ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Signals – Android aplikace pro sběr dat o bezdrátových sítích
Student:	Adam Novák
Vedoucí:	Ing. Miroslav Balík, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Vytvořte Android aplikaci, která bude sbírat informace o WiFi, mobilních sítích a aktuální poloze. Sběr dat se automaticky spustí na základě pohybu uživatele. Uživatel bude moci nasbíraná data nahrát na server, nebo exportovat pro vlastní potřebu. Server nahraná data zpracuje a vygeneruje statistiky a mapy pokrytí. Nasbíraná data budou také cílem různých úkolů (např. nasbírejte 50 nových WiFi). Pro uživatelskou autentizaci využijte Google Sign-In.

1.Analyzujte existující aplikace pro sběr dat o WiFi a mobilních sítích
2.Navrhněte a implementujte server
3.Navrhněte a implementujte Android aplikaci
4.Zajistěte bezpečnou komunikaci mezi Android aplikací a serverem
5.Publikujte aplikaci na Play Store
6.Otestujte aplikaci s uživateli a zhodnoťte výsledky testování

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 15. listopadu 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Signals - Android aplikace pro sběr dat o bezdrátových sítích

Katedra softwarového inženýrství
Vedoucí práce: Ing. Miroslav Balík, Ph.D.

11. května 2018

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Miroslavovi Balíkovi, Ph.D. za konzultace, cenné rady a motivaci k práci. Také bych chtěl poděkovat své rodině a kamarádům, kteří mě při studiu podporovali a motivovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) 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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2018 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2018 Adam Novák. 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 a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Novák, Adam. *Signals - Android aplikace pro sběr dat o bezdrátových sítích*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018. Dostupný také z WWW: ⟨`https://signals.adsamcik.com`⟩.

Abstrakt

Tato práce se zaměřuje na návrh a implementaci Android aplikace a její serverové části pro službu na sběr informací o Wi-Fi a mobilních sítích. Pro tvorbu Android aplikace byl využit programovací jazyk Kotlin a navržena dvě různá uživatelská rozhraní, na kterých bylo následně provedeno testování a jejich srovnání. Serverová část je vytvořena pro běh na procesorové architektuře ARM pomocí ASP.NET Core s využitím ACID NoSQL systému řízení báze dat RavenDB. Výsledkem práce je Android aplikace dostupná na Google Play a její serverová část běžící na Raspberry Pi 3.

Klíčová slova Android aplikace, Wi-Fi sítě, mobilní sítě, Kotlin, ASP.NET Core, C#, ARM, NoSQL

Abstract

This thesis focuses on design and implementation of an Android application and it's server part for collection of data about Wi-Fi and mobile networks. Android application has been written in Kotlin with 2 different user interfaces. Server part uses ASP.NET Core as web framework, RavenDB as NoSQL ACID database management system and it all runs on ARM based Raspberry Pi 3. Finally Android application was published to Google Play.

Keywords Android application, Wi-Fi networks, mobile networks, Kotlin, C#, ASP.NET Core, ARM, NoSQL

Obsah

Úvod **1**

1 Podobná řešení **3**
1.1 WiGLE . . . 3
1.2 OpenSignal . . . 4
1.3 Wifileaks . . . 6
1.4 NetMonster . . . 6
1.5 Souhrn . . . 7

2 Analýza **9**
2.1 Funkční a nefunkční požadavky Android aplikace . . . 9
2.2 Funkční a nefunkční požadavky serveru . . . 10
2.3 Případy užití . . . 10

3 Návrh **13**
3.1 Diagram aktivit . . . 13
3.2 Android . . . 16
3.3 Server . . . 17
3.4 Uživatelské rozhraní Android aplikace . . . 24

4 Implementace **31**
4.1 Android aplikace . . . 32
4.2 Server . . . 38

5 Testování **45**
5.1 Testování vývojářem . . . 45
5.2 Testování uživateli . . . 45
5.3 Hlášení chyb . . . 47

Závěr **49**

Literatura **51**

A Uživatelská a instalační příručka **55**
A.1 Uživatelská příručka 55
A.2 Instalační příručka 55

B Seznam použitých zkratek **57**

C Vývojové nástroje **59**

D Obsah přiloženého CD **61**

Seznam obrázků

1.1 Snímky obrazovky aplikace WiGLE 4
1.2 Snímky obrazovky aplikace OpenSignal 5
1.3 Snímky obrazovky aplikace Wifileaks 6
1.4 Snímky obrazovky aplikace NetMonster 7

2.1 Případy užití pro Android aplikaci 11

3.1 Diagram aktivit nahrávacího procesu v Android aplikaci 14
3.2 Diagram aktivit procesu zpětné vazby na serveru 15
3.3 Zastoupení verzí Androidu 16
3.4 Diagram tříd ukládaných do databáze 19
3.5 Snímky obrazovky standardního uživatelského rozhraní 26
3.6 Snímky obrazovky zobrazující problematickou bílou barvu pozadí . 28
3.7 Snímky obrazovky různých barev pozadí experimentálního uživatelského rozhraní 29
3.8 Snímky obrazovky experimentálního uživatelského rozhraní 30

4.1 Diagram síťové komunikace serveru a Android aplikace 31
4.2 Snímky obrazovky tutoriálů, vytvořených v rámci aplikace 37
4.3 Obrázek znázorňující podpis aplikace lokálním klíčem [10]. 38
4.4 Obrázek znázorňující podpis aplikace klíčem na Google Play [10]. . 38
4.5 Komunikaci mezi částmi MVC v ASP.NET Core [17]. 40
4.6 Příklad jedné dlaždice překryvné vrstvy mapových podkladů . . . 41

Úvod

V dnešním světě se stále zvyšuje počet Wi-Fi sítí a lidé si často ani neuvědomují, že velký počet Wi-Fi sítí na jednom místě se navzájem ruší. Stále častější je také očekávání dostupnosti rychlého mobilního internetu všude v republice. Mobilní operátoři toho využívají a v rámci reklamy publikují mapy dostupnosti, které jsou založené na matematických modelech a nereprezentují skutečnou dostupnost jejich sítě. Často se tak může stát, že podle operátora je signál v daném místě dostupný, přestože opak je pravdou.

Přestože existuje mnoho služeb zabývajících se touto problematikou, tak žádná z nich nenabízí přehled vytížení spektra pro Wi-Fi sítě, nebo automatickou aktivaci sběru dat v mobilní aplikaci. Jednou z hlavních motivací pro tvorbu této práce byla možnost vyzkoušet si mnoho nových technologií, které se v praxi zatím nevyužívají.

Cílem této práce je vytvořit službu, která bude uživatelům poskytovat přehled o dostupnosti mobilního signálu, vytížení Wi-Fi spektra a celkové statistiky o Wi-Fi a mobilních sítích. Android aplikace bude sloužit jako primární uživatelské rozhraní a platforma pro sběr dat. Serverová část bude sloužit pro zpracování dat z Android aplikace a následné vygenerování map pokrytí a statistik.

Kapitola 1

Podobná řešení

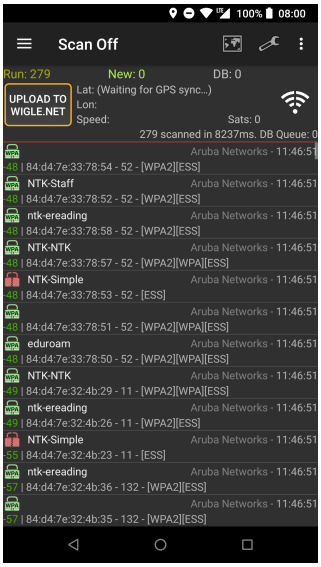
Na Google Play lze nalézt mnoho řešení zabývajících se analýzou Wi-Fi a mobilních sítí. Pro analýzu byla vybrána 4 řešení, která nejlépe odpovídají zvoleným cílům této aplikace.

1.1 WiGLE

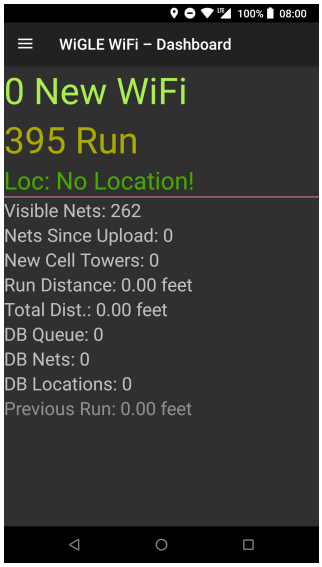
WiGLE (Wireless Geographic Logging Engine) je služba sbírající informace o Wi-Fi sítích. Byla založena roku 2001 a od svého založení má ve své databázi více než 400 milionů Wi-Fi sítí, 5 miliard záznamů a 8 milionů mobilních vysílačů. Ze všech analyzovaných služeb je nejblíže této aplikaci. Služba nabízí svým uživatelům mapu Wi-Fi sítí, statistiky a Android aplikaci. Služba se zaměřuje primárně na Wi-Fi sítě. Mapa pro mobilní sítě služba nenabízí a ve statistikách lze nalézt pouze jejich počet. Její hlavní předností je množství dat, které služba naakumulovala za dobu své existence.

Doba existence této služby je ale také její nevýhodou. Vzhled Android aplikace i přes zjevný pokus o aktualizaci na prvky Material Designu působí zastarale a v určitých částech aplikace není příliš uživatelsky přívětivá. Služba sice sbírá data o mobilních vysílačích, kromě statistiky jejich počtu s nimi však nijak dále veřejně nepracuje. Android aplikace má otevřený zdrojový kód, dostupný na službě GitHub.

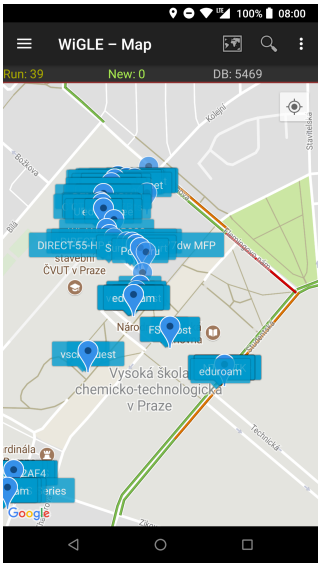
Služba nabízí API, ale s velmi omezenou funkčností.



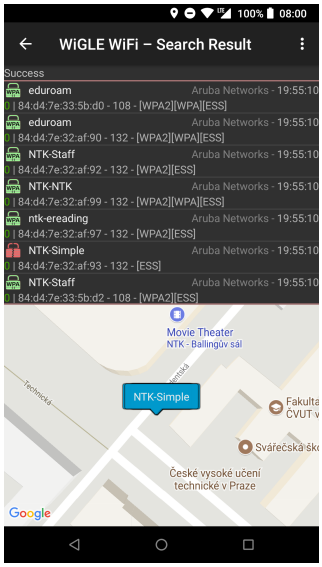
(a) Seznam Wi-Fi



(b) Dashboard



(c) Mapa Wi-Fi sítí



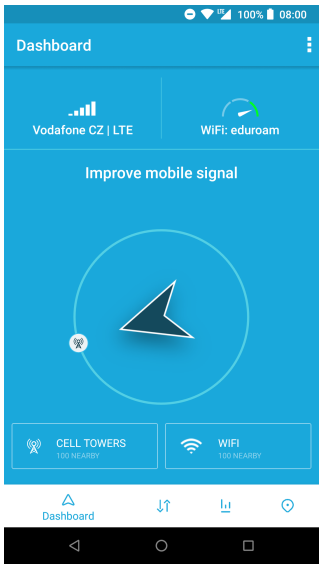
(d) Vyhledávání Wi-Fi v databázi

Obrázek 1.1: Snímky obrazovky aplikace WiGLE

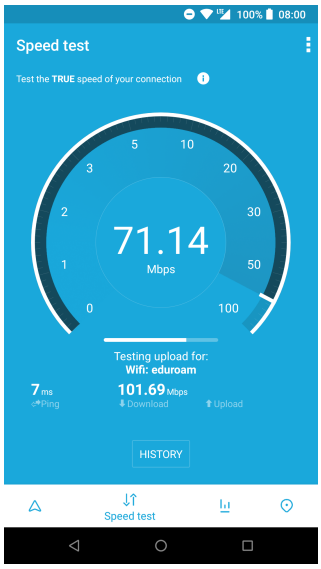
1.2 OpenSignal

OpenSignal provozuje stejnojmenná společnost a byl založen v roce 2010. Služba se primárně zaměřuje na mobilní sítě, jejich pokrytí a rychlost. K dispozici jsou aplikace pro Android i iOS. V mobilní aplikaci je k dispozici databáze Wi-Fi sítí. Ovšem tato databáze je vytvořena s pomocí uživatelů, kteří do ní vložili dané Wi-Fi sítě a jediné dostupné informace o těchto Wi-Fi sítích jsou

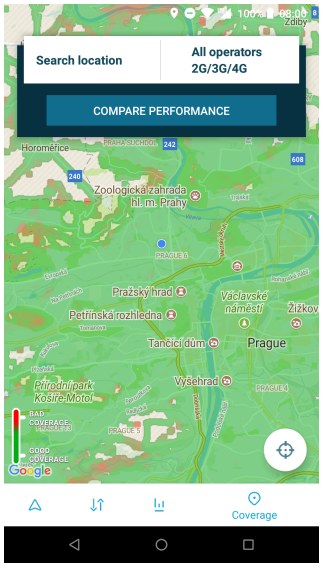
název, zda mají heslo a počet připojení. Android aplikace nabízí možnost automatického sběru dat, který ovšem není závislý na aktivitě, a jeho frekvence záleží na uživatelském nastavení, ze kterého není zcela jasné, jak funguje.



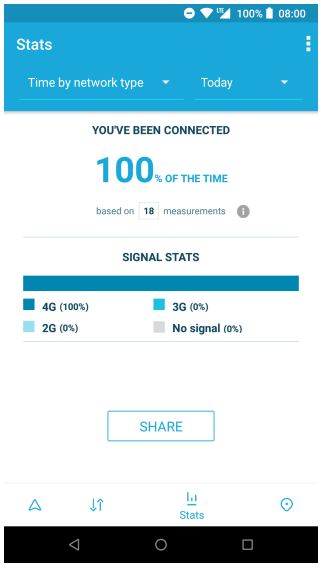
(a) Domovská stránka



(b) Měření rychlosti



(c) Mapa kvality signálu



(d) Mapa kvality signálu

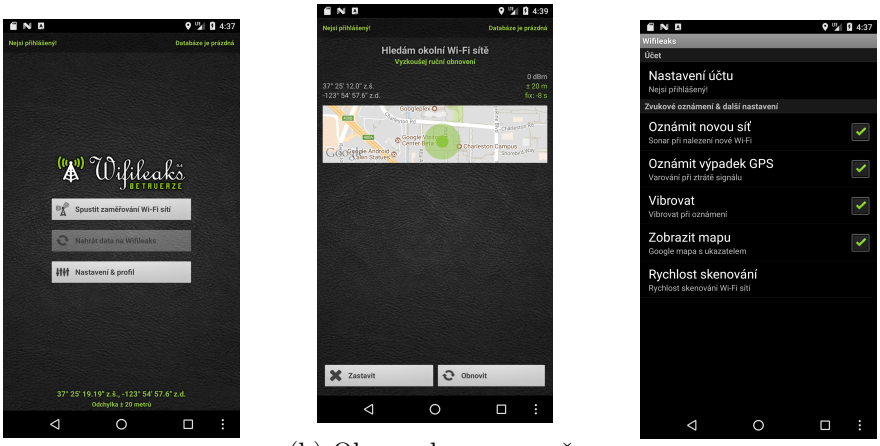
Obrázek 1.2: Snímky obrazovky aplikace OpenSignal

Android aplikace má moderní vzhled, který je založený na Material Designu. Vzhled využívá plovoucího tlačítka pro aktuální polohu a dolní navigace pro celkovou orientaci v aplikaci. Aplikace má jak tmavou, tak světlou verzi.

1.3 Wifileaks

Wifileaks je česká služba na zaznamenávání Wi-Fi sítí a odhadování jejich polohy. Vytvořil ji redaktor Živě.cz Jakub Čížek. Aplikace není již mnoho let aktualizována, ale byla zahrnuta v analýze, protože její uživatelé jsou stále aktivní.

Služba je velmi jednoduchá a nabízí pouze přehled nejčastěji vyskytujících se zabezpečení a názvů Wi-Fi sítí. Služba využívá záznamů s nejsilnějším signálem pro výpočet polohy Wi-Fi sítí. Její hlavní předností je otevřená databáze, která je každý měsíc publikována ve formátu PHP (PHP: Hypertext Preprocessor).

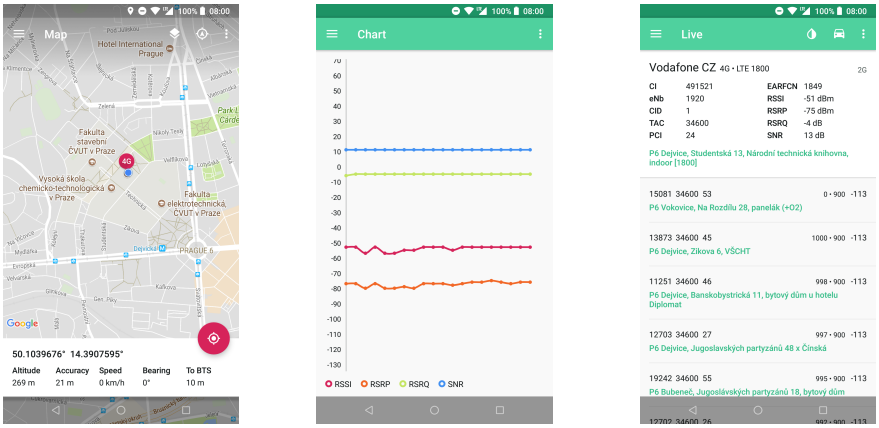


(a) Úvodní obrazovka (b) Obrazovka pro zaměřování Wi-Fi sítí (c) Nastavení

Obrázek 1.3: Snímky obrazovky aplikace Wifileaks

1.4 NetMonster

NetMonster je Android aplikace, zaměřená na zobrazování informací o mobilních sítích. Vytvořil ji český vývojář Michal Mroček. Aplikace využívá pro Českou republiku databázi GSMweb.cz, která obsahuje informace o jednotlivých vysílačích a jejich přesné poloze. Aplikace ukládá veškerá data lokálně, bez možnosti je nahrát na server.



(a) Mapa vysílačů (b) Mobilní signál v čase (c) Vysílače v dosahu

Obrázek 1.4: Snímky obrazovky aplikace NetMonster

1.5 Souhrn

Existuje několik služeb, zabývajících se sběrem informací o Wi-Fi a mobilních sítích, s celosvětovým pokrytím. Každá služba se vždy zaměřuje buď na Wi-Fi, nebo na mobilní sítě, nikdy ne na obě. Služby, zaměřující se na mobilní sítě, mají modernější uživatelské rozhraní než služby zaměřující se na Wi-Fi. Žádná z analyzovaných aplikací nenabízí automatický sběr dat podle aktivity a jediná aplikace OpenSignal je schopná automatického sběru dat.

Kapitola 2

Analýza

V této kapitole se podíváme na případy užití a funkční a nefunkční požadavky pro Android aplikaci a server.

2.1 Funkční a nefunkční požadavky Android aplikace

2.1.1 Funkční požadavky

F1 Aplikace umožní uživateli automatický sběr dat podle aktuální aktivity uživatele.

F2 Aplikace bude podporovat automatické nahrávání dat na měřeném, nebo neměřeném přístupu k internetu, dle uživatelova nastavení.

F3 Aplikace umožní uživateli export dat pro vlastní potřebu.

F4 Aplikace bude podporovat sběr dat o Wi-Fi a mobilních sítích.

F5 Aplikace bude podporovat zobrazení statistik ze serveru.

F6 Aplikace umožní uživateli zobrazení aktuální polohy na mapě.

F7 Aplikace umožní uživateli zobrazení map pokrytí.

F8 Aplikace umožní uživateli sledování aktuálních herních výzev a postupu v nich.

2.1.2 Nefunkční požadavky

N1 Aplikace musí fungovat na operačním systému Android verze 5.0 a novějším.

N2 Aplikace nesmí uživateli zobrazovat ANR dialog.

N3 Aplikace musí být schopna načítat skutečná data ze serveru.

N4 Aplikace musí podporovat více jazyků pro alespoň lokální data.

2.2 Funkční a nefunkční požadavky serveru

2.2.1 Funkční požadavky

F1 Server bude ověřovat uživatele pomocí Google přihlášení.

F2 Server musí umět tvorbu překrytí mapy.

F3 Server bude umožňovat uživatelům nákup služeb za virtuální měnu.

F4 Server bude týdně generovat nové náhodné herní výzvy.

F5 Server musí umět generovat statistiky.

F5 Server bude udržovat uživatelské statistiky pro jednotlivé uživatele.

2.2.2 Nefunkční požadavky

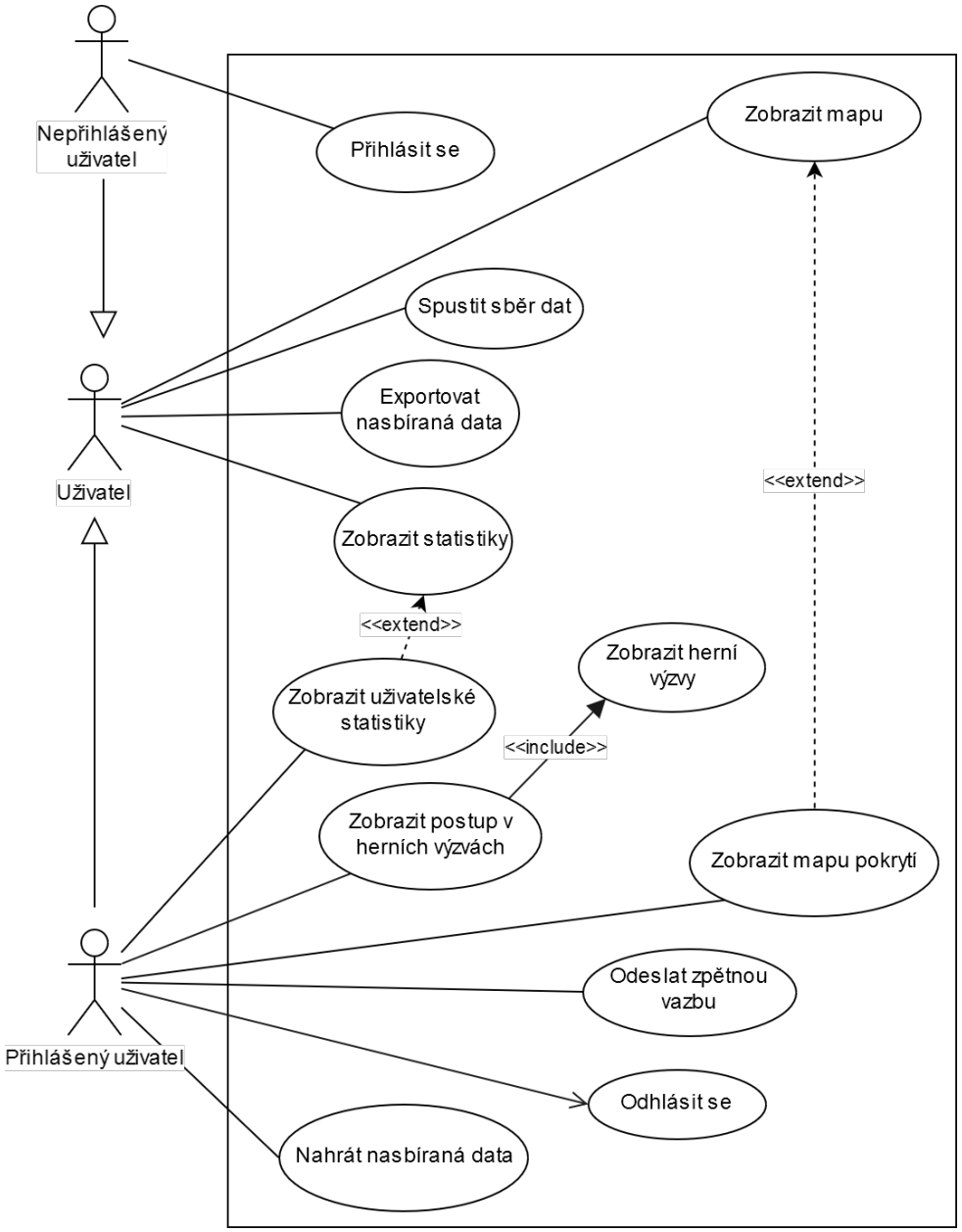
N1 Překryvné vrstvy pro mapu musí být implementovány s využitím Google Maps.

N2 Server musí fungovat na Raspberry Pi 3 Model B.

N3 Server musí využívat pro komunikaci zabezpečené spojení.

2.3 Případy užití

V této části je popsána interakce uživatele s Android aplikací. Pro popsání této interakce je využit diagram případů užití na obrázku 2.1.



Obrázek 2.1: Případy užití pro Android aplikaci

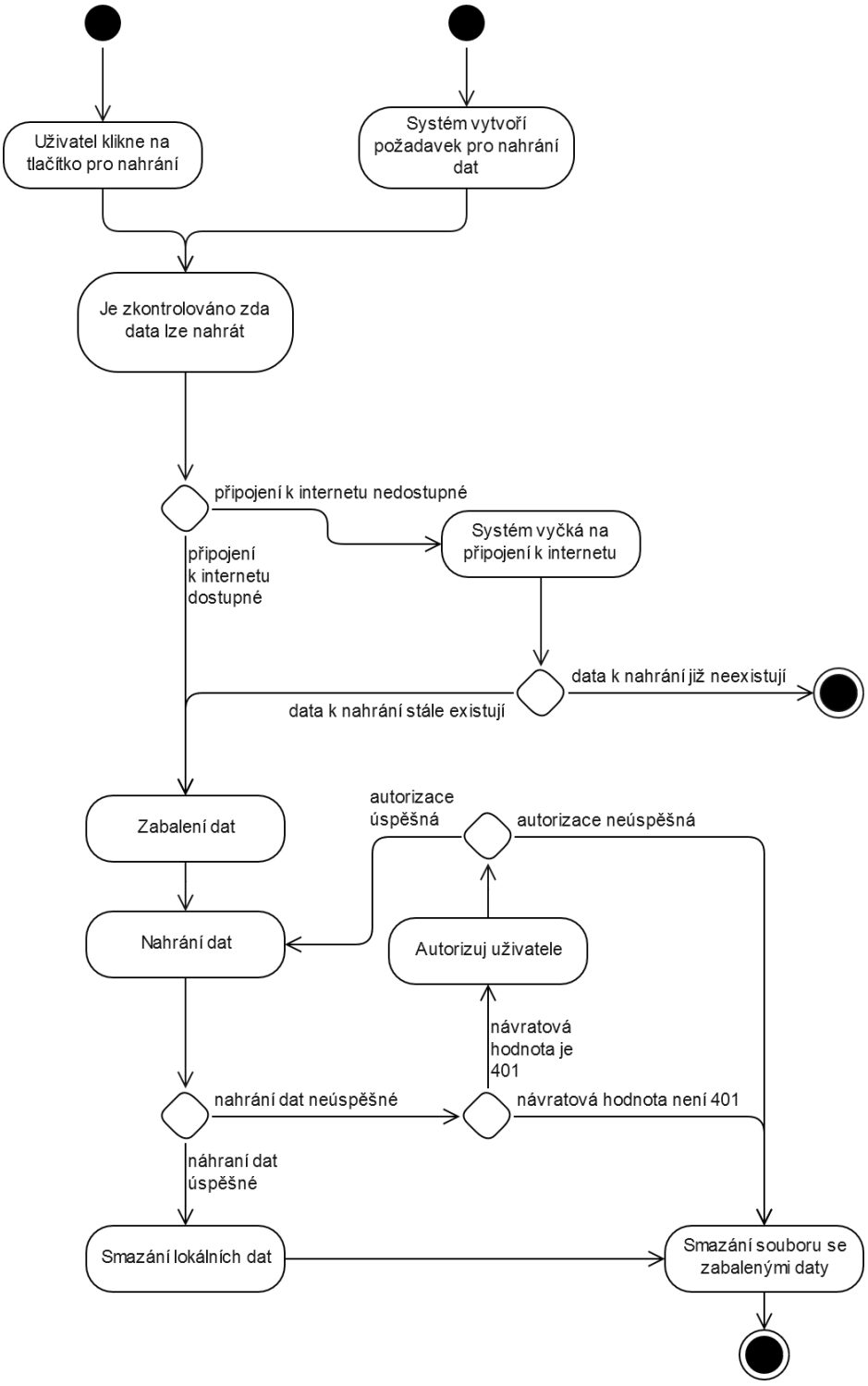
Kapitola 3

Návrh

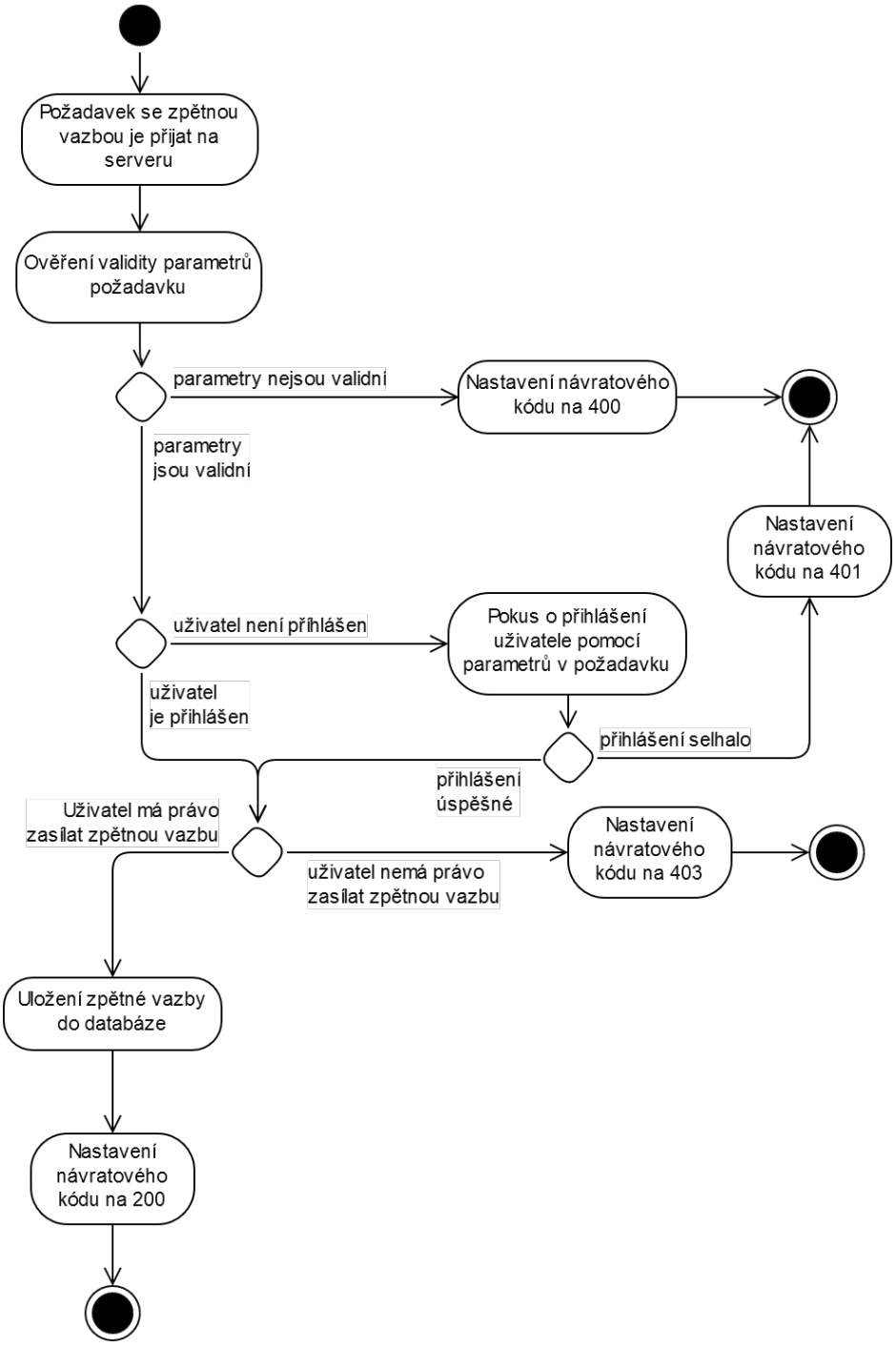
V této kapitole se budeme zabývat diagramy aktivity pro nahrávací proces v Android aplikaci a procesu zpětné vazby na serveru. U Android aplikace se budeme zabývat návrhem dvou uživatelských rozhraní, výběrem minimální podporované verze Androidu a ukládáním dat. U serveru bude proveden výběr platformy, výběr databáze a návrh API pro komunikaci s Android aplikací.

3.1 Diagram aktivit

V této části se budeme zabývat diagramy aktivit. Na obrázku 3.1 je diagram aktivit nahrávacího procesu.



Obrázek 3.1: Diagram aktivit nahrávacího procesu v Android aplikaci



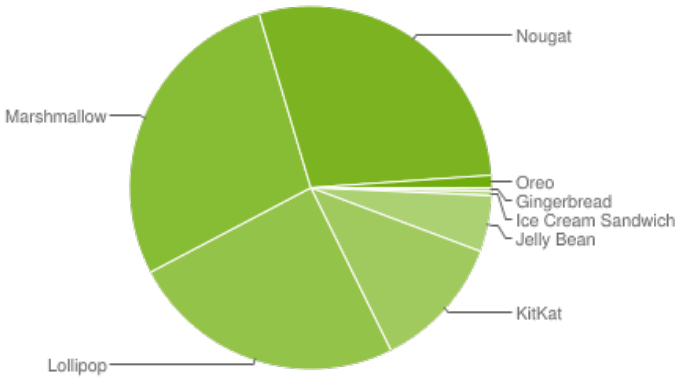
Obrázek 3.2: Diagram aktivit procesu zpětné vazby na serveru

3.2 Android

3.2.1 Minimální podporovaná verze operačního systému

Platforma Android byla jasnou volbou pro mobilní aplikaci. Vývojové nástroje jsou k dispozici zdarma a jednorázový poplatek za možnost publikace na Google Play je také přijatelný. Android má také většinové zastoupení na trhu a autor již vlastní několik zařízení s tímto operačním systémem.

Vybrat minimální verzi je složitější problém. Každá verze Androidu má jinou verzi API, které nemusí být vzájemně kompatibilní. Čím tedy více verzí aplikace podporuje, tím je pravděpodobnější, že bude nutné některé části aplikace vytvořit několikrát pro různé verze API. Bohužel je Android velmi fragmentovaný systém a nejnovější verze využívá vždy jen zlomek celkového počtu uživatelů.



Obrázek 3.3: Zastoupení verzí Androidu k 26. březnu 2018 [1]

Tvůrci Androidu doporučují podporovat alespoň 90 % aktivních zařízení [1]. V tomto případě by to znamenalo podporovat Android 4.4 a novější. Jako minimální podporovaná verze byl však vybrán Android 5 (API 21). API 21 a novější je aktuálně dostupné na 82 % zařízení. Důvodem podpory menšího, než doporučeného množství zařízení je, že Android 5 s sebou přináší velké množství změn, Material Design a relativně malé množství změn v API oproti Androidu 8.1. Pro Android 4.4 by bylo potřeba vytvořit mnoho specifického kódu a aplikaci by bylo mnohem složitější v budoucnu udržovat.

3.2.2 Ukládání dat

Ukládání nasbíraných dat je velmi důležitou částí aplikace a musí být rychlé a odolné proti chybám. Po zhodnocení mnoha možností zůstaly pouze dva způsoby ukládání: ukládání do databáze a ukládání do souboru ve formátu JSON. Nakonec byl vybrán formát JSON, protože je rychlejší a má nenarůstající nízké nároky na výpočetní výkon a tím i baterii.

JSON je otevřený, standardizovaný, člověkem čitelný souborový formát. Nové záznamy v tomto formátu lze přidávat přímo do již existujícího souboru bez nutnosti jeho čtení a díky tomu nebude zápis zbytečně zpomalován. Tento způsob zapisování nových záznamů ovšem představuje riziko syntaktických chyb ve finálním souboru, které ovšem lze snadno opravit buď ručně, nebo automaticky na serveru. Formát tedy splňuje odolnost vůči chybám. Díky své rozšířenosti je formát také jedním z příjemnějších pro technicky zdatnější uživatele, kteří by chtěli využít data pro vlastní potřebu.

3.3 Server

3.3.1 Volba platformy

V této části porovnáme jednotlivé platformy pro server.

3.3.1.1 Webhosting

Většina důvěryhodných webhostingových služeb v době výběru platformy nabízela převážně pouze podporu PHP. Výhodou tohoto řešení jsou převážně nulové pořizovací náklady a nízké náklady na provoz. Nejlepším poskytovatelem v době výběru byl WEDOS internet, který nabízel přibližně za 30 Kč na měsíc webhosting s neomezeným diskovým prostorem pro web. Po prvních průzkumech bylo ale jasné, že PHP na webhostingu nebude dostatečné řešení. Zpracování souborů v tomto rozhraní je velmi pomalé a často naráží na časový limit.

3.3.1.2 Virtuální privátní server

Virtuální privátní server, také známý jako VPS (Virtual Private Server) je server, který se neváže na fyzický stroj. Díky tomu je zaručena veliká škálovatelnost. Výhodou oproti dedikovaným serverům je také nižší cena, která je docílena právě tím, že poskytovatelé mohou lépe využívat zdrojů fyzických strojů. Nevýhodou tohoto řešení je značně vyšší cena na provoz oproti webhostingu. Nejlepší nabídku počátkem roku 2017 pro tuto aplikaci měl DigitalOcean, který za 5 dolarů měsíčně nabízel 512 MB RAM, 1 procesorové jádro, 20 GB SSD disk a 1TB na přenos dat.

3.3.1.3 Raspberry Pi 3 model B

Raspberry Pi 3 model B (zkráceně pouze Raspberry Pi 3, nebo RPi3) je malý, levný počítač, který obsahuje pouze nutné části počítače. Primárním trvalým uložištěm tohoto malého počítače je microSD, které je svou přenosovou rychlostí značně pomalejší, než HDD nebo SSD, které se dnes využívají v počítačích. Raspberry Pi 3 je vybaveno 64-bitovým procesorem s architekturou

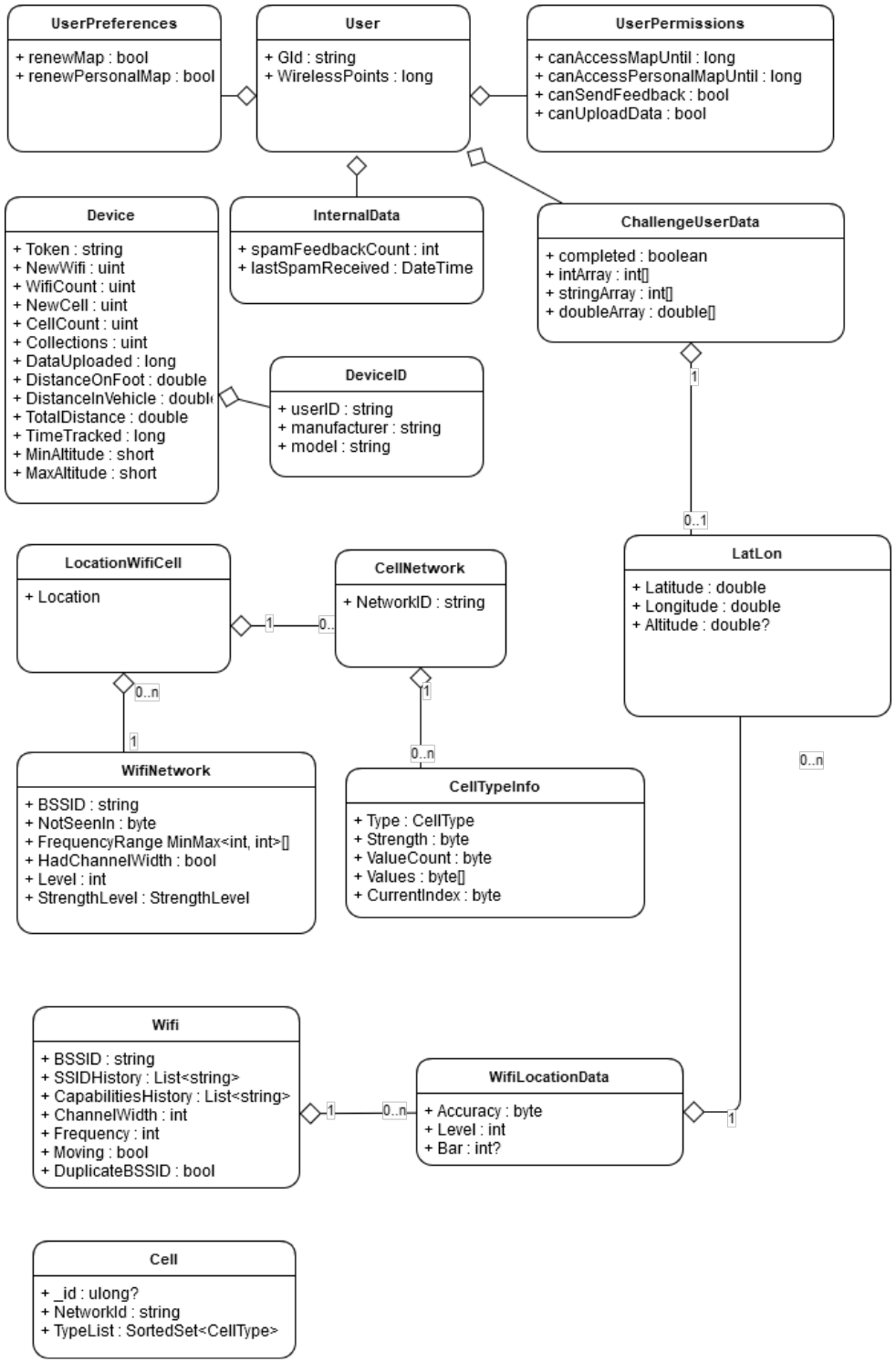
ARM, který je hlavní překážkou pro většinu řešení. Přestože se jedná o 64-bitový procesor, oficiální operační systém je dostupný pouze v 32-bitové verzi. Jedním z důvodů je, že tento procesor nebyl vybrán pro svoji 64 bitovou architekturu, ale protože je to skvělý 32-bitový procesor [2]. Naštěstí na začátku roku 2018 všechny aplikace, nutné pro provoz této aplikace, podporují 32-bitový Linux na architektuře ARM32.

3.3.2 Databáze

Databáze tvoří jednu z hlavních částí serverové aplikace. Je tedy nutné vybrat SŘBD, který splňuje veškeré požadavky. V případě této aplikace se jedná o rychlost, stabilitu a podporu 32 bitové procesorové architektury ARM. Dle osobní preference omezil autor výběr pouze na NoSQL systémy. Nejpopulárnější NoSQL SŘBD je MongoDB [3]. MongoDB bohužel nabízí oficiální podporu pouze pro 64 bitový ARM a bylo tedy nutné hledat dále. Nakonec byl vybrán SŘBD RavenDB. Hlavním důvodem pro tuto volbu byla oficiální podpora Raspberry Pi od nové verze 4.0, vydané v únoru 2018. RavenDB je transakční NoSQL SŘBD, který z definice transakce splňuje ACID. Data jsou v RavenDB ukládána ve formátu JSON a pro veškeré dotazy spoléhá RavenDB na asynchronní indexaci [4].

3.3.2.1 Databázový model

RavenDB neobsahuje žádné schéma. To znamená, že data lze ukládat v libovolném tvaru [5]. Vzhledem k tomu, že jsou do databáze přímo serializovány C# objekty, byl využit diagram tříd, který obsahuje pouze proměnné, které jsou ukládány do databáze. Diagram lze nalézt na obrázku 3.4.



Obrázek 3.4: Diagram tříd ukládaných do databáze

3.3.3 Serverová API pro Android aplikaci

Android aplikace komunikuje se serverovou aplikací pomocí neveřejného API. Jelikož je toto API neveřejné, byl zvolen nejednotný formát návratových hodnot, který je optimalizován pro množství přenesených dat a nižší náročnost na serializaci a deserializaci.

GET /data/stats.json
Požadavek slouží k stažení statistik.

Stavové kódy

- **200** – při úspěchu

Návratová hodnota předem vygenerované statistiky ve formátu JSON

GET /data/generalStats.json
Požadavek slouží pro stažení obecných statistik.

Stavové kódy

- **200** – při úspěchu

Návratová hodnota předem vygenerované obecné statistiky ve formátu JSON

POST /upload
Požadavek slouží pro nahrání dat z Android aplikace na server.

Parametry

- **soubor** ve formátu zip obsahující komprimované soubory s daty

Stavové kódy

- **200** – při úspěchu
- **400** – v případě chyby
- **401** – pokud uživatel není přihlášen

Návratová hodnota požadavek vrací pouze stavové kódy.

GET /map/{zvětšení}/{x}/{y}/{typ}
Požadavek slouží k získání dlaždic mapových podkladů dle tvaru adresy.

Stavové kódy

- **200** – při úspěchu
- **401** – pokud uživatel není přihlášen, nebo nemá přístup k mapovým podkladům
- **404** – pokud dlaždice neexistuje

Návratová hodnota mapovou dlaždici ve formátu PNG

GET /map/personal/{zvětšení}/{x}/{y}
Požadavek slouží pro získání dlaždic osobních mapových podkladů dle zadané polohy a přiblížení.

Stavové kódy

- **200** – při úspěchu
- **401** – pokud uživatel není přihlášen, nebo nemá přístup k osobním mapovým podkladům
- **404** – pokud dlaždice neexistuje

Návratová hodnota mapová dlaždice ve formátu PNG odpovídající souřadnicím „x“, „y“ a přiblížení „zvětšení“.

GET /map/Available
Požadavek slouží pro získání informací o dostupných mapových vrstvách a jejich mezím, které jsou lépe známy z angličtiny jako „bounds“.

Stavové kódy

- **200** – při úspěchu
- **401** – pokud uživatel není přihlášen, nebo nemá přístup k mapovým

Návratová hodnota dostupné mapové podklady s jejich mezemi jako objekty ve formátu JSON.

GET /user/Info
Požadavek slouží k získání informací o uživatelských právech, preferencích a virtuálních bodech přihlášeného uživatele.

Stavové kódy

- **200** – při úspěchu
- **401** – pokud uživatel není přihlášen

Návratová hodnota objekt ve formátu JSON obsahující informace o právech, preferencích a bezdrátových bodech přihlášeného uživatele.

GET /user/Prices
Požadavek slouží k získání informací o aktuálnách cenách doplňkových služeb ve virtuální měně, nazvané bezdrátové body.

Stavové kódy

- **200** – při úspěchu
- **401** – pokud uživatel není přihlášen

Návratová hodnota objekt ve formátu JSON, obsahující informace o cenách jednotlivých doplňkových služeb dostupných za bezdrátové body.

POST /user/updatemaprenew
Požadavek aktualizuje preferenci přístupu k mapě.

Stavové kódy

- **200** – při úspěchu
- **401** – pokud uživatel není přihlášen
- **403** – pokud uživatel nemá dostatek virtuálních bodů

Návratová hodnota 64-bitový integer ve formátu řetězce, obsahující časovou hodnotu doby, kdy vyprší přístup k mapě.

POST /user/updatepersonalmaprenew
Požadavek aktualizuje preferenci přístupu k osobní mapě.

Stavové kódy

- **200** – při úspěchu
- **401** – pokud uživatel není přihlášen
- **403** – pokud uživatel nemá dostatek virtuálních bodů

Návratová hodnota 64-bitový integer ve formátu řetězce, obsahující časovou hodnotu doby, kdy vyprší přístup k osobní mapě.

GET /challenge/list

Požadavek slouží k získání informací o aktuálně dostupných herních výzvách a uživatelském postupu v těchto výzvách.

Stavové kódy

- **200** – při úspěchu
- **401** – pokud uživatel není přihlášen

Návratová hodnota pole ve formátu JSON obsahující objekty s informacemi o typu, názvu, obtížnosti výzvy a uživatelském postupu v této výzvě.

GET /stats/user

Požadavek slouží k získání informací o statistikách aktuálně přihlášeného uživatele.

Stavové kódy

- **200** – při úspěchu
- **401** – pokud uživatel není přihlášen

Návratová hodnota objekt ve formátu json obsahující statistiky uživatele.

GET /feedback/new

Požadavek slouží k zaslání zpětné vazby o aplikaci. Mezi parametry jsou i informace potřebné pro přihlášení z toho důvodu, že server neukládá v cookies informace o výrobci a typu zařízení, které jsou nutnými informacemi pro každou zpětnou vazbu. Informace o tokenu slouží pro zjednodušení implementace a rychlejší ověření uživatele v případě, že není autorizován.

Parametry

- **manufacturer** – výrobce zařízení
- **model** – model zařízení
- **userToken** – uživatelský Google token
- **type** – typ zpětné vazby
- **summary** – souhrn
- **description** – popis

Stavové kódy

- **200** – při úspěchu
- **400** – požadavek nemá parametry ve správném formátu
- **401** – uživatel není přihlášen
- **403** – uživatel nemá oprávnění zasílat zpětnou vazbu

3.4 Uživatelské rozhraní Android aplikace

Uživatelské rozhraní spojuje uživatele s funkcemi aplikace a umožňuje uživateli aplikaci využívat. Vzhled uživatelského rozhraní hraje roli při rozhodování uživatele, zda aplikaci vyzkouší. Pěkné, ale špatně ovladatelné rozhraní může na druhou stranu uživatele od aplikace odradit. Ideální uživatelské rozhraní by mělo uživatele přilákat vzhledem a udržet svou dobrou použitelností a přehledností.

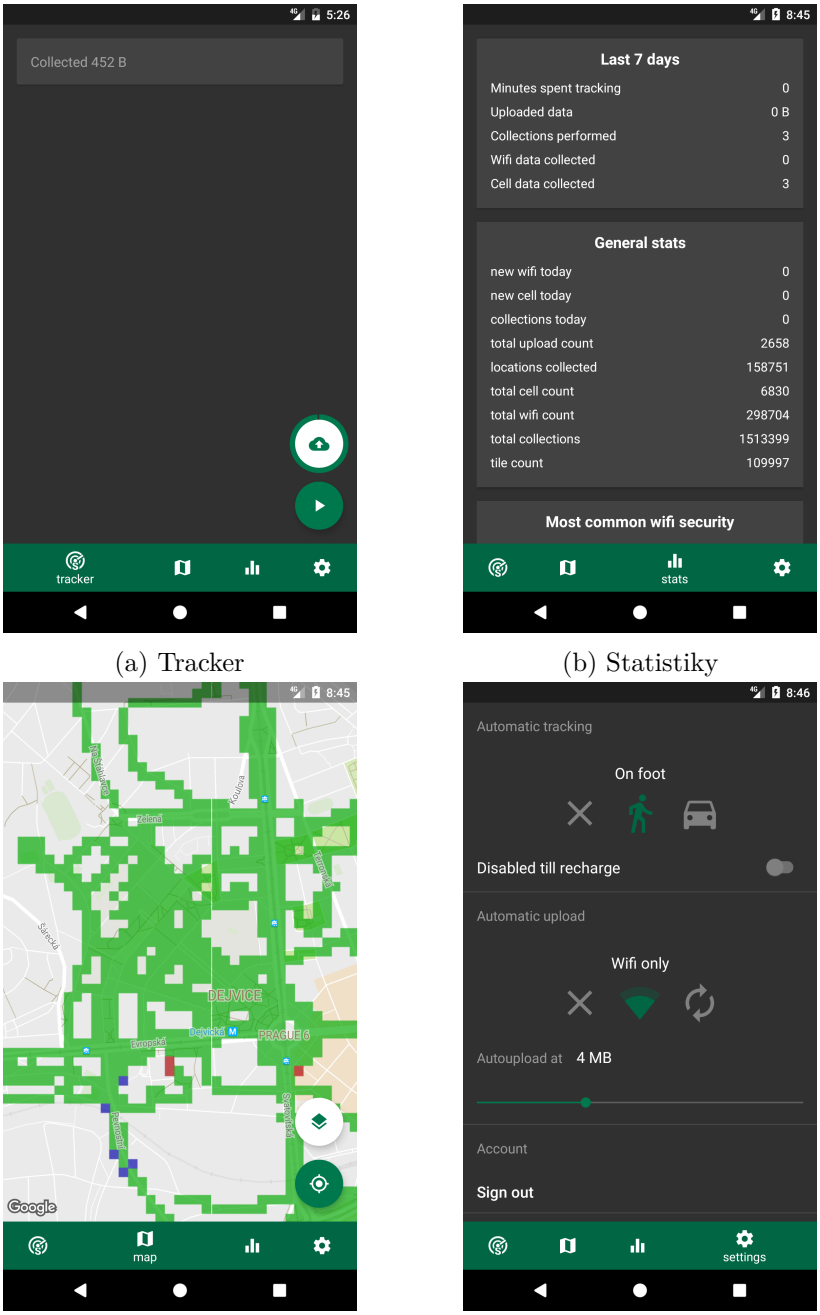
Pro mobilní aplikaci byla navržena dvě uživatelská rozhraní, standardní a experimentální. Cílem standardního uživatelského rozhraní bylo vytvořit rozhraní, ve kterém se uživatelé budou snadno orientovat. Experimentální uživatelské rozhraní bylo naopak zaměřeno na unikátní rozhraní, které využívá plynulých přechodů a podporuje pozadí s libovolnou barvou z barevného modelu RGB. Obě rozhraní využívají převážně vektorové obrázky, aby byla zachována vysoká kvalita na všech rozlišeních a zároveň malá velikost souborů.

3.4.1 Standardní

Material Design [6] od společnosti Google je designovým jazykem, který pomáhá sjednotit vzhled aplikací na platformě Android. Tento jazyk jasně definuje pravidla, kterými by se aplikace měly řídit. Právě pomocí těchto pravidel bylo vytvořeno standardní uživatelské rozhraní. Díky tomu by se uživatelé měli v rozhraní snadněji orientovat. Rozhraní se ovšem nevyhnulo výjimečným odchylkám od tohoto jazyka, kterým se ale nevyhnou ani aplikace od Google.

3.4.1.1 Popis

Hlavním ovládacím prvkem celé aplikace je spodní navigace, která umožňuje přístup do čtyř záložek: tracker, mapa, statistiky a nastavení. Uživatel díky této navigaci má každou část aplikace přístupnou pouze na jediné kliknutí. Dalším důležitým ovládacím prvkem je plovoucí tlačítko, lépe známé jako „Floating Action Button“. Vzhled využívá dvě plovoucí tlačítka pro specifické akce na jednotlivých záložkách. Záložky mohou využít obě tlačítka, jedno nebo žádné. Například na záložce mapy se uživatel může pomocí těchto tlačítek přesunout na svou aktuální polohu, nebo vyvolat seznam dostupných map. Seznam dostupných map využívá transformaci tlačítka v menu, která je sice definovaná v Material Designu, ale implementace pro Android v době tvorby aplikace neexistovala.



(a) Tracker (b) Statistiky

(c) Mapa (d) Nastavení

Obrázek 3.5: Snímky obrazovky standardního uživatelského rozhraní

3.4.2 Experimentální

Experimentální uživatelské rozhraní je založené převážně na jazykově univerzálních ovládacích prvcích. Většina nejpopulárnějších aplikací na Google Play využívá tzv. „hamburger menu“ a/nebo dolní navigaci. Denní a noční vzhled je stále spíše raritou a skutečně libovolnou barvu pozadí nenabízí kromě launcheru žádná aplikace. Hlavním důvodem je způsob, jakým Android primárně pracuje se styly. V téměř všech aplikacích jsou barvy rozhraní definovány v XML jako součást stylu, ze kterého jsou tyto barvy načteny při spuštění aktivity. Tento přístup umožňuje vývojářům jednoduchou definici barev rozhraní, ale případnou úpravu dělá téměř nemožnou. Jediným způsobem, jak je možné změnit styl aplikace, je vytvořit nový styl v XML a poté ho při spuštění aktivity zvolit v metodě onCreate. Tento způsob je velmi omezující a jedním z cílů experimentálního vzhledu bylo pokusit se tento problém vyřešit a nabídnout uživatelům skutečnou volnost v barvě pozadí.

3.4.2.1 Popis

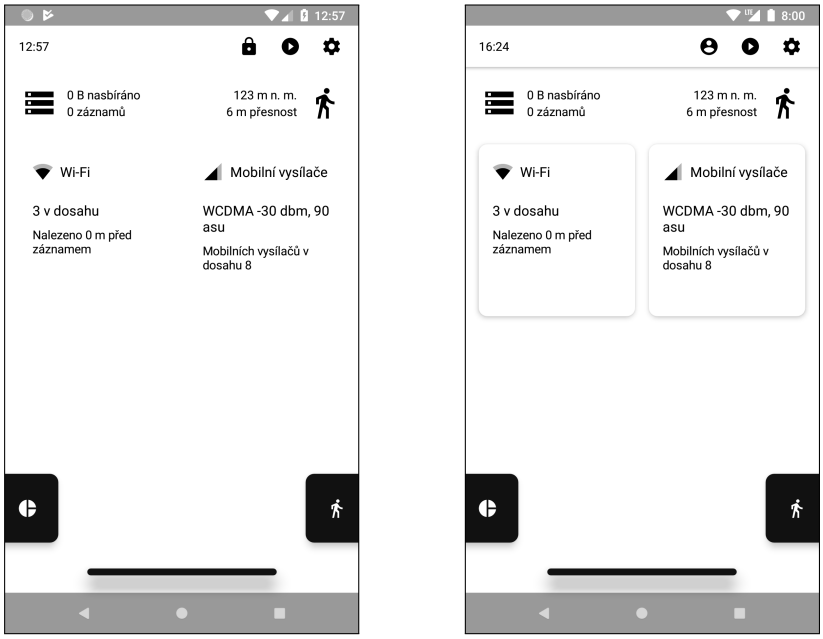
V experimentálním uživatelském rozhraní má každý prvek přiřazenu úroveň. Úroveň je využita pro výpočet barvy pozadí a je vypočtena následujícím způsobem: (úroveň rodiče) + modifikátor + (1 pokud prvek obsahuje pozadí), kde modifikátor je konstanta definovaná pro každý prvek a umožňuje pro dané prvky zvýšit nebo snížit úroveň dle potřeby. Úroveň prvku, který je výše, by nikdy neměla být menší, než prvku, který je níže. Barva pozadí je vypočtena jako (úroveň prvku) ∗ konstanta. Konstanta je v této práci určena jako 17/255 pro barvy definované v barevném modelu RGB.

Barva popředí, na rozdíl od barvy pozadí, je jednotná pro celou aplikaci a odpovídá nejvíce kontrastní barvě k barvě pozadí prvku s průměrnou hodnotou úrovně zaokrouhlené na celá čísla. Pro zvýšení výkonu je v této práci využita konstanta 1, která přibližně odpovídá průměrné úrovni prvků.

Výjimečným prvkem jsou ovládací plochy, které využívají barvu popředí na pozadí a barvu pozadí na popředí. Touto inverzí je dosaženo zvýšení kontrastu oproti barvě pozadí a plochy jsou tak snadněji odlišitelné od pozadí.

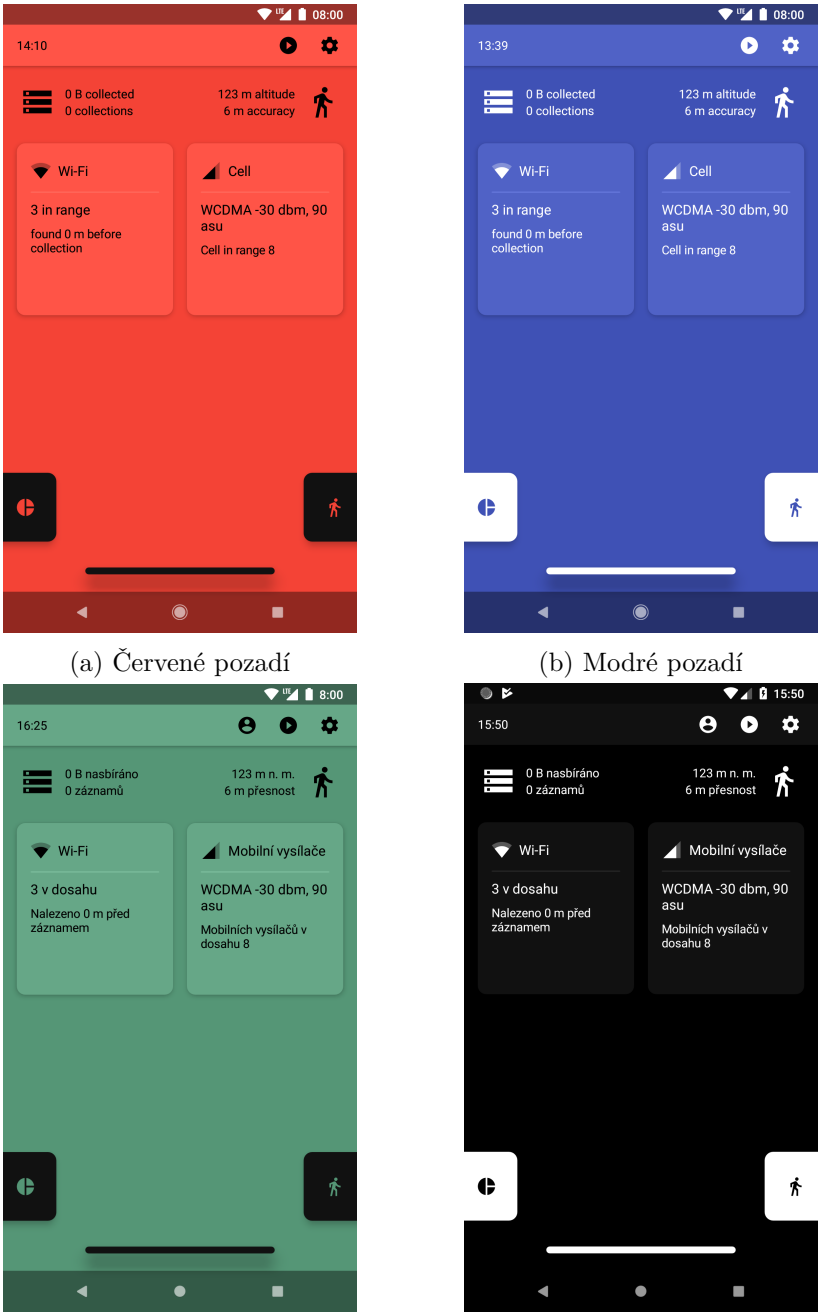
Vzhledem k tomu, že vzhled vyšší vrstvy pouze zesvětluje, dochází u velmi světlých barev ke ztrátě odlišení mezi vrstvami. Tento problém je vidět na obrázku 3.6a. Z tohoto důvodu by každá vrstva měla obsahovat stín, který tento problém napraví (obrázek 3.6b).

Příklady vzhledu pro různé barvy pozadí jsou na obrázcích 3.7a, b, c, d. Výsledný vzhled vytvořený pomocí těchto pravidel je na obrázcích 3.8a, b, c, d.



(a) Bez stínu (b) Se stínem

Obrázek 3.6: Snímky obrazovky zobrazující problematickou bílou barvu pozadí



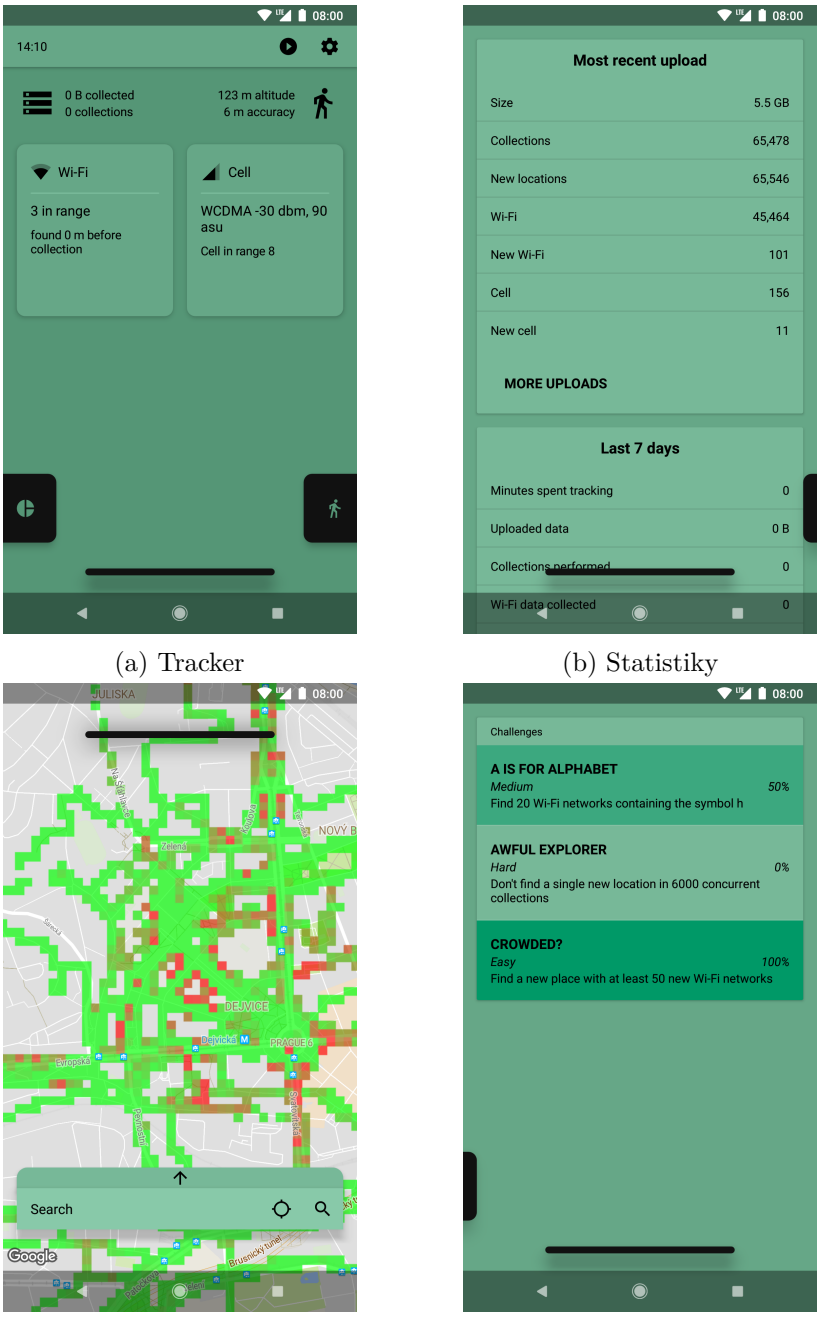
(a) Červené pozadí

(b) Modré pozadí

(c) Zelené pozadí

(d) Černé pozadí

Obrázek 3.7: Snímky obrazovky různých barev pozadí experimentálního uživatelského rozhraní



(a) Tracker (b) Statistiky

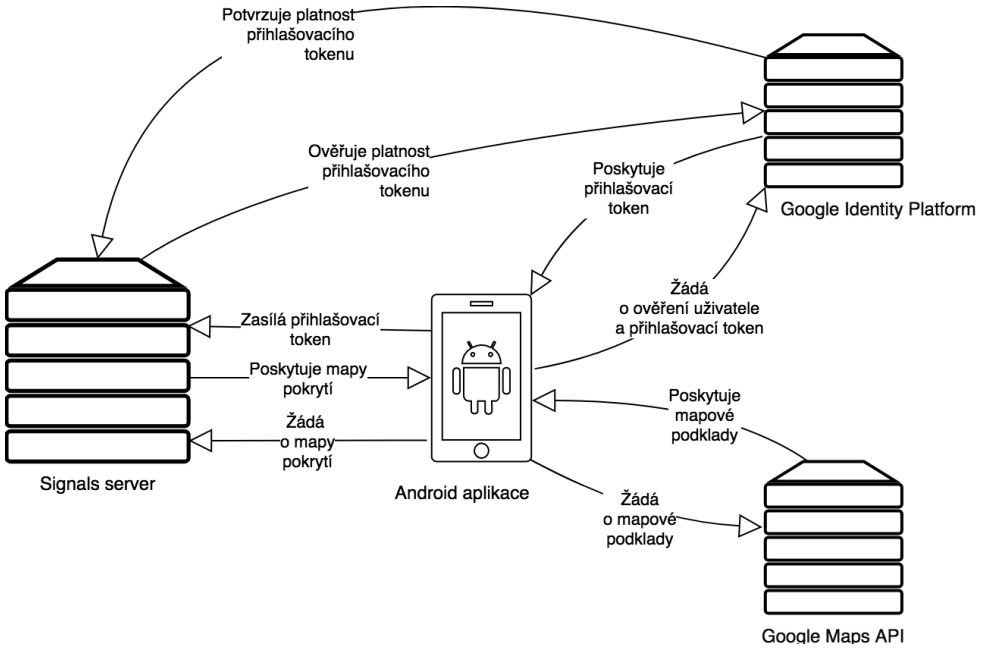
(c) Mapa (d) Aktivity

Obrázek 3.8: Snímky obrazovky experimentálního uživatelského rozhraní

Kapitola 4

Implementace

Obsahem této části je popis implementace Android aplikace a serverové části. Na obrázku 4.1 lze nalézt diagram síťové komunikace Android aplikace a serveru při přihlášení a využívání mapových podkladů. Tento diagram znázorňuje úplnou komunikaci se servery Google, která je vyžadována pro správnou funkčnost aplikace.



Obrázek 4.1: Diagram síťové komunikace serveru a Android aplikace

4.1 Android aplikace

4.1.1 Použité technologie

4.1.1.1 Vývojové prostředí

Jako vývojové prostředí pro implementaci bylo vybráno Android Studio, protože se jedná o oficiální, kvalitní a udržovaný nástroj pro vývoj aplikací na platformě Android. Je založeno na IntelliJ IDEA od společnosti JetBrains. Mezi oficiálně podporované jazyky patří Java, Kotlin a C++. Prostředí má snadnou správu doplňků a využívá Gradle jako sestavovací (build) systém. Tvorba aplikací je v Android Studiu také výhodná díky kvalitnímu emulátoru a rychlé kompilaci pomocí „Instant Run“.

4.1.1.2 Kotlin

Kotlin je staticky typovaný programovací jazyk, kompatibilní s byte kódem Javy. V roce 2017 se stal Kotlin dalším, oficiálně podporovaným jazykem, pro vývoj na platformě Android. Od Android Studia 3.0 je Kotlin také podporován v tomto vývojářském prostředí. Android P (API 28) bude první verzí Androidu, která bude obsahovat optimalizace pro byte kód generovaný aplikacemi napsanými v jazyce Kotlin. Mezi hlavní rozdíly oproti Javě patří rozlišování null a non-null typů, syntaxe, funkce vyššího řádu [1], rozšiřující funkce (extension functions). Kotlin také v rámci experimentálního balíčku coroutines zjednodušuje práci s vlákny. Coroutines jsou experimentální hlavně z důvodu neustálého vývoje tohoto API, v budoucnu by se měly stát součástí standardní knihovny Kotlin [7]. Jednou z výhod tohoto jazyka je jeho častá aktualizace.

4.1.1.3 Travis CI

Travis CI je cloudová služba na průběžnou integraci[2]. Služba je zdarma pro všechny open-source projekty umístěné na službě GitHub. Pro Android aplikaci byl Travis CI nastaven tak, aby sestavil aplikaci, spustil unit testy (ne UI testy) a vypočetl pokrytí kódu, známějším spíše pod anglickým názvem „code coverage“.

4.1.2 Automatizace

Jednou z hlavních výhod Android aplikace, oproti aplikacím analyzovaným 1, je její automatizace. Aplikace dokáže automaticky, na základě aktivity, aktivovat sběr dat. Nasbíraná data poté dokáže automaticky, dle nastavení, nahrát na server. Uživatel se tedy nemusí o nic starat.

[1] Funkce vyššího řádu (Higher Order Functions) jsou funkce, které přijímají jinou funkci jako argument, nebo vrací funkci.

[2] Průběžná integrace je také lépe známa pod zkratkou CI, nebo anglickým názvem Continuous integration

4.1.2.1 Automatický záznam dat

Automatický sběr dat probíhá na základě uživatelem nastavené aktivity. K dispozici má uživatel tři možnosti: vypnuto, pěšky, v pohybu (pěšky + v dopravním prostředku). Pro rozpoznání aktivity aplikace využívá „Activity Recognition API“, které je součástí Play Services od společnosti Google.

Problémem tohoto řešení je, že při uspání zařízení přestane aplikace dostávat informace o aktuální aktivitě. Tento problém byl jednoduše vyřešen pomocí služby, která běží na popředí a probouzí v pravidelných intervalech zařízení z režimu spánku. V průběhu testování bylo využití baterie touto službou zcela zanedbatelné.

4.1.2.2 Automatické nahrávání dat

Pro automatické nahrávání dat nabízí aplikace dva způsoby: dle množství nasbíraných dat a „chytrý“.

První možnost je velmi jednoduchá, jakmile uživatel dosáhne požadovaného množství dat, data se nahrají. Hlavním problémem tohoto řešení je, že uživatel má na svém zařízení téměř neustále nenahraná data. Jako řešení tohoto problému vznikl „chytrý“ způsob nahrávání dat.

„Chytrý“ způsob nahrávání dat nahrává data po určité době uživatelovy neaktivity. Pokud uživatel nasbíral jeden mebibyte dat a více, budou data nahrána po jedné hodině neaktivity. Minimální množství dat, které se nahraje na server pomocí „chytrého“ nahrávání, je 128 kibibytů. Pokud by uživatel nasbíral přesně 128 kibibytů, budou data nahrána po dvanácti hodinách neaktivity. Mezi 128 kibibyty a 1 mebibytem, bude požadovaná doba neaktivity pro nahrání dat spočítána poměrně k množství dat.

4.1.3 Knihovny

Při návrhu uživatelských rozhraní byla složitost implementace až druhořadým parametrem, hlavní důraz byl kladen na vzhled a použitelnost. V několika případech se kvůli tomuto rozhodnutí stalo, že neexistovala knihovna, která by nabízela požadovanou implementaci. Z toho důvodu vznikly 4 knihovny: Slider, Table, Draggable a TouchDelegate. Všechny knihovny byly vytvořeny s jistou mírou univerzálnosti, a tak je možné je využít i v jiných aplikacích. Knihovny jsou publikovány ve veřejném cloudovém repositáři JCenter a využívají CI pro snadnější a rychlejší odhalení problémů.

4.1.3.1 Slider

Knihovna Slider rozšiřuje `SeekBar`, implementovaný v Androidu. Mezi funkce knihovny patří:

- Podpora pole libovolných objektů, místo minimální a maximální hodnoty. Objekty se vždy zobrazují jako řetězce v závislosti na předané funkci.
- Automatické ukládání změn do `SharedPreferences`.
- V případě využití minima a maxima podpora pro hodnotu skoku, o kterou se má Slider posouvat.
- Podpora pro automatickou aktualizaci vybraného `TextView` hodnotou ze Slideru.

Automatické testy pokrývají všechny kritické části knihovny, které jsou testovatelné bez UI testů. V aplikaci je tato knihovna využita v nastavení, pro výběr rychlosti aktualizace aktivity.

4.1.3.2 Draggable

Knihovna Draggable přidává pohyblivá tlačítka se dvěma stavy. Je založená na třídě `AppCompatImageButton`, implementované v knihovně AppCompat od Google. Knihovna umožňuje k tlačítkům přiřadit přidružené objekty, nazvané `DraggablePayload`, které se posouvají po vlastní trase, v závislosti na aktuální poloze tlačítka. Aplikace tato tlačítka využívá v experimentálním uživatelském rozhraní pro přístup k mapě, statistikám a aktivitám.

Vzhledem k složitosti testování této komponenty automatickými testy a problémy s UI testy bylo rozhodnuto nevytvářet automatické testy a testovat knihovnu pouze manuálně.

4.1.3.3 Table

Knihovna Table přidává podporu pro jednoduchou tabulku přidanou do karty (třída `CardView`). Knihovna se skládá ze dvou tříd: `Table` a `TableAdapter`. Třída `TableAdapter` implementuje třídu `BaseAdapter`, kterou implementuje například také `ArrayAdapter`, pro `ListView`. Třída `Table` nedědí z `View`, ale stará se o tvorbu a recyklaci hierarchie tabulky. Do budoucna by bylo lepší třídu Table předělat tak, aby přímo tvořila kořen hierarchie místo toho, aby ji pouze spravovala.

4.1.3.4 TouchDelegate

Knihovna TouchDelegate umožňuje rozšířit oblast, na které jsou registrovány dotyky pro dané `View`. Android již obsahuje třídu `TouchDelegate`, ovšem tato třída má mnohá omezení. Nativní implementace `TouchDelegate` předpokládá, že dané `View` je statické, a tak neaktualizuje oblast při změně pozice daného

`View`. Dalším problémem je, že rodičovské `ViewGroup` lze přiřadit pouze jedinný `TouchDelegate` objekt. Bez modifikací tak nelze rozšířit dotekovou oblast dvou tlačítek se stejným rodičem. Oba tyto problémy knihovna řeší.

Knihovna nevyužívá automatické testy, protože UI testy jsou velmi nestabilní v Travis CI.

4.1.4 Funkce

Android aplikace nabízí uživateli, kromě základní funkcionality vyžadované pro správnou funkčnost, také funkce, které budou popsány v této části.

4.1.4.1 Export dat

Aplikace umožňuje uživatelům export dat pro osobní využití. Export dat je zajištěn pomocí sdílení dat s aplikacemi třetích stran, jako jsou například cloudová uložiště. Případné přímé ukládání by vyžadovalo oprávnění pro přístup k externí paměti, které tato aplikace prozatím nemá. V rámci testování se způsob exportu dat přes sdílení ukázal postačující, a tak nebylo prozatím přímé ukládání implementováno. Pokud se v budoucnu ukáže toto stanovisko jako chybné, může být podpora přidána.

4.1.4.2 Zpětná vazba

Aplikace má integrovanou zpětnou vazbu, kterou mohou uživatelé využít pro zaslání hlášení o chybách a nápadech pro zlepšení. Pokud se zpětná vazba ukáže jako užitečná a neduplicitní, získá uživatel bezdrátové body. Na druhou stranu, pokud uživatel bude zpětné vazby zneužívat a jeho zpětná vazba bude vyhodnocena jako nežádoucí, bude jeho budoucí zpětná vazba automaticky odmítnuta.

4.1.4.3 Vícejazyčná podpora

Aplikace podporuje 2 jazyky: angličtinu a češtinu. Podpora více jazyků je implementována pomocí standardního způsobu překladů na Android platformě, který využívá XML soubory umístěné mezi „resources“ ve složkách „values-(zkratka jazykové mutace, například cs pro češtinu)“. Výchozí jazyková mutace, kterou je v tomto případě angličtina, je umístěna ve složce „values“. Protože bylo rozhodnuto snížit množství informací, které server o uživateli ví, na minimum, server nemá informace o aktuální uživatelské mutaci v Android aplikaci, a proto jsou statistiky dostupné pouze v primárním jazyce, angličtině.

V rámci práce byl vytvořen také koncept překladu textů ze serveru, který je dostupný pro herní výzvy. Koncept je založen na tom, že server dodá identifikaci daného textu a případné parametry. Android aplikace si poté podle identifikátoru najde řetězec, načte ho a doplní do něj parametry. Výhodou

tohoto konceptu je, že uživatel bude mít vždy konzistentně přeloženy texty v aplikaci, nestane se tedy, že by měl stažené anglické statistiky na české aplikaci. Nevýhodou tohoto konceptu ovšem je, že pokud uživatel nebude mít dostupnou poslední verzi aplikace, nebo text nebude z nějakého důvodu přeložen, bude uživateli zobrazena pouze informace o tom, že text chybí. Tento problém lze vyřešit poskytnutím výchozího textu serverem. Tento koncept se osvědčil, ovšem jeho rozšíření na další části aplikace nebylo z časových důvodů v rámci této aplikace možné. Pro rozšíření je nutné vytvořit univerzální rozhraní, přes které bude možno předávat identifikátory a parametry a toto rozhraní implementovat jak na klientovi, tak na serveru.

4.1.4.4 Sběrový zámek

Pro větší uživatelskou kontrolu nad automatickým sběrem dat byl do aplikace přidán sběrový zámek. Díky tomuto zámku může uživatel zamknout automatický sběr dat na danou dobu, nebo do připojení k nabíječce. Zámek je implementován rozšiřitelně, a tak v budoucnu bude možné přidat další druhy zámků. Každý druh zámku je nezávislý na ostatních a automatický sběr dat nezačne, dokud nejsou všechny zámky odemčeny. Pokud je libovolný zámek aktivní, uživateli se na úvodní obrazovce aplikace zobrazí ikona zámku, pomocí které může jednoduše odemknout všechny zámky.

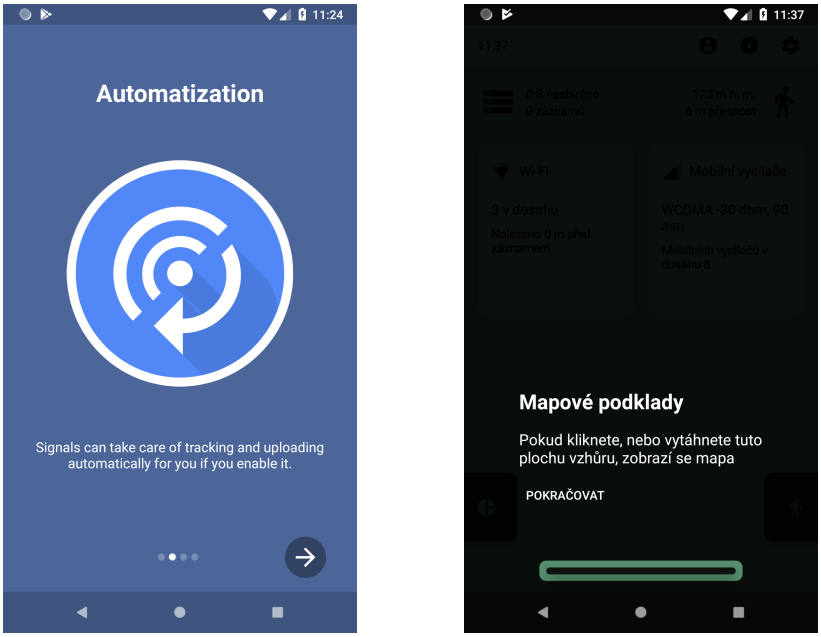
4.1.4.5 Tutoriál

Ideální uživatelské rozhraní je takové, které nevyžaduje žádný tutoriál. Aplikace, které jsou atypické svými funkcemi, nebo vzhledem, by ovšem tutoriál obsahovat měly, aby se uživatelé lépe v aplikaci orientovali.

První implementace tutoriálu byla v rámci standardního vzhledu, která seznámila uživatele s aplikací a donutila ho nastavit si základní preference. Tento přístup se příliš neosvědčil a v rámci testování byli uživatelé spíše zahlceni možnostmi, než seznámeni s aplikací. Proto v rámci experimentálního uživatelského rozhraní byl celý koncept tutoriálu přepracován.

Tutoriál v experimentálním uživatelském rozhraní se zaměřuje primárně na seznámení uživatele s uživatelským rozhraním a až druhotně s funkcemi aplikace. Pro aplikaci byly vytvořen tutoriál pro úvodní obrazovku a pro mapu. Do budoucna je také plánován tutoriál pro nastavení, aby uživatele seznámil s některými více skrytými možnostmi.

V rámci testování bylo zjištěno, že novější přístup k tutoriálu v experimentálním vzhledu se uživatelům více zamlouval. Ovšem ani tento přístup nebyl zcela ideální a do budoucna bude vyžadovat vylepšení, aby uživatele lépe seznámil s aplikací a pomohl jim se v ní zorientovat.



(a) Tutoriál v standardním uživatelském rozhraní

(b) Tutoriál v experimentálním uživatelském rozhraní

Obrázek 4.2: Snímky obrazovky tutoriálů, vytvořených v rámci aplikace

4.1.5 Publikace na Google Play

Každý, kdo chce publikovat aplikaci na Google Play, musí zaplatit jednorázový poplatek $25 [8]. Aplikace jsou publikovány přes řídící panel Google Play.

Před prvotním nahráním aplikace je dobré si rozmyslet název balíčku, protože pozdější změna není pro danou aplikaci možná. Jediným způsobem, jak změnit název balíčku, je vytvoření nové aplikace, které s sebou nese riziko ztráty uživatelů a další případné problémy. Název balíčku musí být unikátní na celém Google Play [9].

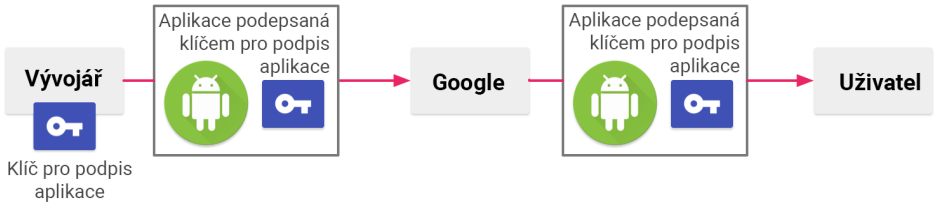
Je také nutné vybrat si způsob podpisu aplikace. Google nabízí dvě alternativy podpisu: podepsání aplikace lokálně, nebo podepsání aplikace na Google Play.

Při zvolení lokálního podpisu aplikace ji vývojář podepíše při sestavení a nahraje na Google Play, kde proběhne ověření klíče a v případě, že je vše v pořádku, je aplikace publikována.

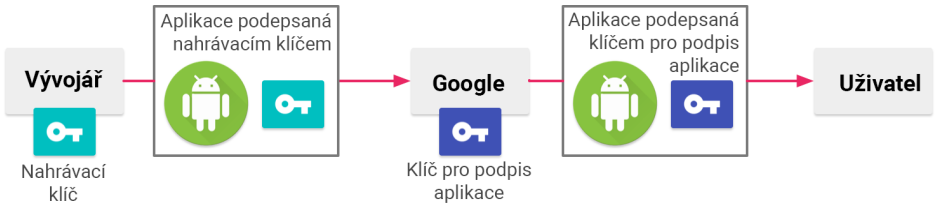
Při zvolení podpisu na Google Play je nutné nejdříve podepsat aplikaci lokálně s využitím nahrávacího klíče, známého také jako „upload key“, poté nahrát na Google Play, kde proběhne znovu podepsání aplikace s klíčem pro podpis aplikace, známého také jako „app signing key“. Výhodou podpisu na Google Play je, že podpisový klíč je bezpečně uložen na serverech Google a v případě, že by nahrávací klíč byl ukraden, nebo ztracen, lze kontaktovat

Google a tento klíč nahradit jiným [10].

Jakmile je aplikace nahraná na Google Play, je nutné vyplnit popis aplikace, snímky obrazovky a další povinné údaje. Jakmile je vše připraveno, lze aplikaci publikovat na Google Play.



Obrázek 4.3: Obrázek znázorňující podpis aplikace lokálním klíčem [10].



Obrázek 4.4: Obrázek znázorňující podpis aplikace klíčem na Google Play [10].

4.2 Server

V této kapitole si představíme hlavní použité technologie na serveru. Prozkoumáme generování mapových podkladů a podíváme se na zabezpečení dat a serveru.

4.2.1 Použité technologie

V této části se seznámíme s technologiemi využitými na serverové části aplikace.

4.2.1.1 ASP.NET Core

ASP.NET Core je volně dostupný webový framework s otevřeným kódem (open-source software). Jeho hlavním vývojářem je Microsoft Corporation a jedná se o nástupce ASP.NET. Je vytvořen na novém multiplatformním frameworku .NET Core. Díky otevřenosti celého frameworku byla komunitou ve verzi 2.0 přidána podpora 32-bitových ARM procesorů na platformě Linux [11]. Od verze 2.1, která podle plánu má být vydána ve druhém kvartálu roku 2018, bude ARM32 podporován na platformě Linux oficiálně. [12].

4.2.1.2 RavenDB

První volbou databázového programu bylo MongoDB. Bohužel MongoDB oficiálně podporuje pouze 64 bitovou verzi procesorové architektury ARM a Raspberry Pi i přes svůj 64 bitový procesor má oficiální operační systém pouze v 32 bitové verzi. Pro Raspberry Pi bylo tedy nutné vybrat jiný databázový program. RavenDB má od verze 4.0 oficiální podporu Raspberry Pi. RavenDB je transakční NoSQL[3] databázový program. Transakce z definice musí splňovat ACID (atomičnost, konzistence, izolovanost, trvanlivost) [14]. Důležitou vlastností je také oficiální podpora ASP.NET Core 2.0.

4.2.1.3 Material Components

Material Components [15] je knihovna s otevřeným kódem, udržovaná společností Google. Mezi její hlavní cíle patří mimo jiné modularita, rychlost a dodržení jazyka Material Design. Knihovna poskytuje komponenty navržené dle Material Designu, které mohou vývojáři využít. Mezi tyto komponenty patří, kromě základních komponent HTML, také snackbar, toolbar, taby nebo karty. Díky této knihovně bylo výrazně sníženo úsilí, nutné pro implementaci webového rozhraní.

4.2.1.4 Nginx

Nginx je webový server, který je v této práci využíván jako reverzní proxy[4]. První verzi vytvořil Igor Sysoev a byla zveřejněna v roce 2004. V roce 2011 byla založena stejnojmenná firma, aby poskytovala komerční podporu a zajistila budoucí vývoj nginxu.

4.2.1.5 Hangfire

Hangfire je framework s otevřeným zdrojovým kódem spravující úkoly, které mají být vykonány na pozadí. Framework se stará o zařazování úkolů do fronty, jejich přiřazení takzvaným „worker threadům“ a persistenci úkolů. V případě vypnutí serveru nedojde ke ztrátě fronty a zpracování může pokračovat po zapnutí. Tento framework také podporuje více aktivních fyzických „worker“ serverů najednou a je tak možné zpracovávat úkoly jedné aplikace na více zařízeních. Server tento framework využívá pro zpracování nahraných dat, týdenní aktualizaci herních výzev, generování statistik, generování map a další.

[3] NoSQL databáze jsou databáze, které se snaží vylepšit některé části SQL databází, jako například: škálovatelnost, rigidní schéma, rychlost a další [13].

[4] Reverse proxy server je typ proxy serveru, který slouží jako prostředník mezi uživatelem a serverovou aplikací [16].

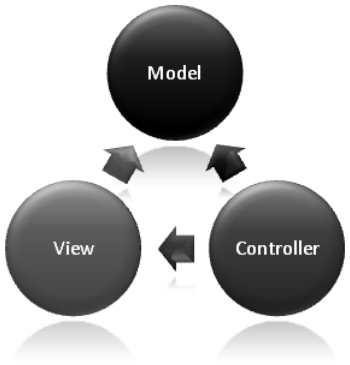
4.2.2 Architektura

Server je založen na architektuře MVC (Model–view–controller). Architektura MVC je tvořena 3 hlavními částmi: model, view a controller.

Model reprezentuje strukturu dat a práci s daty v rámci aplikace. Často spolupracuje s databází pro ukládání a načítání dat.

View reprezentuje uživatelské rozhraní. Typicky je uživatelské rozhraní tvořeno pomocí dat z modelu.

Controller reprezentuje uživatelskou interakci s modelem. [17]

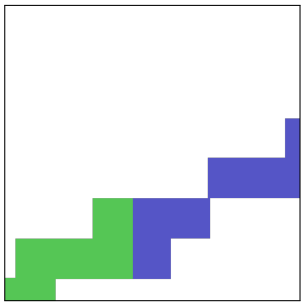


Obrázek 4.5: Komunikaci mezi částmi MVC v ASP.NET Core [17].

Uživatelé jsou po zaslání požadavku přesměrováni do controlleru, který je zodpovědný za práci s modelem. Controller provede pomocí modelu požadovanou uživatelskou akci a/nebo získá výsledky dotazů. Nakonec controller zvolí vhodné view pro zobrazení všech požadovaných dat z modelu.

4.2.3 Generování mapových podkladů

Generování mapových podkladů byla technicky jedna z nejsložitějších částí serverové aplikace. Jako poskytovatel mapových podkladů byly vybrány Google Mapy, protože nabízejí celosvětové mapy a API, které umožňuje přidat překryvné vrstvy. Tyto překryvné vrstvy jsou složeny z jednotlivých dlaždic, v angličtině „tile“, a jsou generovány zvlášť pro každou úroveň přiblížení. Počet všech možných dlaždic pro danou úroveň přiblížení je: $2^{2*\text{úroveň přiblížení}}$. Jednoduše řečeno, při přiblížení o jednu úroveň se každá dlaždice z původní úrovně rozdělí na čtyři nové dlaždice. Každá dlaždice v každém přiblížení musí být stejně veliká a v případě této práce to byl obrázek s výškou i šířkou 256 obrazových bodů.



Obrázek 4.6: Příklad jedné dlaždice překryvné vrstvy mapových podkladů

Pro správné vygenerování jednotlivých částí je nutné převést polohu každého záznamu ze zeměpisné šířky a zeměpisné délky, získané z GPS, na x a y souřadnice části při daném přiblížení. Při převodu je nutné zachovat desetinnou část souřadnice x a y, aby bylo možné dopočítat i přesné obrozové body, na kterých se má daná poloha vykreslit. Vzhledem k chybě v ASP.NET Core 2.0, která na platformě ARM způsobuje občasné fatální selhání dotazu do databáze, je generování map vypnuto. Případná dočasná řešení by zásadně snižovala výkon, a proto bylo rozhodnuto funkci vypnout a poskytovat několik měsíců staré mapy pokrytí.

4.2.4 Zabezpeční a uživatelská data

V této části se podíváme na zabezpečení jednotlivých částí aplikace, ať už proti útočníkům, kteří se snaží získat data uživatele, tak proti uživatelům, kteří se snaží nahrát falešná data.

4.2.4.1 Hypertext Transfer Protocol Secure

První částí, kde je nutné ochránit uživatelská data, je přenos dat mezi Android aplikací a serverem. Toho lze docílit pomocí HTTPS. HTTPS bohužel není automaticky bezpečné a je nutné ho správně nastavit.

Pro navázání komunikace se využívá DHE (Diffie–Hellman key exchange) a ECDHE (Elliptic Curve Diffie-Hellman Ephemeral). Dle doporučení na potlačení útoků na Diffie-Hellmana bylo využito 2048 bitové bezpečné prvočíslo[5] [18]. Pro vyšší rychlost byla zvolena verze s eliptickými křivkami. V průběhu testování bylo ovšem zjištěno, že Android 7.0 nepodporuje eliptické křivky a bylo nutné přidat podporu pro pomalejší verzi algoritmu bez eliptických křivek.

Pro zabezpečení komunikace server využívá TLS verze 1.2. Z bezpečnostních důvodů byla vynechána podpora TLS verze 1.0 [19]. Pro přiblížení se modernímu bezpečnostnímu profilu Mozilly, který by měl být kompatibilní

[5]Bezpečné prvočíslo je prvočíslo ve tvaru 2p + 1, kde p je také prvočíslo.

s Androidem 5.0 a novějším, byla vynechána také podpora TLS verze 1.1 [20]. Podpora pro TLS verze 1.3 bude přidána v budoucnu, jakmile bude TLS 1.3 podporováno stabilními verzemi Nginx a OpenSSL.

Pro využití HTTPS je nutné vlastnit certifikát. Certifikát je digitálně podepsaný veřejný šifrovací klíč, vydaný šifrovací autoritou. Veřejné servery, komunikující s uživateli přes internet, musejí mít certifikát podepsaný certifikační autoritou, kterou všechny hlavní prohlížeče považují za důvěryhodnou, jinak uživatel bude přivítán varovným upozorněním, že webová stránka není bezpečná. Mezi důvěryhodné certifikační autority naštěstí patří i Let's Encrypt, který nabízí certifikáty zdarma. Let's Encrypt vydal svůj první certifikát v roce 2015 a díky společnosti IdenTrust je důvěryhodnou certifikační autoritou [21].

Jelikož aplikace využívá pro identifikaci uživatele cookies, je nutné je alespoň základně zabezpečit. Toho je docíleno pomocí několika nastavení. Nejdříve byla cookies označena jako „Secure“, což informuje prohlížeč o tom, že by cookies měl přenášet pouze přes zabezpečené spojení. Poté následovalo označení cookies jako „HttpOnly“, které informuje prohlížeč o tom, že by cookies neměla být přístupná z JavaScriptu a zabránit tak XSS (Cross-site scripting) útokům [22].

V neposlední řadě byla cookies označena jako „sameSite“ [23]. Tento atribut informuje prohlížeč o tom, že by neměl cookies odesílat na jinou adresu, než je ta, které cookies patří. Tím lze zabránit některým druhům útoků. Nevýhodou všech tří řešení je, že se spoléhají na klienta.

Nastavením cookies a HTTPS byla zabezpečena komunikace mezi serverem a klientem.

4.2.4.2 Zabezpečení webového rozhraní

Pro webové aplikace je vhodné nastavit také hlavičky požadavků, které prohlížeči sdělí další informace o tom, jak se má při komunikaci se serverem chovat a zabránit tak některým útokům na webovou aplikaci a uživatele.

CSP (Content Security Policy) je nejlepším způsobem, jak předcházet „XSS“ a „clickjacking“ útokům. CSP umožňuje nastavit zdroje, ze kterých by měl prohlížeč načítat data. Dle doporučení by CSP mělo být nastaveno tak, aby zabraňovalo načítání inline JavaScriptu, které jsou jedním z hlavních způsobů XSS útoků[24].

V ASP.NET Core může takovéto nastavení ztížit předávání dat z C# do JavaScriptu a v některých případech znepřehlednit aplikaci. CSP má pro tyto případy řešení, nazvané „nonce“. Nonce umožňuje povolit vybrané skripty přidáním náhodně vygenerovaného hashe, který je poté přidán ke každému inline skriptu a do CSP hlavičky. Tento hash musí být jiný pro každý požadavek.

Další důležitou hlavičkou je HSTS (HTTP Strict Transport Security), která prohlížeč informuje o tom, že by měl na danou webovou stránku přistupovat pouze přes HTTPS [25]. Existuje také seznam předběžného načtení

(HSTS Preload List), který obsahuje seznam domén, na kterých by klienti měli využívat pouze HTTPS. Pro snížení velikosti tohoto seznamu nejsou podporovány subdomény, a proto není server z této práce na tomto seznamu umístěn.

4.2.4.3 Zabezpečení proti falešným záznamům ze strany uživatele

Android aplikace obsahuje několik různých zabezpečení proti falešným záznamům.

První ochranou je omezená dostupnost na Google Play. Google Play umožňuje omezit dostupnost aplikace pro zařízení, která neprojdou základními integritními testy, nebo nejsou certifikována společností Google [26]. Certifikovaná zařízení musí projít testy, které kontrolují základní funkce systému Android. Například se tedy na certifikovaném zařízení lze spolehnout na to, že třída HashMap se bude chovat dle Android dokumentace [27].

S certifikací souvisí další část zabezpečení, která se spoléhá na funkci systému Android, která identifikuje, zda poloha pochází skutečně z GNSS[6] a nebyla pouze vygenerována pomocí falešného poskytovatele polohy, určeného k testování aplikací.

Další části ochrany je hash, který je vygenerován v závislosti na datech. Hashovací funkce, která tento hash generuje, byla specificky navržena pro tuto aplikaci. Tento hash je nahráván spolu s daty na server, kde je vygenerován hash pomocí stejné funkce a je ověřeno, že se tyto hashe shodují.

4.2.4.4 Zabezpečení proti falešným záznamům ze strany Wi-Fi sítí

Falešné záznamy mohou být také vytvořeny bez vědomí uživatele. Wi-Fi sítě mají z výroby přiřazeny unikátní BSSID, ale existují Wi-Fi sítě, které mají BSSID upraveno na neunikátní hodnotu. Tyto Wi-Fi sítě mohou v databázi vytvořit kolize, proti kterým se nelze snadno bránit. Server využívá dva způsoby identifikace těchto Wi-Fi sítí.

Prvním a silnějším je označení duplicitního BSSID. Wi-Fi sítě lze takto označit pouze v případě, že při jednom sběru dat nalezneme jedno BSSID vícekrát. To se může dít například ve špatně nastavených sítích, které pro automatické přihlášení uživatelů na různých přístupových bodech využívají právě duplicitní BSSID.

Druhým identifikátorem je označení Wi-Fi sítě jako pohybující se. Wi-Fi síť lze takto označit právě když je nalezena na 2 místech, která jsou od sebe vzdálena natolik, že nelze pochybovat o tom, že by Wi-Fi síť měla takový dosah.

Bohužel i přes tyto dvě detekce mohou některé Wi-Fi sítě s duplicitním BSSID projít neodhaleny.

[6] Mezi globální navigační systémy (global navigation satellite system) patří Galileo (Evropa), GPS (USA), GLONASS (Rusko), BeiDou (Čína) [28].

4.2.4.5 Zabezpečení přihlášení

Pro získání personalizovaných dat se musí každý uživatel přihlásit. Přihlášení proto musí být bezpečné a co nejlépe ověřit, že se skutečně jedná o uživatele, který má mít k těmto datům přístup.

Pro zjednodušení tohoto problému využívá aplikace Google Sign-In. Výhodou Google Sign-In oproti konkurenci je, že každý uživatel této aplikace musí mít Google účet, protože jinak by si aplikaci nestáhl.

Google Sign-In zajišťuje ověření přihlášení uživatele na serveru pomocí tokenu. Tento token uživatel získá po přihlášení v mobilní aplikaci a je odeslán na server. Server token odešle zpět na servery Google pro ověření, že se skutečně jedná o validní přihlašovací token. Jakmile server obdrží odpověď od serverů Google, zkontroluje, zda bylo přihlášení úspěšné a zda byl token opravdu určen pro tuto aplikaci. V případě, že token nebyl určen pro tuto aplikaci, je přihlášení zamítnuto. V opačném případě je identifikační řetězec a token uložen na serveru do cookies spolu s oprávněními uživatele.

Získání přístupu by pro útočníka mělo být obtížné, protože klient zasílá svůj token přes zabezpečené spojení a v případě, že by útočník token získal z jiné služby, server takový token zamítne při ověření cílové aplikace pro daný token.

4.2.4.6 Ochrana osobních údajů

Ochrana osobních údajů je důležitá a snahou autora v této aplikaci bylo snížit množství informací o uživateli na minimum. Při zpracování proto dochází k anonymizaci dat. Při ukládání Wi-Fi, mobilních sítí, nebo polohy jsou tyto informace ukládány bez informace o čase, nebo autorovi. S autorem jsou spojeny pouze agregované informace ve formě osobních statistik.

Součástí ochrany osobních údajů je také možnost uživatele vyřadit svůj přístupový bod Wi-Fi z databáze. Uživatel tak může učinit přidáním řetězce „_nomap“ na konec názvu (SSID) přístupového bodu Wi-Fi. Tuto koncovku zavedla společnost Google v rámci zlepšování ochrany soukromí [29].

4.2.5 Webové rozhraní

Webové rozhraní bylo nutné vytvořit, ale vzhledem k náročnosti mobilní aplikace nebylo prioritním. Z toho důvodu byl vybrán framework Material Design Components, který velmi urychlil tvorbu webového rozhraní předpřipravenými komponentami a styly.

Webové rozhraní není funkčnostmi zcela identické a nenabízí uživatelům například přístup k herním výzvám a nastavením. Na druhou stranu je webové rozhraní jediným rozhraním pro správu serverové aplikace a umožňuje administrátorům spravovat zpětnou vazbu, joby z frameworku Hangfire a přístup k manuálnímu spuštění některých funkcí. Důležitou částí webového rozhraní jsou také zásady ochrany osobních údajů (privacy policy).

Kapitola 5

Testování

V průběhu celého vývoje probíhalo interní testování několika vybranými uživateli, díky kterému se podařilo odhalit mnoho chyb. Testování aplikace probíhalo v průběhu celého jejího vývoje. Díky tomu byly identifikovány některé problémové části aplikace a jejich návrh mohl být vylepšen.

5.1 Testování vývojářem

Aktivní využívání aplikace vývojářem, v průběhu implementace, napomohlo k rychlejší identifikaci chyb. Pro nejdůležitější části aplikace byly vytvořeny automatizované jednotkové testy (unit testy). Při nahrání změn na GitHub dojde k automatickému sestavení aplikace a spuštění testů pomocí Travis CI. Vzhledem k problémům s UI testy na Travis CI jsou v tomto rozhraní vypnuty.

5.2 Testování uživateli

5.2.1 Alfa a beta testování

Po dokončení základních funkcí Android aplikace i serverové části bylo zahájeno alfa testování, které mělo za cíl odhalit pomocí malé skupiny uživatelů případné chyby. Hlavní výhodou více uživatelů na různých zařízeních bylo odhalení slabin aplikace. Tímto způsobem byl odhalen například problém, který vznikl tak, že se aplikace nesprávně spoléhala na správné počáteční nastavení síťového připojení k serveru na zařízení a polovina zařízení se kvůli svému odlišnému nastavení nemohla připojit.

5.2.2 Testování použitelnosti

Testování použitelnosti bylo provedeno pro obě uživatelská rozhraní, aby došlo alespoň k základnímu porovnání experimentálního uživatelského rozhraní

se standardním. Testování se zúčastnilo šest uživatelů s průměrnými až odbornými znalostmi operačního systému Android. Polovina z těchto uživatelů se také účastnila alfa a/nebo beta testování. Uživatelé měli za úkol splnit tři základní úkoly: nastavení neměnné tmavé barvy pozadí, nastavení a manuální spuštění záznamu dat a práce s mapou.

5.2.2.1 Změna barvy pozadí

Prvním úkolem, který uživatelé plnili, bylo nastavení barvy pozadí na tmavou neměnnou barvu. Tento úkol splnili všichni uživatelé bez větších problémů. Jediným problémem pro některé uživatele bylo nastavení neměnné barvy u experimentálního designu. V reakci na to byl přidán popis pro tlačítko s nastavením režimu pozadí.

5.2.2.2 Nastavení a manuální spuštění záznamu dat

Uživatelé měli za úkol omezit sběr dat pouze na Wi-Fi sítě, a tedy vypnout sběr dat o poloze a mobilních sítích a manuálně zapnout sběr dat. Uživatelé bez větších obtíží nalezli v nastavení vypnutí sběru dat o poloze a mobilních sítích. Větší problém uživatelům dělalo manuální zapnutí sběru dat. U experimentálního vzhledu uživatelé často zaměňovali neklikatelné ikonky za tlačítka. U standardního vzhledu uživatelé neměli téměř žádné problémy. V experimentálním vzhledu byla pro lepší uživatelskou orientaci upravena viditelnost některých ikon, vylepšen úvod do aplikace a nahrazena ikonka pro neznámou aktivitu, aby nedocházelo k její záměně s ikonkou pro pomoc.

5.2.2.3 Práce s mapou

Práce s mapou byla pro uživatele činností s největším množstvím dílčích úkolů. Uživatelé měli za úkol zobrazit mapu pokrytí Vodafone CZ u hradu Loket a vrátit se zpět na aktuální polohu. Pro nové uživatele se ukázal tento úkol příliš složitý a bylo nutné pro tyto uživatele doplnit zadání o informaci, že mapy vyžadují přihlášení a u přihlášení také naleznou povolení mapy. Tento problém mohl být zmírněn v případě, že by uživatel měl možnost se s aplikací předem seznámit. Řešení tohoto problému je složitější, a tak tento problém nebyl v rámci práce z časových důvodů vyřešen.

5.2.2.4 Souhrn

Standardní rozhraní bylo dle očekávání pro uživatele snazší na pochopení. Ocenili převážně jasně definované prvky, nízký počet kliknutí pro přístup k požadované funkci, čitelnost textu a snadnou orientaci v uživatelském rozhraní. Vzhled byl hodnocen spíše jako funkční, než hezky vypadající.

Experimentální rozhraní vyniklo převážně svým unikátní vzhledem a ovládáním. Dle očekávání měli uživatelé zpočátku problém s orientací v netradič-

ním uživatelském rozhraní. Uživatelé také ocenili plynulost uživatelského rozhraní, čitelnost textu, vzhled a ovládání. Hlavní problém bylo ovládací tlačítko pro mapu které, i přes jeho popsání v úvodu, bylo stále pro některé uživatele nejasné. Z toho lze odvodit, že úvodní informace jsou nedostatečným prostředkem komunikace rozhraní s unikátními ovládacími prvky.

5.2.2.5 Závěr

Testování odhalilo, že většina uživatelů poté, co se seznámila s experimentálním rozhraním, ho označila jako preferované. Experimentální rozhraní se osvědčilo jako zlepšení oproti standardnímu uživatelskému rozhraní a po ukončení testování bylo zachováno jako jediné.

5.3 Hlášení chyb

Jako základní služba pro hlášení chyb slouží Google Play. Hlášení chyb na Google Play má ovšem mnoho omezení. Automatické hlášení chyb v rámci Google Play bylo přidáno koncem roku 2017 a většina uživatelů ho má v základu vypnuté. Je tedy stále nutné spoléhat se u většiny uživatelů na to, že daný problém nahlásí. Další nevýhodou je nemožnost nahlásit vlastní problém bez pádu aplikace. Proto se vyplatí hledat řešení třetích stran, které tyto problémy nemají. Nakonec bylo zvoleno Firebase Crash reporting, které bylo v březnu 2018 nahrazeno Crashlytics. Mezi hlavní důvody volby této platformy patří fakt, že Firebase provozuje službu pro optimalizované zasílání cloudových zpráv Android zařízením. Server tyto cloudové zprávy využívá pro zasílání souhrnů z nahraných dat a upozornění o splnění herní výzvy.

Závěr

Cílem této práce bylo vytvořit aplikaci pro operační systém Android a server pro záznam a zpracování dat o Wi-Fi a mobilních sítích.

Nejprve byla provedena analýza podobných řešení, která ukázala, že žádné podobné řešení nevyužívá automatického sběru dat podle aktuální aktivity. Případy užití pomohly vytvořit lepší představu o tom, co by aplikace měla nabízet a jak se odliší od konkurence.

Dále byl proveden návrh, ve kterém byla vybrána platforma pro server, navrženo API pro komunikaci Android aplikace se serverem a navržena dvě uživatelská rozhraní pro Android aplikaci. Po dokončení návrhu byla provedena implementace Android aplikace a serverové aplikace. Android aplikace byla implementována v programovacím jazyce Kotlin a v rámci implementace vznikly čtyři Android knihovny. Serverová aplikace využívá webový framework ASP.NET Core spolu se systémem řízení báze dat RavenDB. Pro plánování zpracování jednotlivých jobů byl využit framework Hangfire. Nginx je na serveru využit jako reverse proxy.

V průběhu implementace byla aplikace aktivně využívána, jak uživateli, tak vývojářem, což vedlo k rychlejšímu odhalení problémů. Po dokončení implementace základních funkcí pro server i Android aplikaci byl server spuštěn na Raspberry Pi 3 a Android aplikace nahrána jako alfa verze na Google Play.

Po dokončení implementace proběhlo testování použitelnosti s šesti uživateli, které odhalilo několik problémů, které byly opraveny.

Výsledkem práce je Android aplikace dostupná na Google Play, která se připojí pomocí neveřejného API k serverové části běžící na Raspberry Pi 3.

Autor práce předpokládá rozvoj této aplikace i v budoucnu. Mezi hlavní plánovaná vylepšení Android aplikace patří zefektivnění sběru dat a přidání podpory pro ukládání polohy do formátu GPX, což uživatelům umožní export polohy ve formátu kompatibilním s jinými aplikacemi. U serverové části patří mezi hlavní plánovaná vylepšení rozšíření statistik, zvýšení rychlosti zpracování a přidání odhadu polohy Wi-Fi sítí.

Literatura

[1] Google: Android Developers: Supporting Different Platform Versions. [online], navštíveno: 18.2.2018. Dostupné z: https://developer.android.com/training/basics/supporting-devices/platforms.html

[2] ShiftPlusOne: Raspbian Jessie (64bit) for RPi3? [online], Březen 2016, cit. 5.4.2018. Dostupné z: https://www.raspberrypi.org/forums/viewtopic.php?p=918440#p918440

[3] StackOverflow: Most Popular Technologies. [online], cit. 21.4.2018. Dostupné z: https://insights.stackoverflow.com/survey/2018/#technology

[4] RavenDB: RavenDB Documentation - Getting Started. [online], cit. 30.4.2018. Dostupné z: https://ravendb.net/docs/article-page/4.0/Csharp/start/getting-started

[5] Eini, O.: Getting Started with RavenDB. [online], cit. 30.4.2018. Dostupné z: http://www.codemag.com/article/1203041/Getting-Started-with-RavenDB

[6] Google: Material Design Guidelines. [online], cit. 21.3.2018. Dostupné z: https://material.io/guidelines/

[7] JetBrains: Kotlin coroutines. [online], cit. 21.3.2018. Dostupné z: https://kotlinlang.org/docs/reference/coroutines.html#experimental-status-of-coroutines

[8] Google: Registrační poplatek vývojáře na Google Play. [online], cit. 27.3.2018. Dostupné z: https://support.google.com/googleplay/android-developer/answer/6112435

[9] Google: Configuring Your Application for Release. [online], cit. 29.4.2018. Dostupné z: https://developer.android.com/studio/publish/preparing#publishing-configure

[10] Google: Sign Your App. [online], cit. 16.4.2018. Dostupné z: https://developer.android.com/studio/publish/app-signing.html

[11] Marcu, P.; Choi, H.-K.; Pop, C.; aj.: Announcing .NET Core 2.1 Preview 2. [online], cit. 30.4.2018. Dostupné z: https://github.com/dotnet/core/blob/master/samples/RaspberryPiInstructions.md

[12] Lander, R.: Announcing .NET Core 2.1 Preview 2. [online], cit. 30.4.2018. Dostupné z: https://blogs.msdn.microsoft.com/dotnet/2018/04/11/announcing-net-core-2-1-preview-2/

[13] Yegulalp, S.: What is NoSQL? NoSQL databases explained. [online], Prosinec 2017, cit. 17.4.2018. Dostupné z: https://www.infoworld.com/article/3240644/nosql/what-is-nosql-nosql-databases-explained.html

[14] Microsoft: What is a Transaction? [online], cit. 31.3.2018. Dostupné z: https://msdn.microsoft.com/en-us/library/aa366402(VS.85).aspx

[15] Google: Material Components Web. [online], cit. 23.3.2018. Dostupné z: https://github.com/material-components/material-components-web/

[16] NGINX: What Is a Reverse Proxy Server? [online], cit. 6.5.2018. Dostupné z: https://www.nginx.com/resources/glossary/reverse-proxy-server/

[17] Smith, S.: Overview of ASP.NET Core MVC. [online], Leden 2018, cit. 22.4.2018. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.1

[18] Adrian, D.; Bhargavan, K.; Durumeric, Z.; aj.: Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *22nd ACM Conference on Computer and Communications Security*, Říjen 2015.

[19] Comodo: Deprecation of TLS 1.0. [online], cit. 24.3.2018. Dostupné z: https://www.comodo.com/e-commerce/ssl-certificates/tls-1-deprecation.php

[20] Mozilla: Server side TLS: Modern compatibility. [online], cit. 17.4.2018. Dostupné z: https://wiki.mozilla.org/Security/Server_Side_TLS#Modern_compatibility

[21] Aas, J.: Let's Encrypt is Trusted. [online], Říjen 2015, cit. 24.3.2018. Dostupné z: https://letsencrypt.org/2015/10/19/lets-encrypt-is-trusted.html

[22] Hafner, R.: How to Create Totally Secure Cookies. [online], cit. 28.4.2018. Dostupné z: http://blog.teamtreehouse.com/how-to-create-totally-secure-cookies

[23] Bütler, I.: SameSite Cookie attribute? [online], cit. 28.4.2018. Dostupné z: https://medium.com/compass-security/samesite-cookie-attribute-33b3bfeaeb95

[24] West, M.; Medley, J.: Content Security Policy. [online], cit. 16.4.2018. Dostupné z: https://developers.google.com/web/fundamentals/security/csp/

[25] Mozilla: HTTP Strict Transport Security. [online], cit. 28.4.2018. Dostupné z: https://infosec.mozilla.org/guidelines/web_security#http-strict-transport-security

[26] Google: Zobrazení a omezení zařízení kompatibilních s vaší aplikací. [online], cit. 31.3.2018. Dostupné z: https://support.google.com/googleplay/android-developer/answer/7353455

[27] Google: Compatibility Test Suite. [online], cit. 31.3.2018. Dostupné z: https://source.android.com/compatibility/cts/

[28] Agency, E. G. N. S. S.: What is GNSS? [online], cit. 6.5.2018. Dostupné z: https://www.gsa.europa.eu/european-gnss/what-gnss

[29] Fleischer, P.: Greater choice for wireless access point owners. [online], cit. 30.4.2018. Dostupné z: https://googleblog.blogspot.cz/2011/11/greater-choice-for-wireless-access.html

Příloha A

Uživatelská a instalační příručka

A.1 Uživatelská příručka

Pro nainstalování Android aplikace je vyžadován telefon s Androidem 5.0, GPS, Wi-Fi a podporou mobilního signálu. Aplikace funguje nejlépe na zařízeních s Androidem 8.0 a novějším, podporou 4G, dual-band Wi-Fi a 6 palcovým displejem.

Android aplikace můžete nainstalovat buď pomocí přiloženého APK souboru, nebo z Google Play. Pro snadnější nalezení aplikace na Google Play můžete využít odkaz http://get.signals.adsamcik.com.

Po nainstalování aplikace ji spusťte. Tutoriál v aplikaci vás provede základním ovládáním aplikace.

A.2 Instalační příručka

Tato příručka popisuje sestavení Android aplikace i její serverové části na platformě Windows.

A.2.1 Požadavky

- Visual Studio 2017 a novější,
- server RavenDB 4.0,
- zařízení s operačním systémem Microsoft Windows, nebo Linux s procesorovou instrukční sadou x86, x86-64, nebo ARM32,
- .NET Core Runtime 2.0.

Nainstalujte Visual Studio 2017. Pomocí Visual Studia 2017, nebo novější, sestavte zdrojové kódy do spustitelného formátu. V Publish vytvořte

profil aplikace s publikováním do složky dle vaší volby v konfiguraci „Release“ s cílovým frameworkem „netcoreapp2.0“ a cílovým běhovým rozhraním dle vaší volby. Pokud cílové rozhraní nenaleznete, je nutné ho přidat do souboru „Signals.csproj“. Po nastavení publikace aplikaci publikujte. Vytvořené soubory nahrajte na server dle vaší volby. Na serveru spusťe databázový server RavenDB. Server počítá s konfigurací RavenDB na IPv4 adrese 127.0.0.1 a portu 8080. Stáhněte a nainstalujte .NET Core Runtime 2.0. Přejděte do složky, do které jste nahráli soubory vygenerované pomocí publish, a spusťte server pomocí příkazu „dotnet Signals“.

Pro sestavení Android aplikace budete potřebovat tyto části:

- Android Studio 3.1 a novější,
- server RavenDB 4.0,
- platformu s operačním systémem Windows, nebo Linux a procesorovou architekturou x86, nebo x86-64, nebo AArch32,
- .NET Core Runtime 2.0.

Příloha B

Seznam použitých zkratek

ANR Application Not Responding.

API Application Programming Interface.

APK Android application package.

CI Continuous integration.

CSP Content Security Policy.

DHE Diffie–Hellman key exchange.

ECDHE Elliptic Curve Diffie-Hellman Ephemeral.

GNSS global navigation satellite system.

GPX GPS Exchange Format.

HDD Hard disk drive.

HSTS HTTP Strict Transport Security.

HTML Hypertext Markup Language.

HTTPS Hypertext Transfer Protocol Secure.

JSON JavaScript Object Notation.

MVC Model–view–controller.

PHP PHP: Hypertext Preprocessor.

PHP PHP: Hypertext Preprocessor.

SD Secure Digital.

SSD Solid-state drive.

SŘBD systém řízení báze dat (database management system).

TLS Transport Layer Security.

VPS Virtual Private Server.

XSS Cross-site scripting.

Příloha C

Vývojové nástroje

C.0.1 Android Studio

C.0.2 Git

C.0.3 GitHub

C.0.4 Visual Studio

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `apk` ..... adresář s instalačním balíčkem Android aplikace
  - `signals.apk` ..... instalační balíček Android aplikace
- `src` ..... adresář obsahující zdrojové soubory
  - `android` ..... zdrojové kódy implementace Android aplikace
  - `server` ..... zdrojové kódy implementace serverové aplikace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF