstane zachován i při ukončení aktivity v důsledku uvolňování prostředků. [18]

4.1.0.4 View

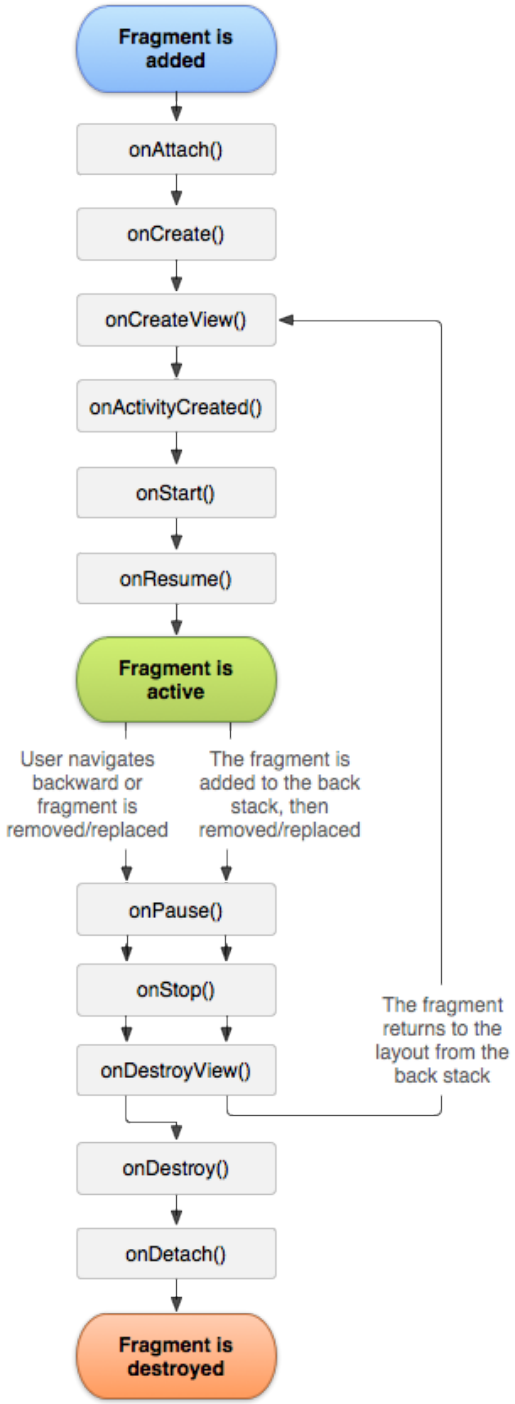
View reprezentuje základní prvek na stavbu UI. View okupuje obdélníkovou oblast obrazovky a je odpovědné za vykreslování a řízení událostí pro tuto oblast. View je také základní třídou pro všechny UI prvky obrazovky. [21]

4.1.0.5 Process

Každá aplikace běží pod nějakým procesem. Ve výchozím stavu si každá aplikace zakládá vlastní proces se jménem ID aplikace. Toto chování je však možné ovlivnit a je možné nastavit aplikaci nebo její komponentě, aby běžela v jiném, již existujícím procesu nebo aby si vytvořila pro své komponenty procesů více. Toto se nastavuje v Android Manifest souboru přidáním parametru



Obrázek 4.1: Životní cyklus aktivity
Zdroj: http://developer.android.com/guide/components/activities.html



Obrázek 4.2: Životní cyklus fragmentu
Zdroj: http://developer.android.com/guide/components/fragments.html

`process="jménoProcesu"`. Pokud zde jméno procesu začíná symbolem „:“, je vytvořen nový proces se jménem `idAplikace:jmenoProcesu`, pokud začíná malým písmenem, spustí se daná komponenta v již existujícím procesu s daným jménem. Každý proces má svou prioritu, kterou ovlivňují komponenty běžící v daném procesu. Pokud je v daném procesu Aktivita na popředí, má prioritu nejvyšší. Pokud v něm běží služba s vysokou prioritou, opět se zvýší i priorita procesu. V případě nedostatku prostředků pak systém zabije procesy s nízkou prioritou. Procesy nesdílí žádné proměnné a tak je nutné k mezi-procesové komunikaci používat tzv. AIDL, popř. Broadcasty.

4.1.0.6 Service

Služba je komponenta aplikace sloužící pro dlouho běžící operace na pozadí, která neposkytuje uživatelské rozhraní. Službu může spustit jiná komponenta aplikace a bude běžet na pozadí i když uživatel přejde do jiné aplikace. [20]. Na službu je možné se napojit a komunikovat prostřednictvím `ServiceConnection`. Pokud služba běží ve stejném procesu, jako zbytek aplikace, je možné komunikovat i přes společný context aplikace popř. přes statické proměnné. Pokud však služba běží v jiném procesu, je nutné se na ni napojit.

Pokud se systém rozhodne, že má nedostatek prostředků, může službu zabít. Jakmile se uvolní dostatečné prostředky, systém ji znovu spustí. Toto chování je možné ovlivnit návratovou hodnotou v metodě `onStartCommand()`, která se spustí pro každé volání `startService()` v dané službě. Je možné definovat, aby se služba znovu po zabití nespouštěla, aby se spustila i s parametry pro veškeré nedoběhnuté úkoly nebo aby se spustila bez spouštěcích parametrů.

Kromě standardní `Service`, ve které si vývojář musí sám hlídat, aby složité operace prováděl ve vlákně na pozadí a aby po skončení operace službu zastavil zavoláním příkazu `stopSelf()`, existuje `IntentService`, která tyto problémy řeší automaticky. V `IntentService` je potřeba přepsat metodu `onHandleIntent(Intent)`, která se zavolá pro každý spouštěcí request ve vlákně na pozadí. Zároveň je zaručeno, že v případě více requestů se zpracovávají sériově – nejprve se provede první a až jakmile `onHandleIntent(Intent)` u prvního requestu skončí, zavolá se tato metoda pro druhý request. Po zpracování všech requestů se služba automaticky zastaví.

4.1.0.7 ContentProvider

Tato komponenta se stará o přístup ke strukturovaným datům. Nejběžnější použití implementuje přístup k databázi (pro tento účel jsou také navrženy metody), nicméně je možné zvolit libovolný způsob samotného ukládání dat (třeba do volatilní paměti). Vývojář má možnost implementovat metody pro vkládání, mazání a upravování záznamů a také pro dotazování se na ně. ContentManager je jediná komponenta, která se inicializuje ještě před samotnou Application třídou. Přistup z ostatních komponent je nepřímý pomocí volání

`getContentResolver()`, při kterým se předává URI, pomocí kterého systém rozhodne, jaký `ContentProvider` se použije. Díky tomu je možné ke Content provideru přistupovat z libovolného procesu a vývojář se nemusí starat o meziprocesorovou komunikaci. Ve výchozím stavu se `ContentProvider` inicializuje v hlavním procesu aplikaci, nicméně je možné povolit inicializaci pro každý proces zvlášť. Pokud se povolí přístup v AndroidManifestu, je možné otevřít `ContentProvider` i pro přístup z jiných aplikací.

4.1.0.8 ContentResolver

Content resolver je systémová komponenta sloužící k přístupu k výše zmíněnému `ContentProvideru`. Kromě samotného přístupu k datům umožňuje nastavit i notifikační URI, pomocí kterého je možné notifikovat zaregistrované observery o změnách v datech.

4.1.0.9 SharedPreferences

Shared preferences slouží k perzistentnímu ukládání hodnot. K hodnotám se přistupuje pomocí klíčů typu `String`. Jako hodnoty jsou podporovány primitivní datové typy, `String` a `Set<String>`. Hodnoty jsou ukládány ve formátu XML do interní paměti zařízení. K preferencím se přistupuje zavoláním metody `context.getSharedPreferences()` z jakékoli komponenty aplikace. Při prvním volání této metody se provede načtení hodnot z interní paměti do operační a poté si je v ní systém pro optimalizaci udržuje. SharedPreferences je možné provozovat i mezi více procesy, nicméně je nutné předat do metody `getSharedPreferences` parametr `MODE_MULTI_PROCESS`, který při volání této metody provede znovunačtení dat, pokud došlo k nějaké změně.

4.1.0.10 Loader

Jelikož náročné operace nemohou probíhat na UI vlákně a je tedy nutné řídit asynchronní operace, poskytuje Android framework loadery, které umožňují jednoduché asynchronní načítání dat do aktivit nebo fragmentů. O řízení loaderů se stará třída `LoaderManager`, která zajišťuje také to, že výsledek asynchronní operace se danému fragmentu nebo aktivitě předá, až když je na popředí. Zároveň se stará o situace, kdy se aktivita nebo fragment restartuje v důsledku změny konfigurace po spuštění asynchronního načítání a před tím, než se načítání dokončí. Loader automaticky podporuje znovunačtení dat, pokud dojde ke změně (notifikace na URI k danému obsahu pomocí content resolveru).[19]

4.1.0.11 Resources

Zdroje, jako jsou texty a obrázky, by vždy měly být odděleny od zdrojového kódu. Pro takový účel slouží v projektu složka `/res`, kam se tyto zdroje dávají.

Vývojář k nim pak může přistupovat ze zdrojového kódu pomocí automaticky generované třídy `R`, které poskytuje unikátní id pro každý zdroj v této složce. Pokud se například uloží obrázek `image.png` do složky `/res/drawables`, bude jeho id v `R.drawable.image`. Android framework pak nabízí metody pro získání textu, obrázku, animace a dalších podporovaných typů, kterým se předá právě toto id.

4.1.0.12 Broadcast Receiver

Pokud potřebuje aplikace reagovat na chování systému, je možné odchytávat některé události, k čemuž slouží `BroadcastReceiver`. Pokud se například v zařízení zapne Bluetooth, systém vytvoří událost, která se pošle všem aplikacím, které mají broadcast receiver zaregistrovaný pro tento typ události. Tato registrace se provádí definováním tzv „intent filteru“ v `AndroidManifest`.

4.1.0.13 Alarm

Ke spouštění akcí v určitý čas slouží tzv. Alarmy, které umožňují spouštět jednorázově i opakovaně komponenty aplikace. Jelikož jsou alarmy řízeny přímo systémem, je jejich spuštění nezávislé na aktuálním stavu aplikace (spustí se i když je aplikace kompletně vypnutá). Zároveň je tímto způsobem možné probudit zařízení z hlubokého spánku, pokud je potřeba.

4.1.0.14 AndroidManifest

Tento soubor slouží k definování veškerých komponent aplikace, jaká oprávnění daná aplikace požaduje a na jaké zařízení může být nainstalována. Dále jsou zde definována veškerá metadata spolu s verzí aplikace a jménem balíčku, který musí být unikátní pro každou aplikaci.

4.1.0.15 Log

U desktopových programů slouží k vypisování debugovacích zpráv příkazový řádek. U mobilního zařízení ale takový prvek chybí. K tomuto účelu právě slouží statická třída `Log`, která navíc umožňuje vypisování zpráv v několika stupních důležitosti. Vývojář si pak může pouze výpisy s nastaveným stupněm `Error`, `Warning` a vyšší, `Info` a vyšší, `Debug` a vyšší nebo `Verbose` a vyšší.

Pro vypsání záznamu v daném stupni slouží metody `Log.v`, `Log.d`, `Log.i`, `Log.w` a `Log.e`. Tyto různé stupně jsou také pro přehlednost barevně odlišeny. Dále je možné ke každému záznamu pro ještě lepší filtraci přiřadit tzv TAG. Kromě výše zmíněných filtrů lze zprávy ještě třídit podle PID a jména balíčku aplikace. Záznamy si pak vývojář může zobrazit přehledně pomocí Android Studia popř. nástroje ADB.

4.2 Architektura aplikace

Projekt je rozdělen na tyto moduly:

- `btscanner` – Knihovna starající se o scanování iBeaconů.
- `loccalc` – Knihovna s algoritmy pro výpočet polohy z naměřených iBeaconů.
- `library` – Knihovna starající se o zaznamenávání polohy za využití výše zmíněné knihovny `btscanner` a `loccalc` a o její následné synchronizaci se serverem.
- `demo` – Demo aplikace znázorňující možnosti implementace knihovny.
- `admin` – Administrátorská aplikace využívající knihovnu `btscanner`.
- `backend` – Appengine server starající se o ukládání dat a výpočet hodnot pro heatmapu.

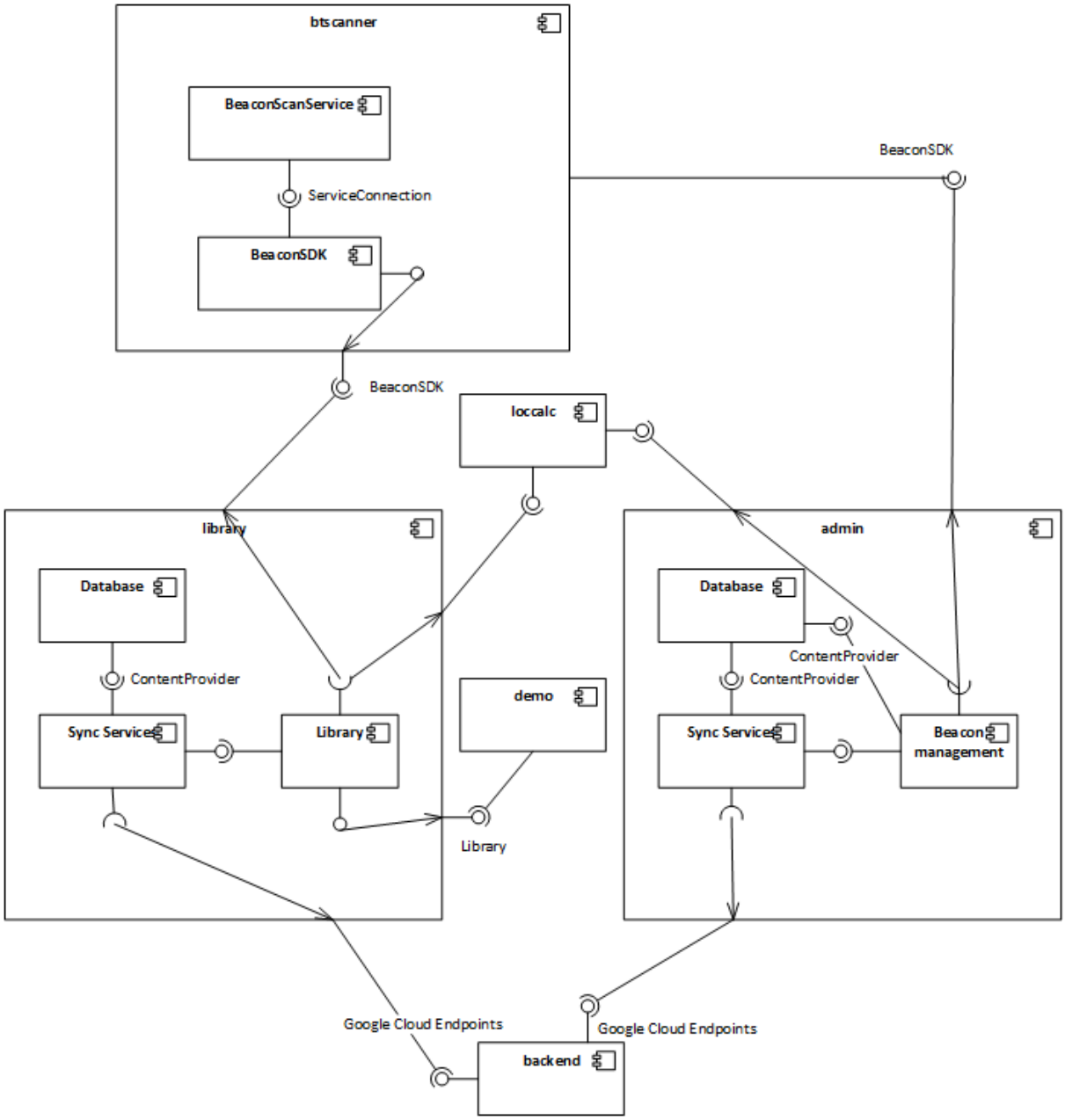
Nejdřív zde bude popsán modul `btscanner` starající se o monitorování okolních iBeaconů, poté budou popsány algoritmy v modulu `loccalc`, následně bude rozebrána samotné knihovna `library`, která bude prezentována vývojáři jako SDK pro použití našeho analytického systému a nakonec bude popsána implementace administrátorské a serverové části. Propojení těchto modulů je znázorněno v komponent diagramu 4.3.

4.2.1 Modul btscanner

Tato knihovna provádí scanování okolních iBeaconů a výpočet jejich vzdálenosti. Použití této knihovny bylo navrženo v souladu s moderním trendem řetězení příkazů. Následující kód spustí scanování iBeaconů spadajících do námi specifikovaných regionů a notifikuje `listener` o naměřených vzdálenostech:

```
BeaconSDK.with(mContext)
        .enableRanging(listener)
        .startScanning(regions);
```

Jako vstupní bod slouží třída `BeaconSDK`, ve které je logika pro zapnutí, vypnutí scanování, nastavení parametrů a pro notifikaci zaregistrovaných observerů o nascanovaných iBeaconech. Z `BeaconSDK` se pak spouští služba `BeaconScanService`, ve které probíhá scanování. Aby systém nedržel zbytečně v paměti celou aplikaci a aby bylo scanování nezávislé na zbytku aplikace, bylo zvoleno spouštění této služby v novém procesu `:scan`. `BeaconSDK` se na tuto



Obrázek 4.3: Diagram komponent projektu

službu napojí a pomocí AIDL mezi sebou tyto 2 komponenty komunikují. Jelikož systém vytváří instanci Application, ze které se inicializuje scanování, pro každý proces zvlášť, bylo nutné implementovat 2 rozdílné verze `BeaconSDK`, aby se neprováděla veškerá logika 2x. Tím vznikly třídy `MainBeaconSDK`, která provádí veškerou business logiku z hlavního procesu a třída `RemoteBeaconSDK`, která běží spolu se službou v procesu `:scan`. Ta má za úkol pouze probudit hlavní proces v případě, že je zařízení v dosahu námi specifikovaných iBeaconů. Toto rozvržení umožňuje scanování pouze v `:scan` procesu, pokud není zařízení v dosahu námi požadovaných iBeaconů s tím, že hlavní proces může být v rámci úspory prostředků vypnutý. Zároveň nám ale toto řešení umožní provádět veškerou business logiku v hlavním procesu, který se automaticky spustí, jakmile se zařízení dostane do blízkosti oněch iBeaconů, což ulehčí práci vývojáři, jelikož nemusí řešit meziprocesovou komunikaci. Pro vývojáře je použití těchto rozdílných implementací naprosto transparentní a callbacky o nascanovaných iBeaconech bude dostávat pouze v instanci v hlavním procesu, který se v blízkosti iBeaconů spustí automaticky.

Scanování je prováděno v cyklech. To znamená, že určitou dobu probíhá scanování a poté se jinou dobu čeká. Tyto intervaly lze nastavit v `BeaconSDK`. Stará se o to třída `CycledLeScanner`, která běží ve službě `BeaconScanService`. U zařízení s Android OS >= 5.0 se používá nové API umožňující nastavení priority a filtrů v třídě `CycledLeScanner21` a u starších zařízení se používá implementace v třídě `CycledLeScanner18`, kde se musí provést filtrace iBeaconů až dodatečně.

Za pomoci BroadcastReceiverů se `BeaconScanService` spustí automaticky, pokud dojde k zapnutí Bluetooth nebo k restartování telefonu. Toto chování je možné ovlivnit v BeaconSDK pomocí metody `setAutoStartEnabled`. Služba je navržena tak, aby se v případě zabití systémem automaticky znovuspustila a pokračovala ve scanování iBeaconů. Toho je docíleno nastavením Alarmu, který spustí service popř. probudí zařízení, pokud se v definovaným čase nespustí znovu scanovací cyklus.

Služba se také stará o detekci viditelnosti iBeaconů pomocí `BeaconMonitor` a počítání vzdálenosti od viditelných iBeaconů pomocí `BeaconRanger`. Detekce viditelnosti má zpravidla větší okno, kdy je iBeacon viditelný aby se zamezilo neustálému přeskakování mezi stavy v případě, že se v průběhu 1 cyklu daný iBeacon nenascanuje, popř. pokud zařízení přejde do hlubokého spánku. To pak k dalšímu scanovacímu cyklu může dojít později, než normálně.

Měření vzdálenosti probíhá na základě naměřené síly signálu (rssi) a vysílací síly `txPower`, která se posílá v iBeacon packetu (viz. 2.2). `txPower` udává zkalibrovanou sílu signálu (rssi) při vzdálenosti 1 metr. Aktuální vzdálenost iBeaconu od zařízení se vypočítá následujícím algoritmem:

```
double ratio = rssi * 1.0 / txPower;
if (ratio < 1.0) {
    return Math.pow(ratio, 10);
} else {
    return (0.89976) * Math.pow(ratio, 7.7095) + 0.111;
}
```

Konstanty 0.89976, 7.7095 a 0.111 byly vypočítány prokládáním křivky na vzorku naměřených rssi při známých vzdálenostech. [8]

Vzhledem k tomu, že síla signálu velmi kolísá, bylo zvoleno použití klouzavého průměru pro uhlazení oněch výkyvů. Ve výchozím stavu se vzdálenost počítá z posledních 10 měření za posledních 20s. Tyto parametry má ovšem vývojář možnost změnit.

4.2.2 Modul loccals

Tato knihovna obsahuje algoritmy pro výpočet polohy zařízení za předpokladu, že jsou k dispozici souřadnice nascanovaných iBeaconů a jejich vzdálenosti.

4.2.2.1 Vážený průměr

Nejjednodušší metoda na výpočet aktuální polohy ze signálu je vážený průměr polohy jednotlivých iBeaconů. Jako váhu zvolíme převrácenou hodnotu vzdálenosti, kterou pro daný iBeacon zařízení naměří.

Na výpočet aktuální polohy jsme dále zkusili použít také minimalizační metody.

4.2.2.2 Minimalizační metody

Jako první jsme zkusili minimalizovat funkci $\sum_i (r_i - d_i)^2$, kde r_i je vzdálenost iBeacony od polohy, kterou počítáme a d_i je naměřená vzdálenost. Minimalizací této funkce dostaneme výslednou polohu v průniku kružnic se středem v poloze iBeacony a poloměrem d_i, pokud tento průnik není prázdný. Při použití právě tří iBeaconů se jedná o metodu trilaterace. Tato metoda funguje dobře, jestliže iBeacony dávají hodnoty s malou chybou. Naopak jediné měření s velkou chybou způsobí výrazný posun řešení. Jelikož signál iBeaconů obecně velmi kolísá (viz tabulka 5.1) a jeho sílu ovlivňují také překážky v prostoru, v naměřených vzdálenostech se mohou vyskytnout i velmi odlehlé hodnoty. Proto je tato metoda pro výpočet optimální polohy nevhodná.

Zkusili jsme tedy použít robustnější metodu za použití optimalizační funkce $\sum_i \left(1 - \frac{r_i}{d_i}\right)^2$, kde r_i je vzdálenost iBeacony od polohy, kterou počítáme a d_i

je naměřená vzdálenost. V ideálním případě, kdy naměřená vzdálenost bude odpovídat vzdálenosti od vypočítané polohy, nabude zlomek ($\frac{r_i}{d_i}$) hodnoty 1, celá optimalizační funkce dá hodnotu 0, a metoda tak najde přesné řešení. Na rozdíl od trilaterace však nemají u této metody odlehlé hodnoty takový vliv na výsledné řešení, a proto je pro použití s iBeacony vhodnější.

Do výsledného algoritmu jsme aplikovali tuto metodu, jelikož byla u ní vypozorována nejmenší chyba. K nalezení optima byl použit algoritmus Levenberg–Marquardt [3]. U našeho testovacího projektu se chyba u této metody pohybovala okolo 2m. Pro srovnání metoda váženého průměru měla průměrnou chybu o 30% větší. Srovnávací video těchto metod je součástí přiloženého DVD.

4.2.3 Modul library

Tato knihovna je napojena na backend a stará se o zaznamenávání polohy pro daný projekt. Použití této knihovny bylo opět navrženo v souladu s moderním trendem řetězení příkazů. Následující kód inicializuje zaznamenávání polohy pro námi specifikovaný `PROJECT_ID` – stáhne aktuální info o projektu, seznam rozmístěných iBeaconů s jejich polohou a poté spustí scanování:

```
Library.with(mContext).setProjectId(PROJECT_ID).start();
```

Knihovna používá modul `btscanner`, ve kterém zapne měření vzdálenosti a ve výchozím stavu jednou za 60s uloží aktuální polohu, pokud je v dosahu zadaných iBeaconů. Knihovna se vždy při inicializaci pokusí stáhnout aktuální info o projektu spolu s iBeacony a uloží tato data do databáze. Pokud stažení selže, použije se nacachovaná verze z databáze. Pokud ani v databázi žádný záznam neexistuje, scanování se nespustí.

Kromě zaznamenávání polohy a její synchronizace umožňuje knihovna také implementovat do cílové aplikace aktuální polohu rozpoznanou z nascanovaných iBeaconů. Tuto funkčnost poskytuje metoda `setLocationListener`, po jejímž nastavení se předaný listener zavolá pokaždé, když je dostupná nová poloha.

Přístup do databáze je řešen pomocí ContentProvideru, což odděluje samotnou databázi od zbytku aplikace a umožňuje do ní přístup z jakékoli komponenty bez nutnosti si udržovat její instanci za pomoci systémové metody `Context.getContentResolver()`. ContentProvider vyžaduje použití authority, která musí být v systému unikátní (systém nepovolí nainstalovat více aplikací se stejnou authoritou ContentProvideru). Proto bývá zvykem do authority zakomponovat ID aplikace. Jelikož se používá ContentProvider v knihovně, která při kompilaci nezná ID aplikace, je potřeba, aby výsledná aplikace vložila do AndroidManifestu následující kód:

```
<provider
    android:name="cz.mpelant.ibloc.library.db.DatabaseProvider"
    android:authorities="${applicationId}.iblocprovider"
    android:enabled="true"
    android:exported="false" />
```

Manifest merger v Gradle build systému pak nahradí `${applicationId}` za skutečné ID aplikace. V kódu je však potřeba výslednou authoritu také zjistit a to se dá provést pouze za běhu aplikace:

```
AUTHORITY = context.getPackageName() + ".iblocprovider";
```

Abychom potřebné statické proměnné inicializovali dřív, než k nim budeme ze zbytku knihovny přistupovat, bylo nutné tuto inicializaci provést v `onCreate` metodě v ContentProvideru, jelikož `ContentProvider` se inicializuje jako první ještě před samotnou `Application` třídou.

Synchronizace naměřených záznamů o poloze na server probíhá automaticky, jakmile se překročí limit, který je ve výchozím stavu nastaven na 20 záznamů. Je to z důvodů úspory energie, neboť odesílání dat najednou spotřebovává méně energie než kdyby se stejné množství dat odesílalo po kouskách. To je způsobeno vysokou režií při inicializaci antény pro vysílání a přijímání, která se při společném posílání dat může provést pouze jednou.

4.2.4 Modul demo

Jedná se o jednoduchou aplikaci ukazující možnosti implementace naší knihovny. Inicializace se provádí ve třídě `App`, což je rozšíření systémové třídy `Application`. Aby se použila tato třída, je nutné v AndroidManifestu nastavit u tagu `application` atribut `android:name=".App"`. Z menu hlavní obrazovky je možné spustit aktivitu zobrazující na mapě plánek objektu a aktuální polohu získanou pomocí naší knihovny z okolních iBeaconů. Plánek se vkládá na Google mapy jako `GroundOverlay`, což umožňuje mapu libovolně natáčet, posouvat přibližovat atd. při zachování správného umístění plánku. URL plánku spolu s potřebnými parametry jako šířka a délka objektu, pozice středu a rotace jsou staženy z backendu pomocí `client.project.get`. Aktuální poloha je pak nastavena Google mapám metodou `setLocationSource`.

4.2.5 Modul admin

Administrátorská aplikace umožňuje spravovat projekty, nastavovat rozmístění iBeaconů a zobrazovat si pro daný projekt výslednou heatmapu návštěvnosti.

4.2.5.1 Načítání dat

Pro autorizaci se používá OAuth 2.0, jehož plnou podporu přináší knihovna Google Play Services. Je potřeba z Google Developer console získat „Client ID for web application“, které se použije v aplikaci jako audience:

```
GoogleAccountCredential.usingAudience(
    context,
    "server:client_id:" + Constants.WEB_CLIENT_ID);
```

Tím se získá instance `GoogleAccountCredential`, která umožňuje zavoláním metody `newChooseAccountIntent` zobrazit dialog pro výběr účtu, se kterým se chce uživatel autentikovat. Jelikož nám stačí pro autentikaci znát pouze e-mailovou adresu, není potřeba vyžadovat po uživateli žádné další oprávnění. Zvolený účet se poté uloží zavoláním metody `setSelectedAccountName` v `GoogleAccountCredential`. Instanci `GoogleAccountCredential` se zvoleným účtem poté použijeme ve vygenerované knihovně starající se o komunikaci s naším API na backendu následovně:

```
new Ibloc.Builder(AndroidHttp.newCompatibleTransport(),
    new AndroidJsonFactory(), mGoogleAccountCredential)
    .setRootUrl(SERVER_URL)
    .build();
```

Tento kód nám vrátí instanci třídy `Ibloc`, která v sobě má naimplementovány metody pro komunikaci s naším API nadefinovaném v `backend` modulu a která se automaticky postará o OAauth 2.0 autentikaci s námi zvoleným účtem u metod, které to vyžadují.

Seznam projektů se tak například získá takto:

```
mIbloc.projects().list().execute().getItems();
```

Pokud server vrátí aplikační chybu, je možné ji zpracovat odchycením výjimky `GoogleJsonResponseException`, kde je možné získat kód chyby a její popis. Pokud však jde o chybu přenosu dat nebo o chybu, která je mimo naší aplikaci, vyhodí volání klasickou `IOException`.

Veškeré volání našeho API je schované ve třídě Api. Aplikace má 3 přístupy k datům:

- Cachovaná data – Seznam projektů a rozmístění jejich iBeaconů jsou uloženy v lokální databázi pro rychlý přístup při slabém nebo žádném internetovém připojení. Data se aktualizují pomocí dedikovaných služeb a přistupuje se k nim pomocí ContentProvideru `DatabaseProvider`. Změny se provádí nejdřív lokálně a poté se spustí patřičná synchronizační služba a provede dané změny i na backendu. Každá entita dědí od `AbstractEntity`, která kromě jiného má parametr udávající stav entity. Pokud je na zařízení verze stažená ze serveru bez provedených změn, má entita stav `CLEAN` a časové razítko stažené z backendu. Pokud se provede nějaká změna v entitě, popř. se vytvoří nová entita, změní se stav na `MODIFIED` a časové razítko se nastaví na aktuální čas. Pokud se entita smaže, nastaví se stav na `DELETED`. Při synchronizaci dat se pak nejdříve synchronizují změněné položky (zavolá se create, update nebo delete API pro danou entitu a uloží se patřičný výsledek do databáze) a poté se stáhnou aktuální hodnoty ze serveru. Backend sám podle časového razítka rozpozná, zda-li má danou změnu provést nebo ignorovat. V obrazovkách se k těmto datům přistupuje přes `CursorLoader`, který v případě změn provede requery a tím pádem se zobrazují vždy aktuální data.

- Online data – Pro seznam spolupracovníků byl zvolen online přístup. K datům se opět přistupuje přes ContentProvider, avšak implementaci nyní zastupuje `OnlineProvider` a podporuje pouze dotazování. V tomto ContentProvideru se při dotazu provede stažení dat ze serveru. Data se poté vloží do MatrixCursoru, který metoda query vrátí. V obrazovkách se k těmto datům přistupuje úplně stejně jako k datům z databáze přes `CursorLoader`, který si data ukládá do volatilní paměti po dobu zobrazení dané obrazovky. Provedené změny se rovnou předají patřičné službě, která je promítne na server a poté notifikuje ContentResolver o dané změně. Systém pak restartuje loader který přes `OnlineProvider` provede znovu dotaz na server a zobrazí se aktuální data.

- Preference – Pro uložení vybraného uživatele a dalších perzistentních nastavení se používají `SharedPreferences`.

4.2.5.2 Systém obrazovek

Pro systém obrazovek bylo zvoleno použití fragmentů s jednou hlavní Aktivitou `MainActivity`. Jedinou výjimkou je `PlaceMapActivity`, která u tabletů obsahuje více obrazovek najednou (mapa s rozmístěním iBeaconů, seznam viditelných iBeaconů) a u telefonů zobrazuje pouze jednu (mapa). Logika ostatních obrazovek je však primárně implementována ve Fragmentech. V hlavní aktivitě `MainActivity` je pak ještě implementován výběr uživatele.

Jelikož se Activity i Fragmenty restartují při rotaci displeje, popř. je systém může zabít při minimalizaci aplikace a poté se musejí znovuvytvořit, je třeba u těchto tříd uchovávat potřebné parametry. Existuje několik možností, jak se

s tímto problémem vypořádat, u této aplikace se převážně požívají argumenty fragmentu. K těm se přistupuje pomocí metody `getArguments` u fragmentu a jsou typu `Bundle`, umožňující mapování hodnot na klíč. Tyto argumenty pak systém v případě nutnosti uloží do nevolatilní paměti a po restartování fragmentu je z ní znovu načte.

Nastavení pozice plánku se provádí v `PlaceMapActivity`. Plánek se při tom zobrazí poloprůhledný přes celou obrazovku a pod ním je zobrazena Google mapa, se kterou lze pomocí gest manipulovat. Uživatel si tak může napozicovat mapu, aby plánek překrýval zvolené místo. Poté se získají souřadnice rohů obrázku na mapě pomocí metody `map.getProjection()` `.fromScreenLocation(Point)` a vypočítá se střed, šířka a výška v metrech. Zároveň se pomocí `map.getCameraPosition().bearing` získá natočení mapy. Tyto 4 údaje se uloží k danému projektu. Jsou totiž potřeba k zobrazení vrstvy s plánkem na dané pozici na Google mapách.

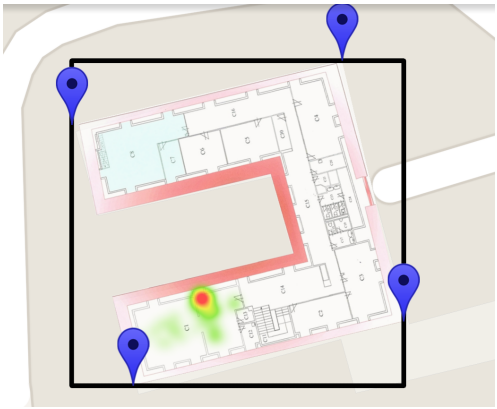
`PlaceMapActivity` také umožňuje nastavit rozmístění iBeaconů. Ty jsou řešeny pomocí pinů na Google mapě. Po kliknutí na pin se zobrazí okénko s informacemi (jméno, UUID, major, minor). Po kliknutí na toto okénko je možné daný iBeacon odstranit, přesunout nebo přejmenovat. Přesunutí probíhá intuitivně přetažením pinu na zvolenou pozici. Pro přidání nového iBeaconu se zobrazí dedikovaná obrazovka `SelectBeaconFragment`, která spustí scanování všech okolních iBeaconů (bez použití filtru) pomocí modulu `btscanner`. U tabletů se tento fragment zobrazuje na obrazovce s mapu pro rychlejší interakci.

4.2.5.3 Zobrazení heatmapy

Zobrazení heatmapy pro daný projekt probíhá ve fragmentu `HeatmapFragment`. Jelikož počet záznamů z měření může pro 1 projekt dosáhnout stovek tisíc záznamů, není vhodné stahovat všechny záznamy do mobilního zařízení. Byla proto zvolena možnost přípravy dat na serveru do formy 2D histogramu, jehož rozměry jsou závislé na velikosti sledovaného objektu, nikoli na počtu měření. Pro jednoduchost výpočtu je histogram vytvořen jako obdélník kolmý na rovnoběžky opisující plánek objektu. Kromě samotných dat se tedy z backendu přenáší i souřadnice jihozápadního a severovýchodního rohu. Pro vykreslení heatmapy se používá knihovna „Google Maps Android API utility library“ 4.3.3. Ta má třídu `HeatmapTileProvider`, které se dají předat vážené souřadnice. V našem případě se prostor definovaný obdélníkem histogramu rozdělí na mřížku v rozměrech histogramu a pro každé políčko se vloží do `HeatmapTileProvider` vážená souřadnice se středem políčka a váhou označující počet měření v daném políčku.

4.2.5.4 Komunikace mezi komponenty

Abychom nemuseli všude mezi pomocnými třídami předávat Context, je uchovávána statická instance třídy `App`, což je potomek `Application`. Instance se získá



Obrázek 4.4: Zobrazení heatmapy. Modré piny zobrazují rohy plánku, černý obdélník znázorňuje umístění histogramu.

zavoláním statické metody `App.getInstance()`. Toto nezpůsobí žádné memory leaky, neboť samotná Application instance je singleton. [22] Navíc nám použití singletonu umožní přistupovat k metodám v naší implementaci `App`. V této třídě také uchováváme instanci Otto event busu 4.3.4, který se používá pro komunikaci mezi různými komponenty aplikace. Pokud nějaká obrazovka inicializuje synchronizaci, které probíhá ve výše zmíněných službách, zaregistruje se na změnu stavu popř. informaci o chybě dané služby pomocí anotace `@Subscribe`. Patřičná služba pak pošle svůj stav do busu a všechny takto zaregistrované třídy tento stav obdrží. Jelikož samotný bus neřeší synchronizaci mezi vlákny, byla pro zjednodušení implementace vytvořena třída `MainThreadBus`, která rozšiřuje třídu `Bus` o metodu `postOnMainThread`. Tato metoda spouští veškeré události pomocí Handleru v hlavním vlákně. Následující kód znázorňuje použití tohoto vzoru:

```
//ProjectsUpdateService.java
//notify registered listeners about status change
App.getBus().postOnMainThread(newStatus);


//PlaceMapActivity.java
//subscribe for ProjectsUpdateService changes
@Subscribe
public void onServiceStatusChanged(ServiceStatus status) {
    ...
}
```

4.2.6 Modul backend

Backend byl postaven na platformě Google App Engine s frameworkem Google Cloud Endpoints. Tento framework umožňuje vytvářet API pomocí anotací a má plnou podporu OAuth 2.0 autentikace. API je definováno ve třídách `Accounts`, `Beacons`, `Projects` a `Client`. API `Accounts` slouží pouze správci serveru pro správu uživatelů, `Beacons` a `Projects` obsluhuje administrátorskou aplikaci a vyžaduje přístup pomocí OAuth 2.0 a `Client` je veřejné API sloužící pro klientskou knihovnu. Zde není vyžadována žádná autentikace, neboť zasílaná měření jsou anonymní. Níže je uveden příklad definice třídy s API:

```
@Api(
    name = "ibloc",
    version = "v1",
    scopes = {EMAIL_SCOPE},
    clientIds = {WEB_CLIENT_ID, ANDROID_CLIENT_ID},
    audiences = {ANDROID_AUDIENCE},
    namespace = @ApiNamespace(ownerDomain, ownerName, ...)

public class Projects {
...
}
```

V hlavičce se definuje název API, který poté poslouží i jako jméno třídy ve vygenerované klientské knihovně. Uvnitř této třídy se pak definují samotné metody pomocí anotace `@ApiMethod`:

```
@ApiMethod(name = "projects.insert")
public Project insertProject(Project project, User user)
    throws ConflictException,
    OAuthRequestException,
    UnauthorizedException {
    ...
}
```

Pokud obsahuje metoda parametr `User`, je k přístupu vyžadován OAuth 2.0 a tento objekt pak obsahuje informace o přihlášeném uživateli včetně jeho e-mailové adresy, která se používá jako ID uživatele. Takto nadefinovaná metoda vygeneruje svoji klientskou implementaci, kterou je možné použít v aplikacích přistupujících k backendu následovně:

```
Project createProject(Project project) throws IOException {
    return mServer.projects().insert(project).execute();
}
```

Získání objektu `mServer` je popsáno v sekci 4.2.5.1.

Pro přístup k DataStore byla zvolena ORM knihovna objectify. Uložení oprávnění bylo vyřešeno pomocí bitových mask. Toto řešení umožňuje ukládat oprávnění formou 1 čísla, což usnadňuje jak komunikaci mezi backendem a aplikací tak ukládání do databáze. Oprávnění se používá mezi uživateli a projekty. Pokud uživatel nemá oprávnění ani k zobrazení daného projektu, není v databázi v tabulce s oprávněním pro daného uživatele o daném projektu žádný záznam. To má za následek, že záznamy v tabulce oprávnění pro daného uživatele odpovídají projektům, které může vylistovat a tím pádem nás netrápí, že se v databázi nelze na konkrétní hodnotu oprávnění dotazovat. Následující kód znázorňuje použití bitových operací pro nastavování a čtení konkrétního bitu:

```
boolean hasPermission(PermissionType type, int permissions) {
    return (permissions & type.getMask()) == type.getMask();
}

int setPermission(PermissionType type,
                          boolean enabled,
                          int permissions) {
    if (enabled) {
        //sets desired bit to 1 using bitwise OR
        //  10001
        // |00100
        // =10101
        permissions |= type.getMask();
    } else {
        //first we need to ensure that the desired bit
        //is set to 1, so we can later set it with XOR to 0
        //  10101
        // ^00100
        // =10001
        permissions |= type.getMask();
        permissions ^= type.getMask();
    }
    return permissions;
}
```

V kódu je pak práce s oprávněním ještě usnadněna pomocí parametrizovaného enumu `PermissionType`:

```
public enum PermissionType {
    PROJECT_VIEW(0), PROJECT_UPDATE(1),
    MANAGE_USERS(5), OWNER(6);

    private int mBitPos;

    PermissionType(int bitPos) {
        mBitPos = bitPos;
    }

    public int getMask() {
        return 1 << mBitPos;
    }
}
```

4.2.7 Optimalizace

V této sekci jsou popsány optimalizace, které se používají pro co největší zvýšení plynulosti aplikace, snížení nároků na prostředky a energetickou úspornost.

4.2.7.1 Přenos dat

Google Cloud Endpoints používá hned několik optimalizací pro přenos dat. První optimalizace slouží k zamezení opakovaného přenosu stejných dat. Implementace je formou tzv. ETagu. HTTP ETag je identifikátor posílaný v těle odpovědi ze serveru, který identifikuje posílaný obsah. Klient si tento identifikátor uloží do vyrovnávací paměti a při dalším dotazu na stejný typ obsahu ho pošle zpět jako hlavičku `If-None-Match`. Poté server zkontroluje, jestli má nová data a pokud ne, vrátí odpověď `HTTP 304 Not Modified`. Pokud na serveru jsou nová data, vrátí je spolu s novým ETagem.

Další optimalizací přenosu dat je komprese pomocí gzip. Komprese pomocí gzip se nastaví v requestu HTTP hlavičkou `Accept-Encoding: gzip`. Odpověď je pak komprimována algoritmem gzip. Tato optimalizace je nejvíce znát při stahování histogramu, jehož velikost byla u testovacího projektu 15.3 kB, avšak díky kompresi se přeneslo pouhých 445 B.

4.2.7.2 Bluetooth scanování

Bluetooth scanování je spouštěno ve vlastním procesu. To zajišťuje nezávislost na hlavním procesu, kde běží zbytek aplikace, což umožňuje systému hlavní proces odstranit z paměti, pokud se aplikace nepoužívá, aniž by ovlivnil scanování. Zároveň je scanování prováděno v cyklech, kdy se po nějakou dobu provádí scanování a pak se určitý čas čeká, než se opět spustí. To umožňuje zařízení po dobu čekání přejít do spánku. U Android verze 5.0 a novější se

používá nové API pro scanování, které umožňuje nastavit prioritu i filtry. Nastavením priority na nízko-energetický mód ještě více přispívá k menší spotřebě baterie. Nastavení filtrů umožňuje nechat systém na nízké úrovni scanovat pouze námi definované iBeacony, což může při velkému množství viditelných iBeaconů vést k nižším nárokům na prostředky.

4.2.7.3 Zakázání komponent

Pokud vývojář vypne automatické spouštění scanování popř. celé SDK, dojde k zakázání celých komponent pomocí třídy `PackageManager` a metody `setComponentEnabledSetting`. Toto chování nejen odstraní služby popř. jiné komponenty ze systému ale také odregistruje BroadcastReceivery, což znamená, že se nebude muset aplikace spouštět při zapnutí/vypnutí Bluetooth nebo při dokončení bootu, jen aby zjistila, že je automatické spouštění zakázané. Toto nastavení je perzistentní a zakázané komponenty zůstanou zakázány, dokud se přes stejnou metodu opět nepovolí.

4.2.7.4 Recyklace View

Vzhledem k tomu, že `ListView` může zobrazovat velké množství položek a mobilní zařízení mají omezenou paměť, `View` položek, které nejsou aktuálně v zobrazované části `ListView`, se v paměti neuchovává. Místo toho se pro každou nově zobrazenou položku při posouvání v `ListView` dané `View` znovu vytváří. Toto vytváření nových `View` však může být časově náročné a tak se zejména při rychlém posouvání UI zasekává.

Z toho důvodu systém do metody `onCreateView()`, která vytváří `View` pro nově zobrazenou položku, předává `convertView`, což je `View`, které díky posouvání listu již není viditelné a může se tedy zrecyklovat.

Pakliže se `View` položky skládá z více dalších `View`, je potřeba si pro dosažení nejlepšího výsledku tyto prvky pamatovat spolu s daným `View` položky a nevyhledávat je při každým naplňování dané položky daty. Toto se provádí vytvořením třídy, která si drží odkaz na tyto `View` a referenci na tuto třídu předáme do `View` položky přes metodu `setTag()`. Znázornění tohoto principu najdete v ukázce níže.

```
class ViewHolder { //trida pro drzeni View
    @InjectView(R.id.name)
    TextView mName;
    ...
}


@Override
public View getView(int position, View convertView,
    ViewGroup parent) {
    if (convertView == null) {
        convertView = mInflater.inflate(resourceId, parent, false);
        convertView.setTag(new ViewHolder(convertView));
    }
    ViewHolder holder = (ViewHolder) convertView.getTag();
    //naplneni noveho radku daty
    holder.mName.setText(getItem(position).getName());
    ...
}
```

4.2.7.5 Databáze

Přístup do databáze je vždy řešen z jiného vlákna, jelikož čtení/zápis na interní paměť může způsobovat zaseknutí UI, pokud by operace běžela z hlavního vlákna. Při vkládání většího množství položek do databáze bylo použito sloučení těchto požadavků do jedné transakce. Tím se vkládání razantně zrychlilo. Bez použití jedné transakce pro všechny požadavky by totiž SQLite vytvořilo samotnou transakci pro každý požadavek, což je velmi neefektivní, protože transakce provádí samotný zápis na disk, což s sebou přináší určitou režii. Je lepší na disk zapsat jednou větší množství dat než vícekrát menší části.

4.3 Použité technologie

4.3.1 Android support libraries

https://developer.android.com/tools/support-library/features.html

Podpůrné knihovny umožňující použití nových API i na starších verzích Android OS.

4.3.2 Google Play Services

https://developer.android.com/google/play-services/index.html

Google Play Services mají za cíl poskytování API ke Google službám jako jsou Google Maps, Google+, OAuth 2.0, Google Drive a další nezávisle na verzi OS. Jsou distribuovány automaticky přes Play Store, takže okolo 93% zařízení s Android OS 2.3 a novější mají nainstalovanou aktuální verzi. [11]

Knihovna pro vývojáře slouží jako rozhraní a k napojení na Play Services v zařízení.

4.3.3 Google Maps Android API utility library

https://developers.google.com/maps/documentation/android/utility

Jedná se o pomocnou knihovnu pro Google Maps pro Android OS, která přidává následující funkce:

- Podpora pro heatmapy
- Textové piny na mapách
- Shlukování pinů v závislosti na zoomu
- Dekódování tras získaných z Directions API
- Výpočet sférické vzdálenosti mezi 2 souřadnicemi

4.3.4 Otto

http://square.github.io/otto

Otto event bus je knihovna určená pro komunikaci mezi komponenty v aplikaci. Třída, která má zájem o příjem události implementuje patřičnou metodu, kterou označí anotací `@Subscribe`:

```
@Subscribe
public void answerAvailable(AnswerAvailableEvent event) {
    // React to the event somehow!
}
```

Pro odebírání těchto událostí je potřeba instanci této třídy zaregistrovat do busu pomocí metody `register`. Odregistrování odběru se provádí analogicky metodou `unregister`. U Android obrazovek je zvykem daný fragment nebo aktivitu zaregistrovat v metodě `onResume` a odregistrovat v metodě `onPause`.

Pro poslání události pak stačí zavolat:

```
bus.post(new AnswerAvailableEvent(42));
```

Otto pak najde všechny zaregistrované třídy, které mají metodu s parametrem `AnswerAvailableEvent` a anotací `@Subscribe` a u nich pak danou metodu zavolá.

4.3.5 Butter Knife

http://jakewharton.github.io/butterknife

Knihovna zjednodušující práci s `View`, u kterých provádí injekci do parametrů třídy, které jsou označeny anotací `@InjectView`. Pomocí anotace `@OnClick` je také možné přiřadit dané metodě událost kliknutí na patřičné `View`. Kromě injekce jednotlivých `View` je možné provést injekci i skupiny `View`:

```
@InjectViews({ R.id.first_name, R.id.middle_name, R.id.last_name })
List<EditText> nameViews;
```

Při kompilaci se vygeneruje vnitřní třída `$$ViewBinder` pro každou třídu, která používá tyto anotace, za použití `Annotation Processing` v Javě. Díky tomu nedochází k degradaci výkonu způsobeného používáním reflexe.

4.3.6 Picasso

http://square.github.io/picasso

Picasso slouží k asynchronnímu načítání obrázků a jejich zobrazení do `ImageView` v Androidu. Knihovna stažené obrázky uchovává v LRU cahce v operační paměti a interní paměti zařízení pro zrychlený přístup k již staženým obrázkům. API je velmi intuitivní:

```
Picasso.with(context).load(IMAGE_URL).into(imageView);
```

4.3.7 Styled dialogs

https://github.com/avast/android-styled-dialogs

Tato knihovna usnadňuje použití dialogů v Android OS a zároveň přináší zpětnou kompatibilitu pro nový vzhled představený v Android 5.0.

4.3.8 Objectify

https://github.com/objectify/objectify

Objectify umožňuje objektový přístup k Datastore na Google App Engine. Umožňuje ukládání, získávání, mazání a dotazování se na objekty:

```
@Entity
class Car {
    @Id String vin; // Can be Long, long, or String
    String color;
}

ofy().save().entity(new Car("123123", "red")).now();
Car c = ofy().load().type(Car.class).id("123123").now();
ofy().delete().entity(c);
```

4.3.9 ORMLite

http://ormlite.com/ http://en.wikipedia.org/wiki/ORMLite

ORMLite je open source Framework, který poskytuje odlehčené objektově relační mapování mezi Java třídami a SQL databázemi.

Vyznačuje se jednoduchým použitím – Jednotlivé sloupce v tabulce se vytváří přidáním anotace `@DatabaseField` k proměnné třídy. Samotná tabulka se pak vytvoří zavoláním metody `TableUtils.createTable()`, které se předá třída definující danou tabulku. V projektu (v Android modulech) se používá pouze k vytváření tabulek. Samotné přidávání a dotazování se už řeší pomocí ContentProvideru a Cursoru pro menší nároky na prostředky.

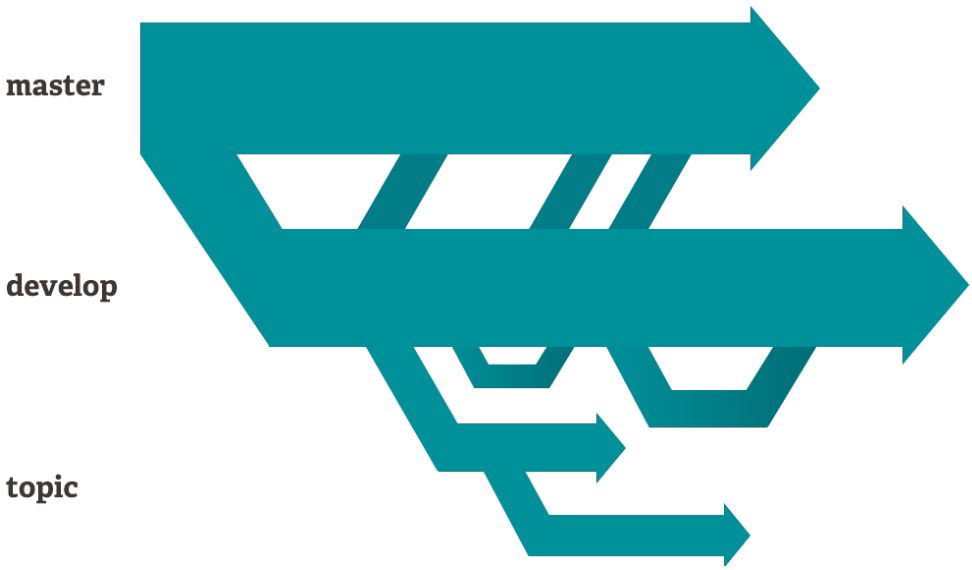
V následující ukázce kódu je ukázána definice databázové tabulky pomocí Java třídy:

```
public class Table {
    @DatabaseField(id = true)
    Long id;
    @DatabaseField
    String subject;
    @DatabaseField
    String description;
}
```

4.3.10 GIT

http://git-scm.com/

Vývoj takto rozsáhlého projektu není možný bez nějakého nástroje na správu verzí, přestože na projektu pracuje pouze jeden člověk. Pokud se například provedou na programu nějaké změny, které se ukážou být špatnou cestou, není nic jednoduššího než obnovit stav projektu z jednoho z předchozích commitů. Dále je možné díky nástroje pro správu verzí kdykoli u každého souboru



Obrázek 4.5: Typické použití větví v GITu – repositář je rozdělen do hlavní větev, vývojovou větev, z které se vytváří větve pro vývoj na sobě nezávislých modulů.
Zdroj: http://www.zdrojak.cz/clanky/nette-framework-mvc–mvp/

nahlédnout na jeho libovolnou verzi a porovnat, k jakým změnám od této verze došlo.

Takových nástrojů existuje mnoho (GIT, SVN, Mercurial...). Výhoda GITu však spočívá v jeho rozdělení na lokální a vzdálený repositář. Díky manuální synchronizaci lokálního repositáře se vzdáleným je tak možné pracovat s tímto verzovacím systémem i bez připojení k serveru.

Další výhodou tohoto nástroje je možnost rozdělit vývoj do větví, ve kterých například může paralelně probíhat vývoj různých na sobě nezávislých modulů. Jakmile vývoj takového modulu je u konce, můžou se větve opět sloučit. Pokud se zjistí, že daný modul nebude použitelný, může se celá větev zahodit. Dále je běžné použití větví pro rozlišení na „Master“ větev, kde je vždy aktuální stabilní verze programu a „Devel“ větev, kde probíhá vývoj a testování nových funkčností. Po vydání nové verze se většinou obě větve sloučí. Znázornění použití větví je na obrázku 4.5.

4.3.11 Fiddler

http://www.telerik.com/fiddler

Fiddler je debugovací server, který umožňuje prohlížet obsah HTTP komunikace. Když se v mobilním zařízení nastaví pro danou WiFi proxy server s IP adresou počítače, na kterém tento nástroj běží, a stáhne se vygenerovaný

certifikát, je možné na počítači, který je ve stejné síti, prohlížet veškerou komunikaci (i https) mezi mobilním zařízením a Internetem.

Tento nástroj byl použit pro ladění komunikace mezi backendem a mobilní aplikací.

KAPITOLA 5

Testování

Tato kapitola se zabývá testováním mobilní aplikace. Je zde popsán způsob testování, jaké technologie byly použity a jakých výsledků bylo při testování dosaženo.

Testování je prováděno na těchto zařízeních:

1. Google Nexus 5
    - Běžně používaný telefon
    - Android 5.1
    - Baterie 2300 mAh Li-ion
    - Qualcomm Snapdragon 800
2. Google Nexus 5
    - Telefon v továrním nastavení
    - Bez SIM karty
    - Android 5.1
    - Baterie 2300 mAh Li-ion
    - Qualcomm Snapdragon 800
3. HTC One mini 2
    - Telefon v továrním nastavení
    - Bez SIM karty
    - Android 4.4.2
    - Baterie 2100 mAh Li-Poly
    - Qualcomm Snapdragon 400

Tabulka 5.1: Měření vzdálenosti iBeaconu

Skutečná vzdálenost	Naměřený průměr	Sm. odchylka	Průměrná chyba
0.8 m	0.97	0.15	22 %
1.6 m	2.05	0.23	29 %
2.6 m	2.14	0.36	19 %

5.1 iBeacon signál

Při tomto testu bylo provedeno měření kvality signálu iBeacon. K měření bylo použité zařízení 1. Zařízení bylo umístěno ve 3 různých vzdálenostech od iBeaconu a nebyla mezi zařízením a iBeaconem umístěna žádná překážka. V tabulce 5.1 jsou uvedeny statistiky těchto měření. Výsledky měření jsou znázorněny v grafech v appendixu C.

Jak je vidět, signál docela kolísá a tudíž nelze očekávat vysokou přesnost od tohoto způsobu měření. Na druhou stranu je tato technologie velmi málo náročná na spotřebu energie a tudíž se nízká přesnost dá předpokládat.

5.2 Spotřeba energie

V použitých iBeaconech Kontakt.io je baterie CR2477N s kapacitou 950 mAh [31], která dokáže iBeacon napájet přibližně rok a půl při výchozím intervalu vysílání, který je 350 ms.

Pro měření spotřeby zařízení byly použity Nexus 5 a Htc One Mini 2 v továrním nastavení (zařízení 2, 3), které byly přepnuty do módu letadla, aby neprobíhala žádná synchronizace, která by mohla ovlivnit měření. Telefony byly vždy před začátkem měření nabity na 100%. Měření probíhalo vždy 9h a 10 minut a byly použity následující scénáře, které mají ukázat samotnou spotřebu zapnutého Bluetooth a spotřebu naší demo aplikace.

- Mód letadla (WiFi i Bluetooth vypnuty)
- Mód letadla se zapnutým Bluetooth.
- Mód letadla se zapnutým Bluetooth a nainstalovanou naší demo aplikací.

Pro měření byl použit nástroj `dumpsys`, který je součástí Android OS. Před každým měřením bylo zapnuto detailní měření spotřeby pomocí:

```
adb shell dumpsys batterystats --enable full-wake-history
```

Detailní statistiky od posledního nabití se po uplynutí daného intervalu získaly následujícím příkazem:

Tabulka 5.2: Měření spotřeby energie u Nexus 5 (zařízení 2)

Scénář	Konečné % baterie	Spotřeba mAh
Mód letadla	98	38.5
Mód letadla + Bluetooth	99	39.8
Mód letadla + Bluetooth + demo	97	45.7
Mód letadla + Bluetooth + demo fixed	95	139

Tabulka 5.3: Měření spotřeby energie u HTC One mini 2 (zařízení 3)

Scénář	Konečné % baterie
Mód letadla	100
Mód letadla + Bluetooth	96
Mód letadla + Bluetooth + demo fixed	96

```
adb shell dumpsys batterystats --charged > batteryStats.txt
```

Součástí těchto statistik je i přibližná hodnota spotřebovaných mAh (Computed drain), které nás zajímají. Tato informace je bohužel dostupná pouze u Androidu 5 a novějším, takže u zařízení 3 chybí. Výsledky jsou zaneseny v tabulce 5.2 resp. 5.3 spolu s % baterie, které při konci měření udával systém.

Jak je vidět, samotné zapnuté Bluetooth nemá u těchto zařízení téměř žádnou spotřebu. Udávané % po konci měření je u zařízení 2 dokonce vyšší než u scénáře bez zapnutého Bluetooth, avšak toto chování může způsobit rozdílné počáteční nabití zařízení. Mobilní telefony totiž kvůli menšímu opotřebení baterií udržují po nabití stav blízký 100%, nikoli však přesných 100% a tudíž se nelze při takto malé spotřebě spoléhat na toto číslo. [30] Mnohem důležitější je v tomto případě vypočítaná spotřeba v mAh uvedená v posledním sloupci. U zařízení 3 má Bluetooth trochu větší spotřebu než u zařízení 2, avšak i tady jsou 4% za 9h zanedbatelné a u testu bez zapnutého Bluetooth, který sice skončil s udávaným číslem 100% baterie, nemůžeme spotřebu přesně určit kvůli chybějícímu údaji v mAh, takže ve výsledku zde může Bluetooth spotřebovat i méně než 4% baterie.

Test s demo aplikací u zařízení 2 skončil až podezřele dobře. Spotřeba aplikace byla minimální (konečný stav 97% a spotřeba pouhých 45.7 mAh). Po podrobnějším prozkoumání byla zjištěna chyba v knihovně `btscanner` způsobená přechodem procesoru do hlubokého spánku. Když je procesor v hlubokém spánku, přestává se počítat hodnota `SystemClock.uptimeMillis()`, která se používá v systému pro naplánované spuštění kódu s určitým zpožděním (`handler.postDelayed(callback, delay)`). To mělo za následek, že se sice

scanovací služba díky Alarmu spustila pro každý cyklus, avšak procesor okamžitě přešel do režimu spánku a tak daný cyklus nedoběhl. Z logů pak šlo vyčíst, že za 9 hodin prošel celý cyklus scanování pouze 41x.

Byl proto implementován `WakeLock`, který po dobu scanování udrží procesor probuzený a test byl proveden znovu. Tentokrát cyklus proběhl 543x a projevila se i větší spotřeba baterie. I v tomto případě je však spotřeba velmi pozitivní. U zařízení 2 byl konečný stav 95% a spotřeba pouhých 139 mAh. U zařízení 3 pak byl konečný stav 96%, což je stejné číslo jako u scénáře bez nainstalované aplikace, nicméně z logů šlo vyčíst správné chování. Díky chybějící informaci s spotřebě v mAh nelze u tohoto zařízení spotřebu přesně vyčíst a musíme si vystačit pouze s udávaným %, které nemusí být z výše uvedených důvodů úplně přesné, avšak postačí to k vytvoření závěru, že scanování iBeaconů má i u Androidu 4.4 téměř zanedbatelnou spotřebu.

5.3 Stress test

Stress test je velmi důležitý u mobilních aplikací, jelikož dokáže jednoduše odhalit pády či zamrznutí aplikace, což jsou nejzásadnější chyby, které mohou lehce uživatele odradit od používání a způsobit negativní hodnocení jinak kvalitní aplikace.

Pro stress testování administrátorské aplikace bylo použito nástroje Monkey dodávaného spolu s Android SDK. Tento nástroj ovládaný příkazovou řádkou funguje na principu posílání do vybraného zařízení pseudo-náhodných, avšak zopakovatelných událostí, jako jsou kliky, dotyky, gesta nebo systémové události.

Nástroj lze omezit na jméno balíčku, čímž je zaručeno, že se v průběhu testu nedostane pryč z aplikace. Dále je možné nastavit počet vygenerovaných událostí, s jakou prodlevou se posílají nebo zda-li se má test zastavit, když aplikace spadne. Události se generují náhodně, avšak generování lze ovládat pomocí čísla „Random seed“. Pokud se Monkey spustí se stejným tímto číslem, vygeneruje stejnou sekvenci událostí. [15]

Aplikace byla tímto nástrojem testována s následujícími parametry:

- Omezení balíčku nastaveno na `cz.mpelant.ibloc.admin`
- Počet generovaných událostí nastaven na 5000
- Žádná prodleva mezi událostmi

Pro zlepšení efektu stress testu bylo zařízení za běhu tohoto nástroje neustále nakláněno různými směry aby docházelo ke změně orientace obrazovky. Při rotaci obrazovky se totiž právě spuštěná aktivita restartuje, což je velmi často právě příčinou pádu aplikací.

Při testu byly zjištěny drobné chyby, po jejichž opravení aplikace prošla několika cykly stress testu bez jediné chyby.

Jelikož systém může při nedostatku prostředků zabít komponenty aplikace, bylo potřeba otestovat správné chování aplikace i u těchto případů. Toto chování se dá nasimulovat následujícím příkazem, ve kterém je jako poslední argument id aplikace:

```
adb shell am kill cz.mpelant.ibloc.admin
```

U administrátorkého modulu bylo testováno správné obnovení stavu obrazovek při znovuspuštění aplikace a u demo modulu pak správné restartování scanovací služby. Byl zjištěn nedostatek u restartování služby. Pokud se totiž nechalo na systému, aby službu restartoval, nebylo zaručeno, aby se tak stalo v rozumné době. Mohl tak klidně nastat případ, že systém naplánoval znovuspuštění služby až za 2h, což je pro použití v tomto typu aplikace nedostačující:

```
Killing 30548:cz.mpelant.ibloc.demo/u0a121 (adj 11):
        kill background
Killing 30675:cz.mpelant.ibloc.demo:scan/u0a121 (adj 11):
        kill background
Scheduling restart of crashed service cz.mpelant.ibloc.demo/
        cz.mpelant.scanner.service.BeaconScanService in 7752058ms
```

Bylo proto zajištěno, aby se služba znovu spustila pomocí Alarmu a nespoléhala se na automatické restartování systémem. (více info v sekci 4.2.1)

5.4 Usability test

Jakmile bylo otestováno, že je administrátorská aplikace dostatečně stabilní, byl proveden usability test, který má za cíl prověřit funkčnost a především použitelnost aplikace. Vybraným uživatelům se předkládaly úlohy a pozoroval se postup vykonávání požadovaných činností. Důležitou součástí tohoto testu byla i zpětná vazba po dokončení testu s případnými návrhy na zlepšení aplikace.

5.4.1 Popis testu

Účelem testu je prověřit stěžejní funkce administrátorská aplikace a získat od uživatelů co nejvíce informací, které napoví, jak aplikaci upravit, aby byla uživatelsky přívětivá. Scénář testu byl zvolen tak, aby si uživatel vyzkoušel všechny základní funkce mobilní aplikace, aby se zjistilo, zda-li jsou uživatelské prvky ideálně umístěny a jestli je navigace v aplikaci intuitivní.

5.4.2 Profil uživatelů testujících aplikaci

Vzhledem k cílové skupině byli k testování pozváni uživatelé, kteří již mají zkušenosti s používáním telefonu s operačním systémem Android, aby byli obeznámeni s obecnou navigací u aplikací pro tuto platformu a tudíž měli pro testování stejné podmínky.

5.4.3 Metoda testování

Testování proběhne na Google Nexus 5 (zařízení 1), kde bude vytvořen nový účet, pod kterým budou uživatelé aplikaci testovat. K tomuto účtu bude přidán projekt s již existujícími záznamy pro zobrazení heatmapy. Po ověření splnění podmínek z kapitoly 5.4.2 bude uživatelům předložen seznam úkolů, které mají provést. V asistenci s dozorem testu budou samostatně provádět dané úkoly. Po splnění všech úkolů bude testovaným osobám předložen post-test dotazník. Poté s nimi bude vedena diskuze, která se promítne v realizaci závěrů.

Průběh celého testování bude nahráván aplikací pro zaznamenávání dění na obrazovce mobilního telefonu spolu se zvukem z vestavěného mikrofonu. V testovacím zařízení bude zapnuta vizualizace dotyků.

5.4.4 Testovací úkoly

Scénář byl zvolen tak, aby prozkoumal stěžejní funkce aplikace a zároveň aby byl v chronologickém uspořádání a aspoň z části simuloval běžné používání aplikace. U každého úkolu je zde popsáno i optimální řešení.

1. Vytvořte nový projekt, pojmenujte ho vašim jménem. Při vytváření projektu rovnou vložte URL plánku, která je již zkopírovaná ve schránce zařízení.
   Optimální: Uživatel si spustí aplikaci, vybere z menu „New Project“, vyplní jméno, vloží ze schránky URL plánku a potvrdí tlačítkem „Create“.

2. Nastavte pozici plánku, aby aspoň částečně lícoval s vámi zvoleným místem.
   Optimální: Uživatel ze seznamu projektu dlouhým podržením na nově vytvořený projekt vybere možnost „Edit Project“, následně standardními multidotekovými gesty přiblíží, posune a otočí mapu tak, aby plánek lícoval s libovolným místem. Pozici plánku potvrdí tlačítkem „Save“.

3. Umístěte na libovolné místo tyto 3 iBeacony a pojmenujte je kódem, který je uveden zespodu iBeaconů. (Uživatel dostane do ruky 3 iBeacony s napsaným kódem zespodu)
   Optimální: Uživatel z menu vybere položku „Add beacon“, ze seznamu zvolí postupně všechny 3 iBeacony, které jsou v dosahu, do dialogu, který

se objeví napíše jejich kód a poté přidržením prstu na pin na mapce ji umístí na libovolné místo. Umístění potvrdí tlačítkem „Save“.

4. Upravte libovolně pozici jedné iBeacony a přejmenujte ji.
   Optimální: Uživatel klikne na libovolný pin reprezentující iBeaconu, zobrazí se informační popup. Uživatel na něj klikne, vybere možnost „Edit position“ a stejně jako v předchozí úloze iBeaconu umístí na libovolné místo. Poté opět klikem na informační popup vybere z menu „Edit name“ a přejmenuje ji, stejně jako v předchozí úloze.

5. Nyní si zobrazte heatmapu u projektu „eman project“.
   Optimální: Uživatel se vrátí tlačítkem zpět na seznam projektů, klikne na projekt „eman project“ a tím se dostane na zobrazení heatmapy.

6. Vyfiltrujte pouze záznamy ze dne 20.4.1015.
   Optimální: Uživatel klikne na tlačítko pro nastavení dne začátku filtru, kde zvolí 20.4.2015, dále na tlačítko pro nastavení dne konce filtru, opět zvolí 20.4.2015 a nakonec vybere tlačítko pro nastavení času konce, kde vybere 23:59.

7. Vraťte se zpět do předchozího, vámi vytvořeného projektu a přidejte spolupracovníka s e-mailovou adresou „devtest@eman.cz“ (povolte mu všechna práva).
   Optimální: Uživatel se vrátí tlačítkem zpět na seznam projektů, dlouhým podržením na projekt s jeho jménem vybere možnost „Edit Project“, z menu pak možnost „Manage users“. Tím se dostane na seznam spolupracovníků, kde opět výběrem položky „Add“ z menu zobrazí dialog pro přidání nového spolupracovníka. Zde vyplní e-mailovou adresou „devtest@eman.cz“ a zaškrtne všechna oprávnění.

5.4.5 Post-test dotazník

Post-test dotazník slouží k získání prvotních dojmů, pocitů a názorů k právě vykonaného testu.

1. Jak se vám administrátorská aplikace líbila? (známka 1–5, jako ve škole)
2. Přišlo vám uspořádání ovládacích prvků jasné a přehledné?
   - Ano
   - Spíše ano
   - Spíše ne
   - Ne
3. Přidal byste do aplikace nějaké funkce? Pokud ano, jaké?
4. Pokud máte jiné připomínky, zde je prosím uveďte.

5.4.6 Výsledky testu

Zde budou rozepsány nejpřínosnější záznamy testů. Záznam z tohoto testu je součástí přiloženého DVD.

5.4.6.1 Testovaná osoba 1

Muž, 20-30 let.

1. Tento úkol zvládl uživatel bez problémů optimální cestou.
2. Uživatel neprovedl dlouhý stisk, avšak pouze tapnul na projekt. Díky automatickému přesměrování do nastavení projektu, pokud není nastavení dokončeno, se však na na cílovou obrazovku dostal (objevila se při tom hláška, že uživatel musí nejdřív nastavit pozici plánku). Zde klikal na nefunkční tlačítko „My location“. Dále bylo uživateli naznačeno, že může mapu i otáčet a hned si uvědomil, že jde o Google mapy a že zná multidoteková gesta a zbytek úkolu splnil správně.
3. Tento úkol zvládl uživatel bez problémů optimální cestou.
4. Tento úkol zvládl uživatel bez problémů optimální cestou.
5. Tento úkol zvládl uživatel bez problémů optimální cestou.
6. Zde uživatel vybral pouze konečné datum správně a muselo mu být napovězeno, že musí vybrat i začátek. Výběr správného času provedl také až s nápovědou. Poté nesprávně klikl na tlačítko „Now“ v domnění, že se tím potvrdí výběr časového intervalu, ale okamžitě si uvědomil, že to neměl dělat.
7. Uživatel si sám neuvědomil, že na seznamu projektů musí na daný projekt provést dlouhý stisk a vybrat položku „Edit project“. Místo toho si krátkým stiskem zobrazil heatmapu a přemýšlel, jak se dostat do editace. Po nápovědě, že se do editace dostane pouze ze seznamu projektu, již provedl zbytek úkolu optimální cestou.

5.4.6.2 Testovaná osoba 2

Muž, 20-30 let.

1. Tento úkol zvládl uživatel bez problémů optimální cestou.
2. Uživatel neprovedl dlouhý stisk, avšak pouze tapnul na projekt. Díky automatickému přesměrování do nastavení projektu, pokud není nastavení dokončeno, se však na na cílovou obrazovku dostal (objevila se při tom hláška, že uživatel musí nejdřív nastavit pozici plánku). Zde klikal na nefunkční tlačítko „My location“. Zbytek úkolu proběhl optimální cestou.

3. Uživatel se nejprve snažil iBeacony přidat stiskem na požadovanou pozici na plánku a až poté ho napadlo, že se přidávají výběrem z položky v menu. Dále iBeaconu přidal na mapku a správně ji umístil, avšak přehlídl tlačítko „Save“ a snažil se přidat rovnou další iBeaconu.

4. U tohoto úkolu se uživatel snažil najít v menu položku pro úpravu iBeaconů. Když nenašel, snažil se dlouhým stiskem a dvojklikem na piny vyvolat kontextové menu. Až s nápovědou, aby na piny pouze klikl, se mu povedlo zobrazit info okénko a menu pro editaci. Místo přejmenování si spletl položku a opět spustil editaci pozice.

5. Tento úkol zvládl uživatel bez problémů optimální cestou.

6. Tento úkol zvládl uživatel bez problémů optimální cestou.

7. U tohoto úkolu uživatel opět vlezl nejdřív do zobrazení heatmapy a až posléze si uvědomil, že se musí vrátit na seznam projektů a dlouhým stiskem vybrat položku „Edit project“. Zbytek úkolu proběhl optimální cestou.

5.4.6.3 Testovaná osoba 3

Muž, 20-30 let.

1. Tento úkol zvládl uživatel bez problémů optimální cestou.

2. Uživatel neprovedl dlouhý stisk, avšak pouze tapnul na projekt. Díky automatickému přesměrování do nastavení projektu, pokud není nastavení dokončeno, se však na na cílovou obrazovku dostal (objevila se při tom hláška, že uživatel musí nejdřív nastavit pozici plánku). Zde klikal na nefunkční tlačítko „My location“. Zbytek úkolu proběhl optimální cestou.

3. Tento úkol zvládl uživatel bez problémů optimální cestou.

4. Zde uživatel opět hledal nastavení pro editaci iBeacony v menu Actionbaru a když nenašel, napadlo ho kliknout na info okénko u dané iBeacony. Zbytek úkolu proběhl optimální cestou.

5. Tento úkol zvládl uživatel bez problémů optimální cestou.

6. Zde uživatel vybral pouze konečné datum správně a muselo mu být napovězeno, že musí vybrat i začátek. Výběr správného času provedl také až s nápovědou.

7. U tohoto úkolu si uživatel opět sám neuvědomil, že na seznamu projektů musí na daný projekt provést dlouhý stisk a vybrat položku „Edit project“. Místo toho si krátkým stiskem několikrát zobrazil heatmapu

Tabulka 5.4: Post-test dotazník

Odpověď 1	Odpověď 2	Odpověď 3
Jak se vám administrátorská aplikace líbila? (známka 1–5, jako ve škole)		
1	1	2
Přišlo vám uspořádání ovládacích prvků jasné a přehledné?		
Ano	Spíše ano	Spíše ano
Přidal byste do aplikace nějaké funkce? Pokud ano, jaké?		
upozornění na vypnutý bluetooth	–	–
Pokud máte jiné připomínky, zde je prosím uveďte.		
–	Pro uživatele Androidu je aplikace přehledná vzhledem k použití nativních prvků material design.	Zviditelnit prechod do nastaveni projektu

a přemýšlel, jak se dostat do editace. Po nápovědě, že se do editace dostane pouze ze seznamu projektu, již provedl zbytek úkolu optimální cestou.

5.4.7 Výsledky dotazníku

Výsledky dotazníku jsou v tabulce 5.4. Uživatelé si přejí upozornění na vypnuté Bluetooth a lepší zviditelnění přechodu do nastavení projektu.

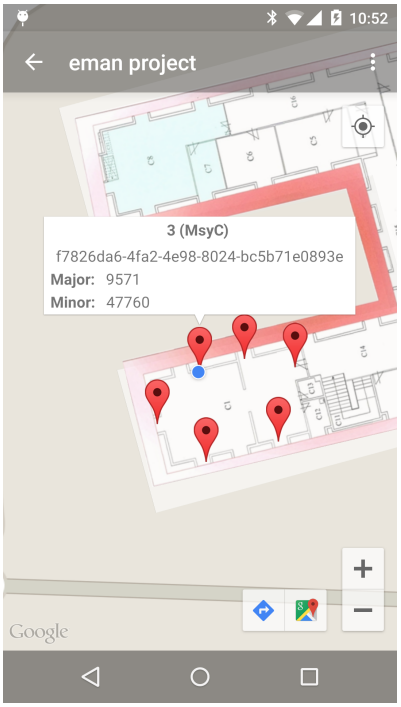
5.4.8 Vyhodnocení testu

Testování odhalilo několik nedostatků v aplikaci.

První nedostatek se nachází již na seznamu projektů, kde uživatelům nedochází, že se do nastavení, přejmenování a mazání projektu dostanou dlouhým stiskem na daný projekt. Tento nedostatek by šel vyřešit přidáním viditelného tlačítka ke každému projetu, které by vyvolalo toto menu. Dále by také bylo vhodné přidat přechod do nastaveni ze zobrazení heatmapy.

Při umístění plánku nefungovalo tlačítko „My location“. Funkce mojí polohy je totiž napojena na vypočítanou polohu z rozmístěných iBeaconů a protože při nastavení pozice plánku ještě nejsou v projektu žádné iBeacony, není dostupná žádná poloha. Řešení by mohlo být použití skutečné polohy, pokud nejsou rozmístěny iBeacony a až po rozmístění iBeaconů polohy z nich vypočítané.

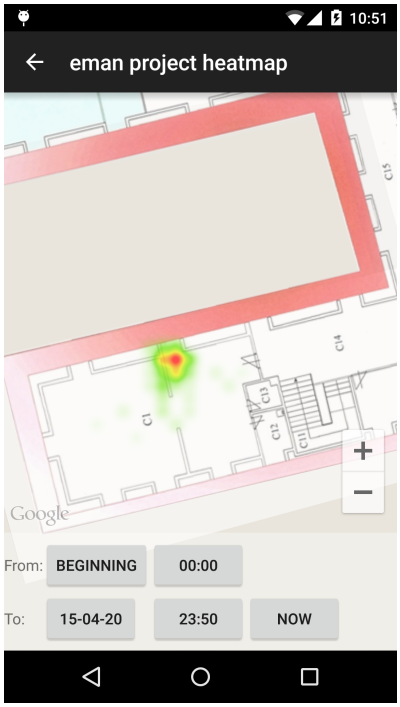
Dalším kamenem úrazu je samotná práce s iBeacony ve formě pinů na mapě (5.1). Přidání nové iBeacony by se mohlo přesunout z overflow menu na viditelnou pozici v Actionbaru, aby uživatelé měli hned na očích, jak se přidávají iBeacony. Dále by bylo dobré detekovat dlouhý stisk na pin,



Obrázek 5.1: Obrazovka nastavení projektu

který by vyvolal kontextovou nabídku pro editaci, mazání a přejmenování, avšak tato funkčnost v Google Maps API chybí. Nabízí se tedy možnost nápovědy, která by s poloviční průhledností překryla mapu a informovala uživatele, že kliknutím na pin se zobrazí informační okénko pro danou iBeaconu, a kliknutím na toto okénko se vyvolá kontextové menu pro editaci. V tomto kontextovém menu by taky stálo za to přejmenovat položku „Edit name“ na „Rename“, jelikož začíná stejně jako položka „Edit position“ a uživatel si ji díky nepozornosti může splést.

Poslední nedostatek byl objeven v obrazovce pro zobrazování heatmapy (5.2) a to konkrétně u výběru časového úseku. Zde si uživatel neuvědomil, že tlačítko s výchozím textem „beginning“ nastavuje datum začátku časového intervalu. Jako řešení by mohlo být získat z backendu čas prvního měření a nastavit výchozí text u tohoto tlačítka na tuto hodnotu.



Obrázek 5.2: Obrazovka heatmapy

Závěr

V průběhu této práce byl realizován systém pro zaznamenávání pohybu lidí v uzavřeném prostoru a následné analýzy v podobě heatmapy. Byla vytvořena administrátorská aplikace pro Android OS umožňující nastavení projektů a zobrazení heatmap, backend na platformě Google AppEngine a knihovna pro Android OS určena pro klientské aplikace, která kromě zaznamenávání polohy uvnitř budovy umožňuje uživatelům polohu také zobrazit.

Určení polohy probíhá za použití minimalizační metody, popsané v části 4.2.2.2, z naměřených vzdáleností od okolních iBeaconů. U této metody byla vypozorována průměrná chyba v závislosti na počtu a rozmístění iBeaconů okolo 2m.

Před samotnou implementací byla provedena analýza zadání a proveden průzkum již existujících řešení, na jejichž základě byly vytvořeny funkční a nefunkční požadavky. Dále byla detailně prozkoumána technologie Bluetooth Low Energy a protokol iBeacon, který tuto technologii používá, a podpora těchto technologií u platformy Android. Zároveň byla prostudována platforma Google App Engine, na které byl postaven backend k tomuto systému.

Po shrnutí všech funkčních i nefunkčních požadavků a prozkoumání technologií potřebných k realizaci tohoto systému byly navrženy obrazovky pro administrátorskou aplikaci. Na základě dat potřebných v daných obrazovkách bylo navrženo API a databáze backendu. Tato databáze je řešena pomocí DataStore, které je součástí platformy App Engine. Jedná se o typ NoSQL, díky čemuž je možné backend v případě potřeby škálovat.

Po dokončení návrhu obrazovek, API a databáze již byly známy veškeré informace potřebné k implementaci. Nejprve byl implementován modul pro scanování iBeaconů, jelikož nativní podpora pro tuto technologii na platformě Android chybí. Byl kladen důraz na co nejmenší nároky na prostředky a tudíž i na spotřebovanou energii. Následně byl implementován backend, který poskytuje API pro administrátorskou aplikaci i klientskou knihovnu. Poté byla vytvořena administrátorská aplikace, která využívá již hotový modul pro scanování iBeaconů a je napojena na backend. Dále byla naprogramována samotná klientská knihovna spolu s algoritmem pro výpočet polohy. Tento algoritmus byl posléze přesunut do samotného modulu, aby ho mohla používat i administrátorská aplikace a zobrazovat v nastavení projektu aktuální vypočítanou polohu pro snadnější optimalizaci rozmístění iBeaconů. Pro otestování

knihovny byla vytvořena demo aplikace, která slouží i jako jednoduchý příklad použití knihovny.

Aplikace byla testována především pomocí stress testu, který odhalil několik chyb. Tyto chyby byly posléze odstraněny. Zároveň byla změřena spotřeba energie. Měření ukázala, že samotné Bluetooth i scanování v naší knihovně má zanedbatelnou spotřebu baterie. Aplikace je tedy vhodná i pro dlouhodobé spuštění na mobilních zařízeních návštěvníků akce, která tento analytický systém použije.

Nakonec byl ještě proveden usability test administrátorské aplikace pro ověření její funkčnosti a použitelnosti. Během tohoto testu se objevilo několik nedostatků, převážně neintuitivní přechod do nastavení projektu a výběr časového úseku pro zobrazení heatmapy. Všechny tyto nedostatky byly odstraněny.

Aplikaci lze dále rozšířit například podporou pro více mobilních platforem nebo vytvořením webového rozhraní administrace. Náš systém lze také integrovat do existujících systémů pro festivaly, konference a další akce tohoto typu.

Literatura

[1] Android Runtime. http://en.wikipedia.org/wiki/Android_Runtime, [Online; cit. 2015-01-10].

[2] Google App Engine. http://en.wikipedia.org/wiki/Google_App_Engine, [Online; cit. 2015-01-10].

[3] Levenberg–Marquardt algorithm. http://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt_algorithm, [Online; cit. 2015-01-10].

[4] Piconet. http://cs.wikipedia.org/wiki/Piconet, [Online; cit. 2015-01-10].

[5] Sundar Pichai to take over as Android chief from Andy Rubin. http://gadgets.ndtv.com/mobiles/news/sundar-342195, 2013, [Online; cit. 2015-01-10].

[6] Ben Elgin: Google Buys Android for Its Mobile Arsenal. http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal, 2005, [Online; cit. 2015-01-10].

[7] Bluetooth SIG: The Low Energy Technology Behind Bluetooth Smart. http://www.bluetooth.com/Pages/low-energy-tech-info.aspx, [Online; cit. 2015-01-10].

[8] David G. Young: Understanding ibeacon distancing. http://stackoverflow.com/a/20434019/522545, [Online; cit. 2015-01-10].

[9] David G. Young: What is the iBeacon Bluetooth Profile. http://stackoverflow.com/a/19040616/522545, [Online; cit. 2015-01-10].

[10] Elise Vogler: Bluetooth vs. Wi-Fi Power Consumption. http://science.opposingviews.com/bluetooth-vs-wifi-power-consumption-17630.html, [Online; cit. 2015-01-10].

[11] Ewan Spence: Google Has 93% Of Android Users On The Latest Google Play Service. http://onforb.es/1qx22Es, [Online; cit. 2015-01-10].

[12] Google: Google Cloud Platform Overview. https://cloud.google.com/appengine/docs/java/endpoints, [Online; cit. 2015-01-10].

[13] Google: Gradle Plugin User Guide. http://tools.android.com/tech-docs/new-build-system/user-guide, [Online; cit. 2015-01-10].

[14] Google: Manifest Merger. http://tools.android.com/tech-docs/new-build-system/user-guide/manifest-merger, [Online; cit. 2015-01-10].

[15] Google: UI/Application Exerciser Monkey. http://developer.android.com/tools/help/monkey.html, [Online; cit. 2015-01-10].

[16] Google: Activities. http://developer.android.com/guide/components/activities.html, 2015, [Online; cit. 2015-01-10].

[17] Google: Android SDK. http://developer.android.com/sdk/index.html, 2015, [Online; cit. 2015-01-10].

[18] Google: Fragments. http://developer.android.com/guide/components/fragments.html, 2015, [Online; cit. 2015-01-10].

[19] Google: Loaders. http://developer.android.com/guide/components/loaders.html, 2015, [Online; cit. 2015-01-10].

[20] Google: Services. http://developer.android.com/guide/components/services.html, 2015, [Online; cit. 2015-01-10].

[21] Google: View. http://developer.android.com/reference/android/view/View.html, 2015, [Online; cit. 2015-01-10].

[22] hackbod: Singletons vs. Application Context in Android? http://stackoverflow.com/questions/3826905/singletons-vs-application-context-in-android#comment8224394_38885605, [Online; cit. 2015-01-10].

[23] Jay Alabaster: Android founder: We aimed to make a camera OS. http://www.pcworld.com/article/2034723/android-founder-we-aimed-to-make-a-camera-os.html, [Online; cit. 2015-01-10].

[24] Kontakt.io: Bluetooth Beacon. https://store.kontakt.io/our-products/1-bluetooth-beacon.html, 2015, [Online; cit. 2015-03-16].

[25] kuuco: Bluetooth 4.0 BLE CC2540 iBeacon Module. http://www.ebay.com/itm/Bluetooth-4-0-BLE-CC2540-iBeacon-Module-/371241298110?roken=cUgayN&soutkn=Ri6jXD, 2015, [Online; cit. 2015-04-26].

[26] Mikhail Galeev: Make the most of Bluetooth LE advertising mode. http://www.eetimes.com/document.asp?doc_id=1280724, [Online; cit. 2015-01-10].

[27] Open Handset Alliance: Android Overview. http://www.openhandsetalliance.com/android_overview.html, [Online; cit. 2015-01-10].

[28] Pelant, M.: *Klient Redmine pro OS Android: Bakalářská práce*. ČVUT v Praze, Fakulta informačních technologií, 2013.

[29] Phil Smith: Comparing Low-Power Wireless Technologies. http://www.digikey.com/en/articles/techzone/2011/aug/comparing-low-power-wireless-technologies, [Online; cit. 2015-01-10].

[30] Quentyn Kennemer: Your Smartphone's Battery Gauge is Lying to You (and it's not such a bad thing). http://goo.gl/uO86Ym, [Online; cit. 2015-01-10].

[31] Renata SA: 3V Lithium Batteries. http://www.renata.com/consumer-products/3v-lithium-batteries, [Online; cit. 2015-01-10].

[32] Rob Price: Android sales are in decline for the first time ever. http://www.businessinsider.com/android-sales-in-decline-for-first-time-ever-2015-2, 2015, [Online; cit. 2015-03-16].

[33] Sibin P. Thomas: What is the difference between Bluetooth 4.0, 3.0 and 2.0? http://www.quora.com/What-is-the-difference-between-Bluetooth-4-0-3-0-and-2-0, [Online; cit. 2015-01-10].

PŘÍLOHA A

Seznam použitých zkratek

PaaS Platform as a service

BLE Bluetooth Low Energy

BT Bluetooth

UUID Universally unique identifier

ART Android runtime

IDE Integrated Development Environment

URI Uniform resource identifier

AIDL Android Interface Definition Language

rssi Received signal strength indication

XML Extensible markup language

API Application programming interface

UI User interface

MDPI Medium dots per inch

REST Representational state transfer

URL Uniform resource locator

OS Operating system

SMS Short Message Service

VM Virtual machine

USB Universal Serial Bus

ADB Android Debug Bridge

SDK Software development kit

SSL Secure Sockets Layer

HTTP Hypertext Transfer Protocol

HW Hardware

NoSQL Not only SQL

SQL Structured Query Language

ORM Object-relational mapping

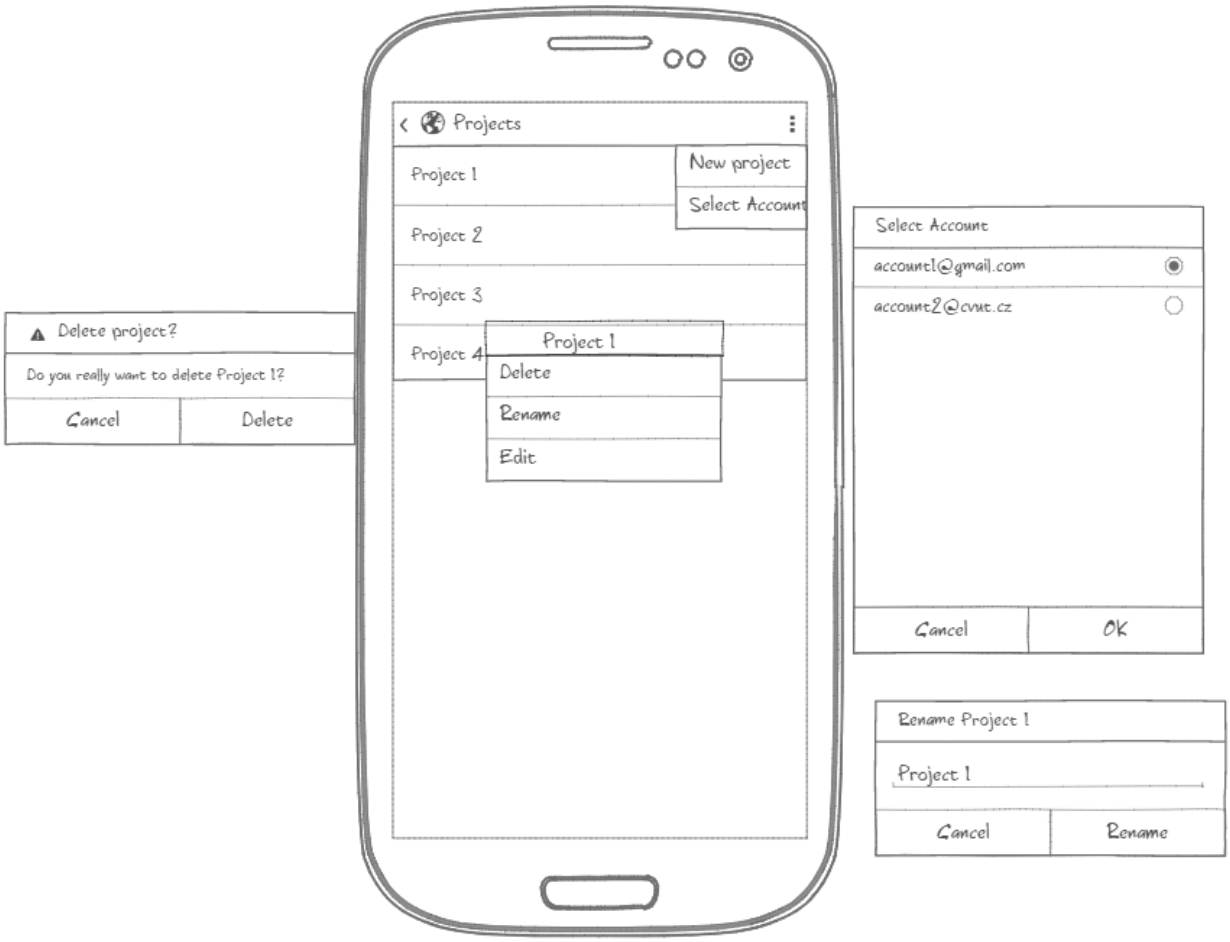
JSON JavaScript Object Notation

SVN Apache Subversion

PŘÍLOHA B

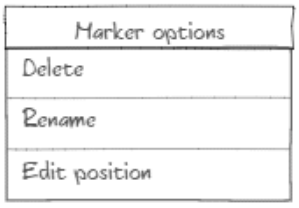
Návrhy obrazovek

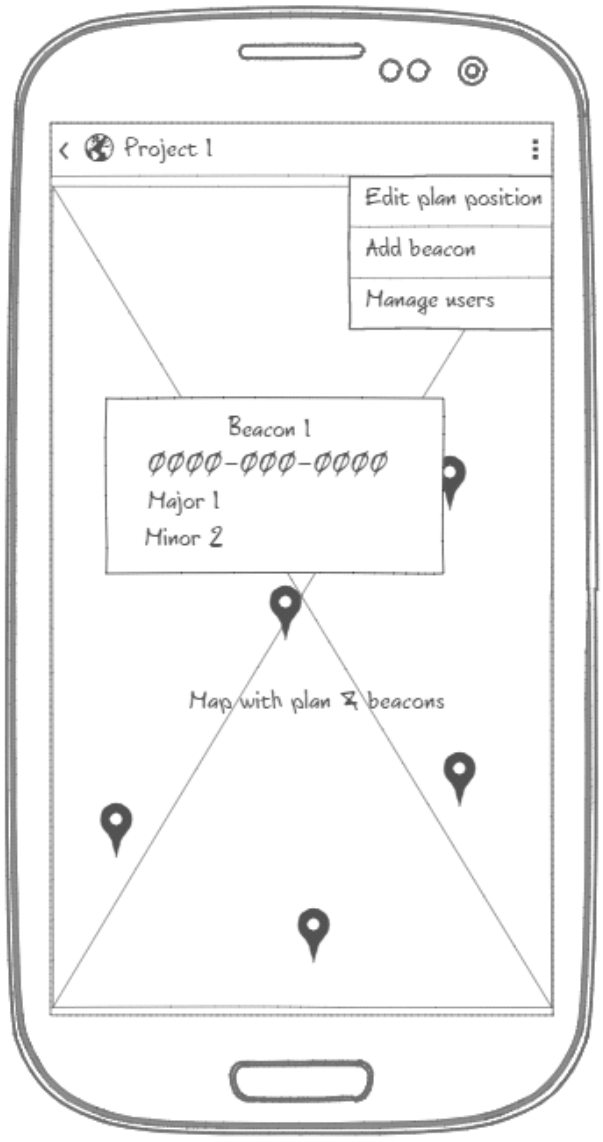
Project list



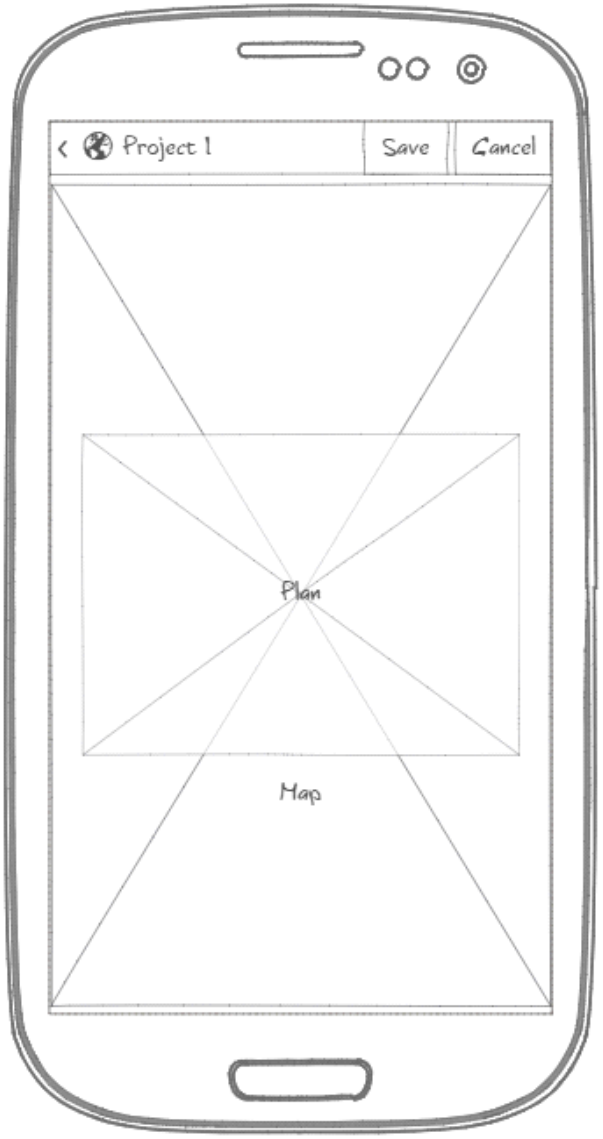
Wireframes Page 1

Project edit

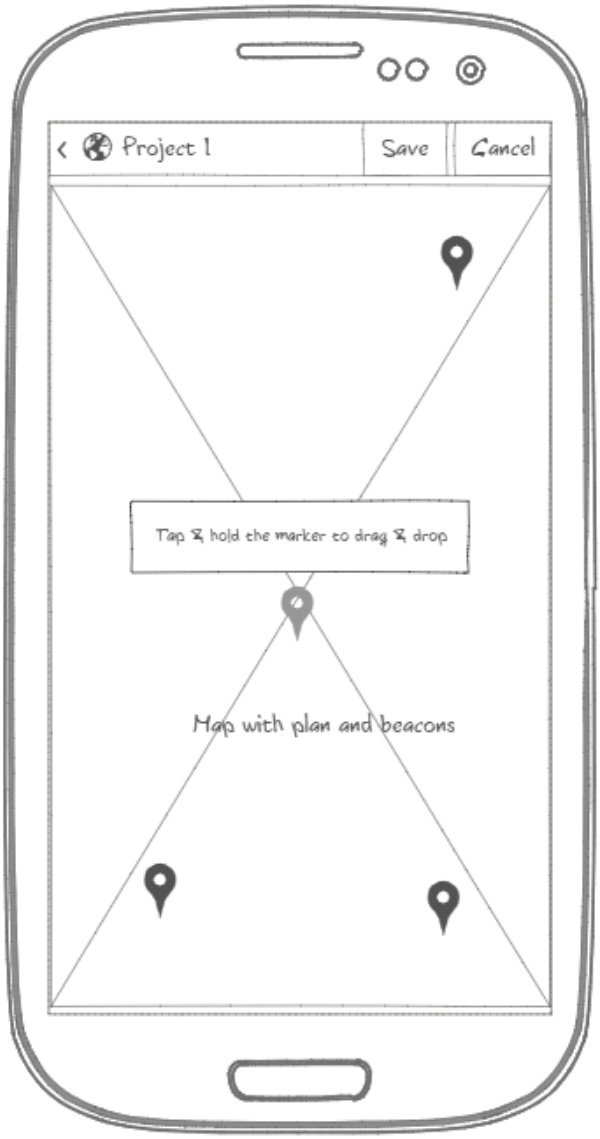




Project edit (plan)



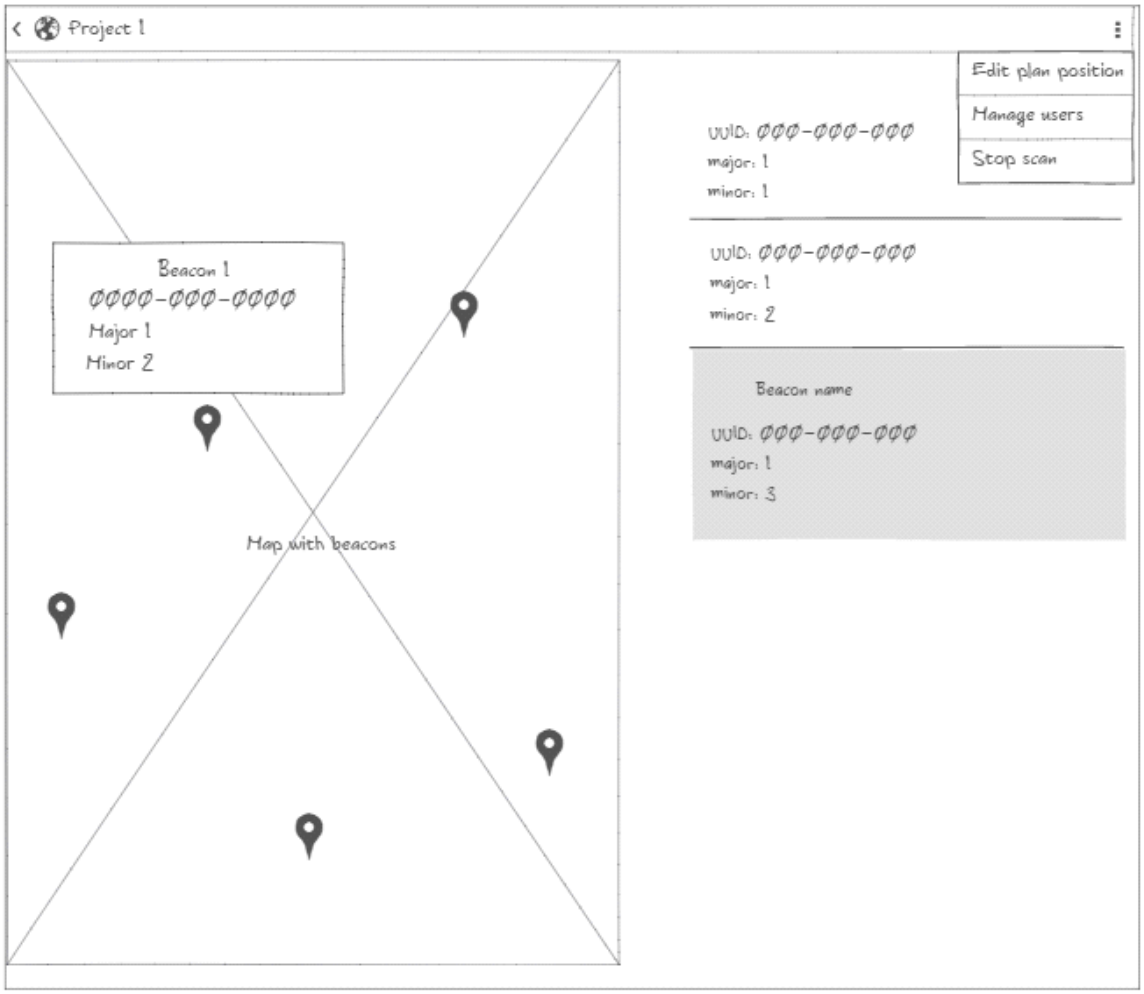
Project edit (marker position)



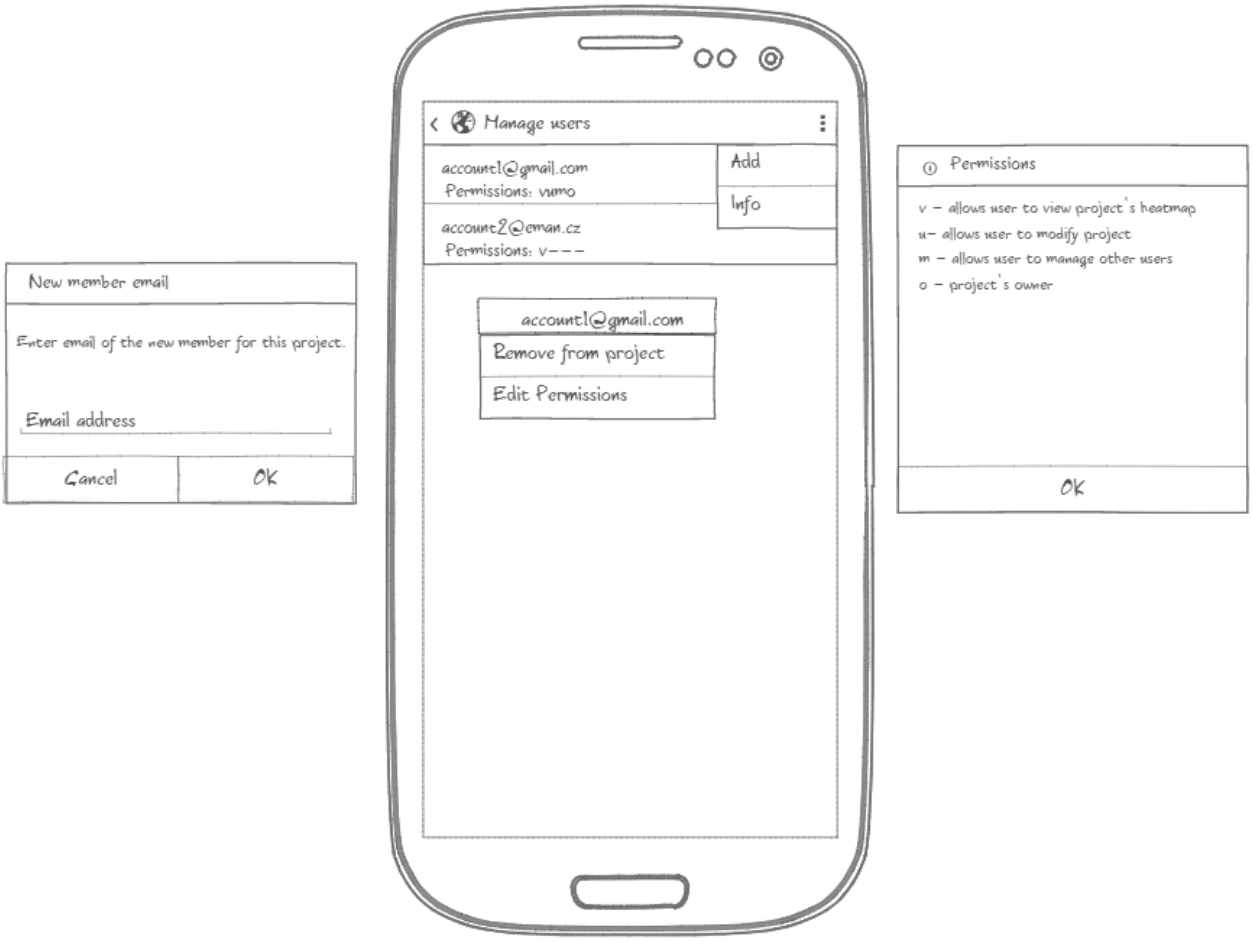
Add beacon

Add beacon
UUID: 000-000-000
major: 1
minor: 1
UUID: 000-000-000
major: 1
minor: 2
Beacon name
UUID: 000-000-000
major: 1
minor: 3
Set Beacon Name
UUID: 000-000-000
major: 1
minor: 3
Beacon name
No Name
Cancel
OK

Project edit (tablet)



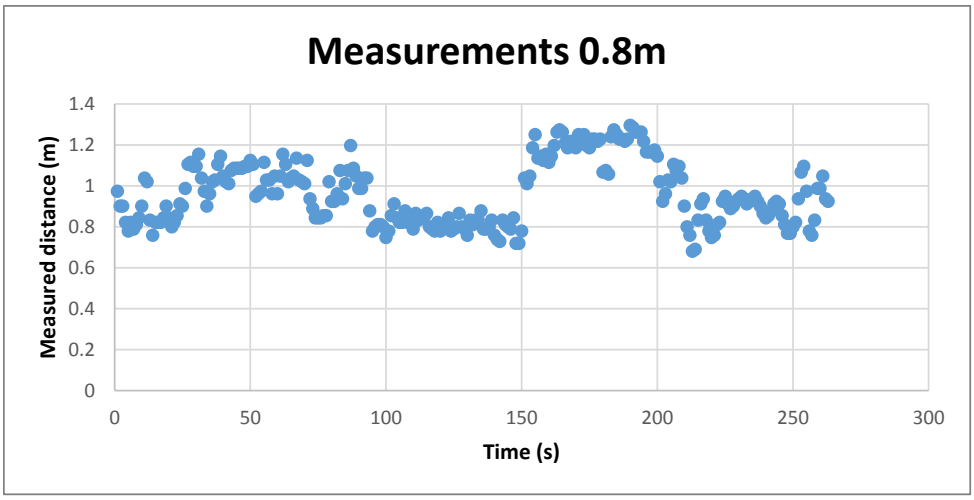
Manage users

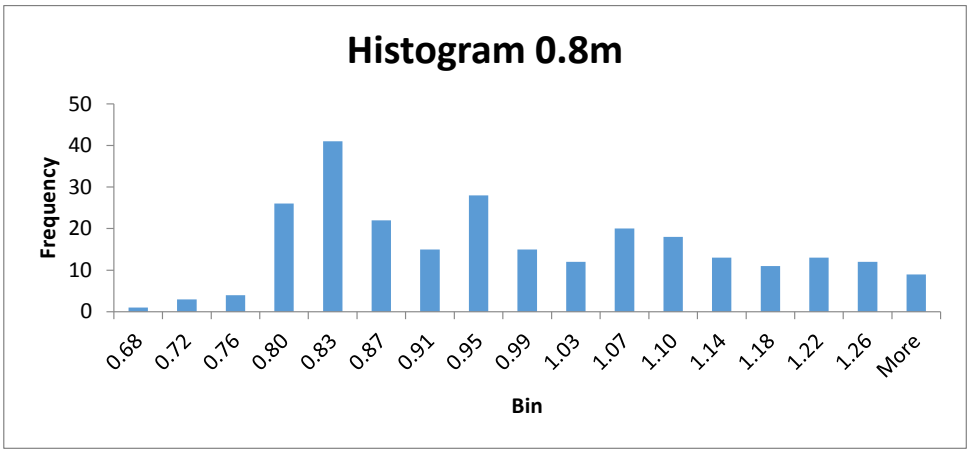


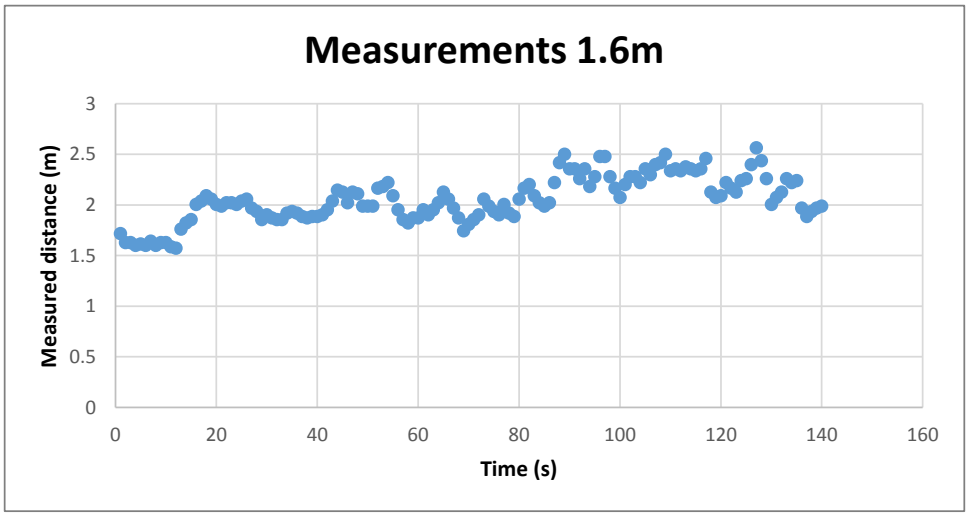
Project heatmap

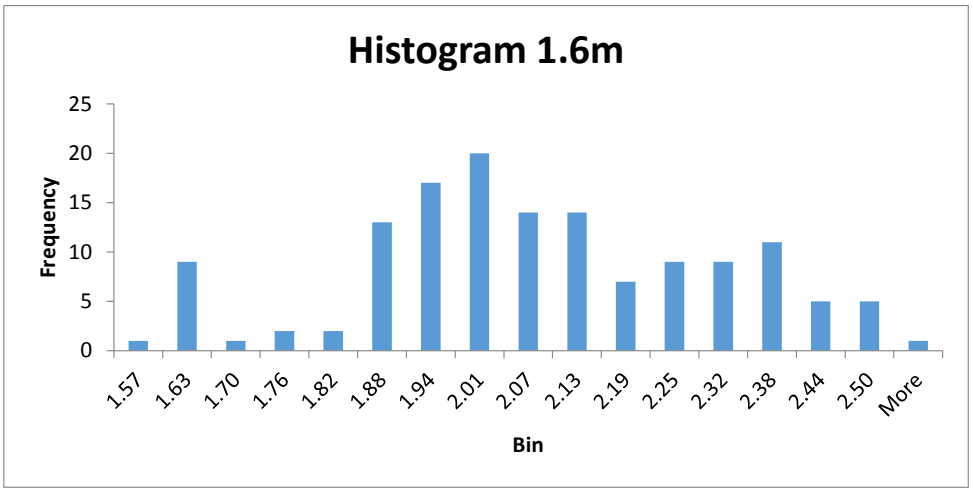
PŘÍLOHA C

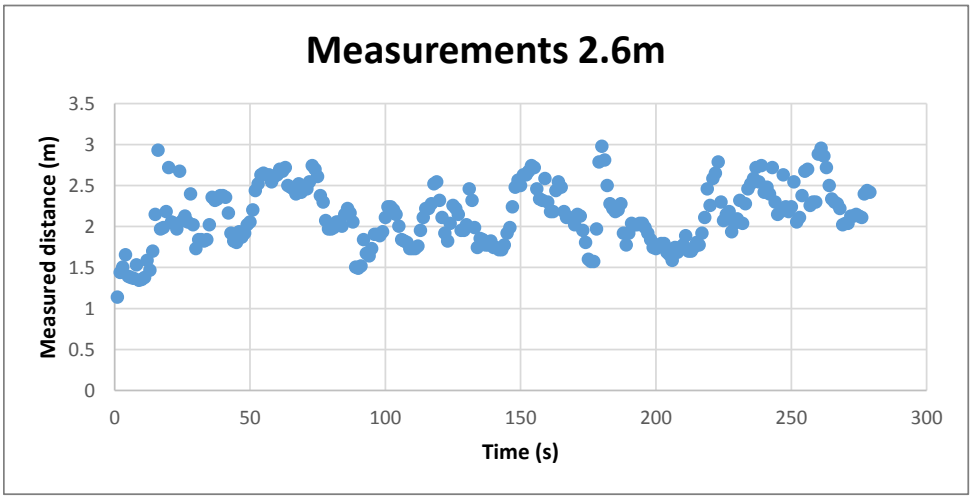
Výsledky měření vzdálenosti iBeaconů

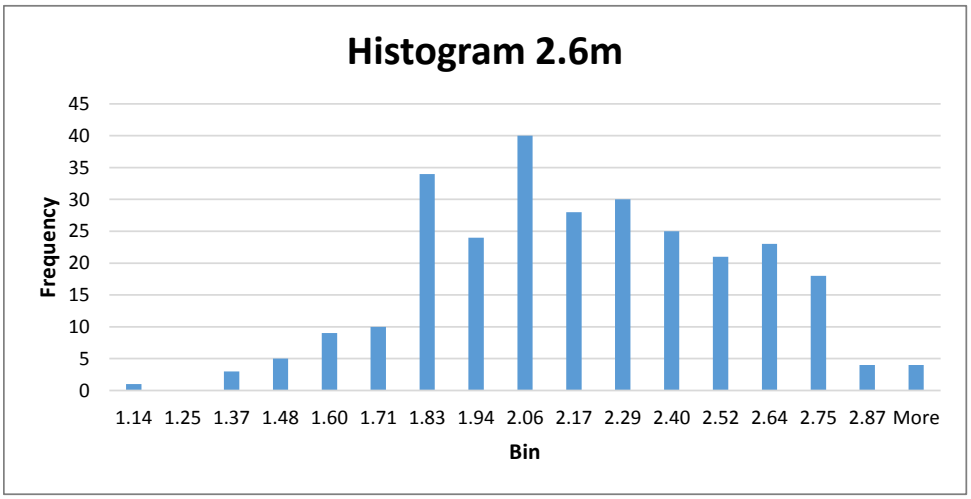

Measurements 0.8m
Measured distance (m)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
250
300
Time (s)


Histogram 0.8m
Frequency
50
40
30
20
10
0
0.68
0.72
0.76
0.80
0.83
0.87
0.91
0.95
0.99
1.03
1.07
1.10
1.14
1.18
1.22
1.26
More
Bin


Measurements 1.6m
Measured distance (m)
3
2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
120
140
160
Time (s)


Histogram 1.6m
Frequency
25
20
15
10
5
0
1.57
1.63
1.70
1.76
1.82
1.88
1.94
2.01
2.07
2.13
2.19
2.25
2.32
2.38
2.44
2.50
More
Bin


Measurements 2.6m
Measured distance (m)
3.5
3
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250
300
Time (s)


Histogram 2.6m
Frequency
45
40
35
30
25
20
15
10
5
0
1.14
1.25
1.37
1.48
1.60
1.71
1.83
1.94
2.06
2.17
2.29
2.40
2.52
2.64
2.75
2.87
More
Bin

PŘÍLOHA D

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `apk` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF
- `attachments` ..... adresář s přílohami
  - `app video.mp4` ..... ukázka funkčnosti administrátorské aplikace
  - `testing`
    - `location` ..... srovnávací video metod pro výpočet polohy
    - `usability` ..... záznam usability testu