FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Diplomová práce

Automatizace dopravního průzkumu ulic

Studijní program: N0613A140028 Informační technologie
Autor práce: **Bc. Tomáš Krechler**
Vedoucí práce: Ing. Jan Kolaja, Ph.D.
Ústav nových technologií a aplikované informatiky

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání diplomové práce

Automatizace dopravního průzkumu ulic

Jméno a příjmení:	**Bc. Tomáš Krechler**
Osobní číslo:	M21000161
Studijní program:	N0613A140028 Informační technologie
Zadávající katedra:	Ústav nových technologií a aplikované informatiky
Akademický rok:	2022/2023

Zásady pro vypracování:

1. Proveďte rešerši dostupných aplikací pro sběr informací o dopravní vytíženocti městkých ulic.
2. Navrhněte mobilní aplikaci, která bude snímat polohu uživatele chodce a hladinu okolního hluku. Uživatel by měl mít možnost zadávat okamžitě i zpětně další parametry, které není možné zaznamenat strojově, např. přítomnost chodníku, jeho šířka, pohodlí, přítomnost zeleně apod. Aplikaci implementujte.
3. Vytvořte webovou aplikaci, která bude naměřená data shromažďovat a rovněž zobrazovat.
4. Zajistěte bezpečnost serveru i aplikace, navrhněte způsob anonymizace naměřených dat ve webové aplikaci a tento implementujte.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	40 – 50 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] Developer Guides. Android Developers [online]. [vid. 2021-05-20]. Dostupné z: https://developer.android.com/guide

[2] CENEK, Petr. Použití lineárního referenčního systému pro popis dopravních sítí, Dostupné z http://gisak.vsb.cz/GIS_Ostrava/GIS_Ova_2000/Sbornik/Cenek/Referat.htm

Vedoucí práce:	Ing. Jan Kolaja, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	12. října 2022
Předpokládaný termín odevzdání:	22. května 2023

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2022

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské/diplomové/rigorózní/disertační práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše. Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s §47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

Datum: 22.5.2021

Podpis: Bc. Tomáš Krechler

Poděkování

V první řadě bych rád poděkoval vedoucímu mé diplomové práce panu Ing. Janu Kolajovi Ph.D. za podnětné rady, testování aplikace, pomoc při zpracovávání práce, a za čas, který mi věnoval. Dále bych chtěl poděkovat manželce Bc. Kateřině Krechlerové, MBA za podporu a trpělivost v průběhu tvorby. Současně bych chtěl poděkovat všem testerům, kteří aplikaci otestovali a nasbírali data z různých místech Česka.

Abstrakt:

Diplomová práce se zabývá tvorbou systému pro dopravní průzkum ulic, ten je tvořen mobilní Android aplikací s webovou Node.js aplikací. Teoretická část se zaměřuje na již existující aplikace a projekty a jejich použití, s důrazem na projekt Noise-Planet. Následuje popis architektury a přehled verzí, typů práce s daty, senzorů a stavebních bloků aplikací systému Android. Byly popsány možnosti komunikace mezi klientem a serverem, přesněji protokoly a technologie a formáty přenášených dat. Dále byly vymezeny architektury a běhová prostředí pro webové aplikace. Součástí předchozích kapitol také jsou doporučení ohledně bezpečnosti dat a použití těchto technologií. Několik stran se věnuje popisu, co je to GIS a jak může být využit v tomto projektu.

Praktická část detailně popisuje celkovou implementaci obou aplikací, návrhy příslušných databází. Mapy jak v mobilní, tak ve webové aplikace používají knihovny a funkce Google Maps. Na tyto mapy jsou v mobilní aplikaci vykreslovány cesty uživatele a ve webové aplikaci jsou data shromažďována a utvářena do teplotních map. Součástí jsou i implementace bezpečnostních opatření. Aplikace byla testována v terénu několika uživateli, koncept byl ověřen. Práce rozebírá výsledky různých typů a způsobů měření, na základě kterých byly naimplementovány úpravy, případně byla navržena doporučení pro další rozšíření a vylepšení aplikací.

Klíčová slova:

mobilní aplikace, webová aplikace, Android, iOS, Node.js, bezpečnost, hluk, hlasitost, GIS, geografický informační systém, Noise-Planet, Google Maps, Google mapy, ulice, doprava, průzkum, cesta, teplotní mapa, automatizace, server, komunikace, databáze, senzor, REST, GPS, mikrofon, SQLite, MariaDB

Abstract:

The diploma thesis deals with the creation of a system for traffic surveying of streets, which is composed of a mobile Android application and a web Node.js application. The theoretical part focuses on existing applications and projects and their use, with an emphasis on the Noise-Planet project. It is followed by a description of the architecture and an overview of versions, types of work with data, sensors, and building blocks of the Android system applications. The possibilities of communication between the client and the server were described, more precisely the protocols, technologies, and formats of transmitted data. Furthermore, architectures and runtime environments for web applications were defined. Part of the previous chapters also includes recommendations regarding data security and the correct use of these technologies. Several pages are devoted to describing what GIS is and how it can be used in this project.

The practical part describes in detail the overall implementation of both applications and designs of databases. Maps in both the mobile and web applications utilize libraries and features of Google Maps. These maps are used in the mobile application to plot the user's routes and in the web application to gather data and visualize into heat maps. Implementations of security measures are also included. The application was field tested by several users and the concept was validated. The work discusses the results of various types and methods of measurement, based on which modifications were implemented, or recommendations were proposed for further expansion and improvement of the applications.

Keywords:

mobile application, web application, Android, iOS, Node.js, security, noise, volume, GIS, geographic information system, Noise-Planet, Google Maps, Google maps, street, traffic, survey, path, heat map, automation, server, communication, database, sensor, REST, GPS, microphone, SQLite, MariaDB

Obsah

Seznam obrázků

Seznam tabulek

Seznam kódů

Seznam rovnic

Seznam zkratek

z. s. .....Zapsaný spolek
JSON .....JavaScript Object Notation
HTTP.....Hypertext Transfer Protocol
HTTPS.....HTTP Secure
ART.....Android RunTime
HAL .....Hardware Abstraction Layer
API .....Application Programming Interface
DEX .....Dalvik Executable Format
AOT .....Ahead-Of-Time
JIT .....Just-In-Time
GC.....Garbage Collector
HTML .....Hypertext Markup Language
XML.....eXtensible Markup Language
SMS.....Short Message Service
OS .....Operační Systém
CRUD.....Create, Read (Retrieve), Update, Delete
AWS.....Amazon Web Services

GPS ..... Global Positioning Systém (Globální Polohovací Systém)
ARM..... Advanced RISC Machine
KMM..... Kotlin Multiplatform for Mobile
REST..... Representational State Transfer
SOAP ..... Subjective, Objective, Assessment and Plan
XSLT..... eXtensible Stylesheet Language Transformations
XSD..... XML Schema Definition
SSH ..... Secure Shell
FTP..... File Transfer Protocol
SFTP..... Secure FTP
IIS..... Internet Information Services
TLS ..... Transport Layer Security
MVC..... Model-View-Controller
DOM ..... Document Object Model
CSS ..... Cascading Style Sheets
JS..... JavaScript
SQL..... Structured Query Language
IP..... Internet Protocol
GIS ..... Geographic Information System (Geografický Informační Systém)
UTC..... Coordinated Universal Time
URL..... Uniform Resource Locators
YAML..... YAML Ain't Markup Language
NPM..... Node Package Manager
ASP ..... Active Server Pages
.NET..... Network Enabled Technologies
JWT..... JSON Web Token

1 Úvod

Tato diplomová práce se zabývá vývojem mobilní aplikace pro dopravní průzkum ulic. Má za cíl sbírat především data o hlučnosti různých částí měst. Dále také uživateli vyplněná data ohledně přítomnosti a šířky chodníku, pocitu bezpečí a zeleně v okolí jím prošlé cesty. V práci také jsou návrhy k vylepšením a dalšímu rozvoji aplikace.

Podnětem k tomuto tématu byla nezisková organizace Pěšky městem, z. s., jejímž hlavním cílem je hájit chůzi jako nejlepší způsob dopravy na krátkou vzdálenost. Proč tato vize? Protože ročně zemře v České republice až 10 000 lidí kvůli špatnému ovzduší, na každého Pražana (včetně dětí) připadá tři čtvrtě motorového vozidla a v neposlední řadě 80 % dětí nemá dostatek přirozeného pohybu. [1] Tato aplikace by mohla spojit příjemné s užitečným a dávat odměny za průchod městem, přičemž během získávání těchto odměn bude aplikace snímat okolní hluk.

Mobilní aplikace je napsána pro operační systém Android z důvodu, že tento systém je nejrozšířenější na světě, momentálně ho používá téměř tři čtvrtiny uživatelů chytrých telefonů. [2] K vývoji aplikací pro Android je vhodné využít Android Studio, které usnadní správu zdrojových souborů s kódem, texty i obrázky, a dokáže sestavit soubor, který lze na zařízeních rovnou instalovat. Android aplikace lze psát v Javě, nebo Kotlinu, já jsem zvolil jazyk Java. K funkčnosti požadovaného řešení je potřeba vytvořit bloky, z kterých se sestává aplikace. A to zaznamenávání hlasitosti v prostoru a čase. Z těchto bodů utvářet cesty, které jsou editovatelné uživatelem, který tam zároveň může doplnit jím vnímané vlastnosti cest. V neposlední řadě je nutné mít způsob odeslání dat na server, k tomu je využita serializace dat do formátu JSON a pro přenos je použit protokol HTTPS.

Ke shromažďování dat od uživatelů existuje webová aplikace běžící nepřetržitě na školním serveru. Webová aplikace kromě shromažďování dat dokáže i tato data zobrazit v podobě teplotních map, které překrývají mapy od společnosti Google. Také je možná filtrace dat dle období, nebo denní hodiny.

K zajištění anonymizace se rovnou z pořízeného záznamu zvuku vypočítá číslo značící hlasitost. Nedochází tedy k ukládání zvukových dat. Data se odesílají na server šifrovaným protokolem a bez dalších informací o uživateli jsou shromažďována v databázi. Data se běžným uživatelů zobrazují nefiltrovaně, pro filtraci dat dle času je zapotřebí přidělení oprávnění. Přístup do serverové databáze je zabezpečený heslem.

2 Existující řešení

2.1 Noise-Planet

Noise-Planet je výzkumný projekt a platforma, která se zaměřuje na studium hluku v prostředí a jeho dopad na lidské zdraví a kvalitu života. Projekt je založen na spolupráci mezi CNRS *(Centre national de la recherche scientifique)* a Université Gustave Eiffel ve Francii. Cílem projektu je poskytnout nástroje a metody pro sběr, analýzu a vizualizaci hlukových dat, aby se zlepšilo povědomí o hluku v prostředí a podpořil výzkum a plánování na snižování hluku.

Noise-Planet zahrnuje několik komponent:

NoiseCapture: Aplikace pro chytré telefony, která umožňuje uživatelům měřit úroveň hluku pomocí vestavěného mikrofonu chytrého telefonu, včetně jejich lokace. Aplikace umožňuje kalibrovat zařízení několika různými způsoby. Hlavním přínosem aplikace je sdílení dat uživatelů do komunitní databáze a mapy. [3]

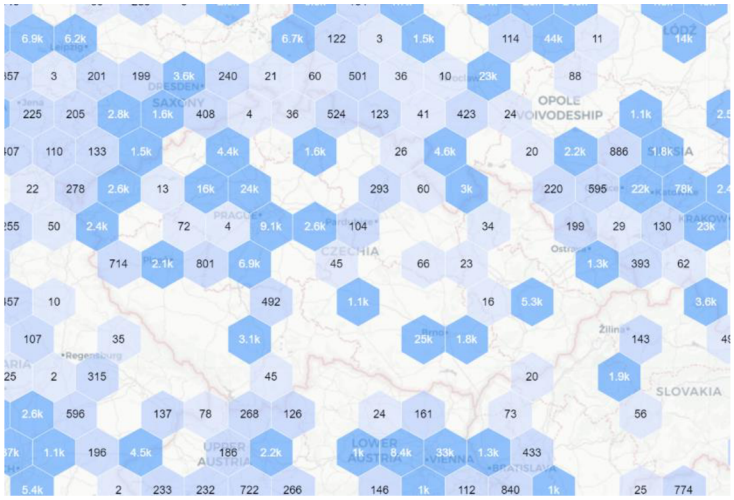
NoiseModelling: Je bezplatný *open-source* nástroj k vytváření map enviromentálního hluku, obzvláště velkých městských oblastí. Tento nástroj lze využít jako Java knihovnu, nebo přes webovou aplikaci. Je provázán s prostorovou databází H2GIS, případně PostGIS, které jsou uzpůsobeny k zpracování velkých objemů prostorových dat. [4]

OnoMap (Open noise Map): Noise-Planet zahrnuje otevřenou datovou platformu, která shromažďuje hluková data z celého světa a poskytuje je veřejnosti a výzkumníkům k analýze a studiu. Je to ústřední součást projektu Noise-Planet, protože poskytuje všechny nástroje pro ukládání, katalogizaci, sdílení a zobrazování dat o šumu na základě běžných a standardizovaných jazyků a kódování od OGC. [5]

V České republice také existují jednotlivci, kteří přispěli měřeními do komunitní mapy, ale jedná se spíše o ojedinělé případy, kdy uživatel měřil hluk v jednom místě. Výjimkou může být například měření v Sokolově na Obrázek 1.

Obrázek 1: Data v mapě projektu Noise-Planet – Sokolov

Agregovaná data jsou znázorněna na Obrázek 2. Zde každý šestiúhelník obsahuje hodnotu, která říká, kolik bylo naměřených vteřin v daných oblastech.



Obrázek 2: Data oblastí v mapě projektu Noise-Planet – Česká republika

Co se týče využití veřejně dostupných dat z těchto map a databází, nepodařilo se mi dohledat, že by byly využity v nějaké vědecké práci českých autorů. Ani použití dat nasbíraných v Česku.

2.2 Další aplikace

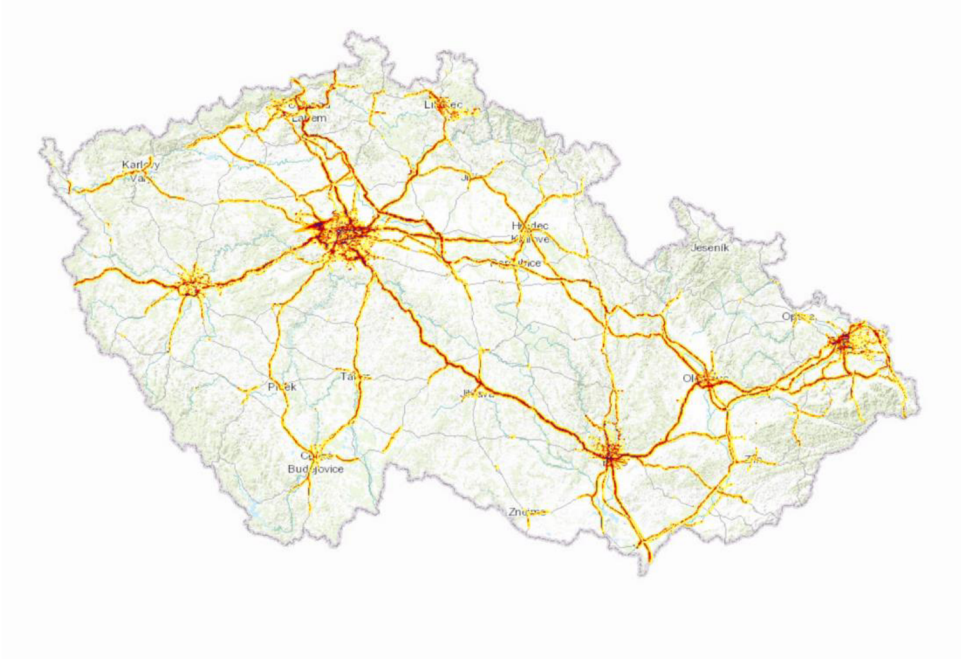
Na tržišti aplikací pro jak pro Android, tak pro iOS existují desítky aplikací ke snímání hladiny hluku a případného zobrazení spektrogramů, některé i s ladičkou pro hudební nástroje. Například: Sound Meter, Decibel X, Sound Meter Pro, Decibel Meter a další.

Další kategorií podobných aplikací jsou ty, které mají mapy a umožňují uživateli zaznamenávat různé prvky světa, či zakreslovat body, úsečky, křivky či plochy a k nim další data. Jejich případy použití jsou mimo jiné zaznamenávání stromů v parcích, výmolů a jiných vlastností silnic i za jízdy, nebo dopravního značení, osvětlení a dalších. Některé tyto aplikace nabízejí i propojení s jinými mapami, jako inženýrských sítí nebo katastru. Z českých to jsou ArcGIS, GeoGIS, GISELLA.

Dalším podobným zaměřením je sledování dopravy dálnic a silnic, či sledování okolí silnic za pomocí kamer. S tím souvisí i vyhodnocení nasbíraných dat za pomocí metod počítačového vidění. Z těchto dat pak lze také vytvářet různé mapy provozu, zeleně, dopravního značení a další. Na tato témata vzniklo v posledních letech několik bakalářských a diplomových prací.

Na odkaze *https://geoportal.mzcr.cz/SHM2017* lze najít hlukovou mapu z roku 2017 od Ministerstva zdravotnictví. Ta vznikla v rámci třetího kola strategických hlukových map. Webová aplikace také obsahuje přehled počtu hlukem zasažených osob, domů, školských a lůžkových zdravotnických zařízení v katastrech jednotlivých obcí.

Strategické hlukové mapy se zpracovávají v pětiletých cyklech pro okolí hlavních pozemních komunikací, po kterých projede více než 3 000 000 vozidel za rok, v okolí hlavních železničních tratí, po kterých projede více než 30 000 vlaků za rok, pro okolí hlavních letišť, s více než 50 000 vzlety nebo přistáními za rok a pro aglomerace Praha, Brno, Ostrava, Olomouc, Plzeň, Liberec a Ústí nad Labem – Teplice. Zpracovatelem těchto map byl Zdravotní ústav se sídlem v Ostravě. [6]



Obrázek 3: Hluková mapa – snímek z aplikace Ministerstva zdravotnictví

3 Platformy

Proč vůbec mobilní telefon? Jelikož je na světě přibližně 6,8 až 6,9 miliard registrovaných chytrých telefonů a každý telefon má mikrofon a většina má senzor k zaznamenávání pozice. [7] [8] A přibližně 68 % populace Země (5,44 miliard) má chytrý telefon. To je tříprocentní nárůst oproti minulému roku. [9] Zároveň je to nepoužívanější zařízení k přístupu k internetu, používá ho 91 % uživatelů. [10]

Alternativou by mohl být sběr pomocí specializovaných drobných zařízení rozmístěných po městě, případně nějaká specifická zařízení vytvořená ze senzorů a například Arduina, nebo Raspberry Pi. Ale to by bylo náročné, co se týče distribuce, a navíc cenově nákladné v porovnání s distribucí aplikace přes tržiště mobilních telefonů. Výhodou telefonu je, že už ho vlastní necelých 80 % české populace a aplikaci stačí distribuovat přes tržiště aplikací. [11]

Nejrozšířenější platformy pro mobilní zařízení jsou Android od Google (72 % uživatelů), iOS od Apple (27 % uživatelů). Dále existují Windows Mobile od Microsoftu, KaiOS, Tizen od Samsungu a další. Tyto ale aktivně využívá méně než jedno procento uživatelů chytrých telefonů. [2]

Android je open-source platforma, která je používána na široké škále zařízení různých výrobců. Naopak iOS je uzavřený operační systém vyvinutý společností Apple pro své zařízení iPhone, iPad a iPod Touch. Je známý pro svou jednoduchost, stabilitu a bezpečnost. Celkově je hodnocení těchto operačních systémů značně subjektivní a závisí na preferencích uživatele a konkrétních potřebách.

Každá platforma má své vlastní uživatelské rozhraní a designové prvky, které se mohou lišit v intuitivnosti, přehlednosti a jednoduchosti používání. Stabilita a bezpečnost jsou důležité faktory při výběru mobilního operačního systému. iOS je často považován za jednu z nejstabilnějších platforem, zatímco Android může být náchylnější k chybám a malwaru kvůli různorodosti hardwaru a softwaru. Android nabízí širokou škálu zařízení různých výrobců s různými cenovými hladinami a specifikacemi, zatímco iOS je dostupný pouze na zařízeních od Apple. Platformy Android a iOS mají obrovské množství aplikací dostupných v jejich oficiálních obchodech. Tržiště pro Android Google Play obsahuje ke konci třetího čtvrtletí 3 553 050 aplikací, zatímco App Store pro Apple 1 642 759. Také existuje tržiště Amazon Appstore, které funguje pro Android a Windows Phone. To obsahuje 483 328 aplikací. [12]

3.1 Android

Hlavní výhodou i nevýhodou Androidu je vysoká rozmanitost zařízení. To dává uživatelům možnost volby mezi velkou škálou zařízení a přizpůsobení, ale na druhou stranu při velké rozmanitosti jak hardware, tak software, můžou nastat problémy v konzistenci funkcí zařízení. Fragmentace také zvyšuje potřebu různorodosti zdrojových souborů aplikací, viz v další podkapitole 3.1.2. Android poskytuje relativní volnost jak v přizpůsobení systému uživatelem (*widgety*, změna vzhledu, nastavení výchozích aplikací), tak ve vývoji aplikací a funkcí. Na druhou stranu může otevřený kód operačního systému usnadnit tvoření škodlivého kódu.

Tento operační systém umožňuje uživatelům instalovat a používat aplikace z oficiálního obchodu Google Play, které poskytuje téměř okamžitý přístup k obrovskému publiku, ale i z jiných zdrojů (přímo stažené z internetu, nakopírované z disku atd.), po potvrzení uživatelem.

Díky komunitě programátorů, podpoře mateřské společnosti a rozsáhlé dokumentaci je možné najít mnoho užitečných zdrojů a návodů. Oficiální dokumentaci lze najít na *https://developer.android.com/docs*.

3.1.1 Architektura

Architektura Androidu je navržena tak, aby byla modulární a rozšiřitelná, což umožňuje výrobcům zařízení přizpůsobit a rozšířit Android pro jejich konkrétní zařízení. Tato architektura umožňuje různorodost zařízení, a tak nabízí širokou škálu možností pro vytváření aplikací. Software Androidu se dělí na šest klíčových komponent, znázorněných na Obrázek 4.

Vývojáři aplikací pro Android mohou využívat Javu nebo Kotlin pro psaní kódu aplikací, které jsou následně spouštěny na virtuálním stroji Dalvik (DVM) nebo Android Runtime (ART). Jeden z klíčových aspektů architektury Androidu je také tzv. Sandbox model, který zajišťuje, že jedna aplikace nemá přístup k datům nebo zdrojům jiných aplikací bez explicitního povolení uživatele. To zvyšuje zabezpečení a ochranu soukromí uživatelů. [13]

Obrázek 4: Architektura Androidu, převzato z https://developer.android.com/guide/platform

Základem platformy Android je linuxové jádro. Například Android Runtime (ART), využívá tohoto jádra pro základní operace jako je správa paměti na nízké úrovni nebo vytváření vláken. Použití linuxového jádra umožňuje systému Android využívat klíčové bezpečnostní funkce a umožňuje výrobcům zařízení vyvíjet ovladače hardwaru pro předem známé jádro.

Hardware Abstraction Layer (HAL) slouží jako standardizované rozhraní, které umožňuje hardwarovému vybavení zařízení komunikovat s Java API (Application Programming Interface) na vyšší úrovni. HAL se skládá z několika modulů, z nichž každý implementuje rozhraní pro konkrétní typ hardwarové komponenty, jako je kamera nebo modul Bluetooth. Pokud je potřeba přístup k hardwaru zařízení, systém Android načte odpovídající modul knihovny pro danou hardwarovou komponentu.

U zařízení se systémem Android verze 5.0 nebo vyšší běží každá aplikace ve vlastním procesu a s vlastní instancí Android Runtime. ART je navržen tak, aby spouštěl více virtuálních strojů na zařízeních s omezenou pamětí spouštěním souborů *Dalvik Executable format* (DEX), což je formát bajtového kódu navržený speciálně pro Android. Tento formát je optimalizován pro minimální nároky na paměť. Nástroje pro tvorbu, jako je d8, kompilují zdrojové kódy Java do bajtkódu DEX, který lze spustit na platformě Android. Mezi hlavní funkce ART patří kompilace předem (AOT) a *just-in-time* (JIT), sběr odpadu – uvolňování paměti přes *garbage collectory* (GC), podpora ladění aplikací (*profiler*, diagnostické hlášky a výjimky). Před verzí Android 5.0 byl virtuální stroj Dalvik běhovým prostředím systému Android. Aplikace běžící na ART by měla být schopna fungovat také na Dalviku, ale opačné tvrzení nemusí nutně platit. Android také obsahuje sadu základních běhových knihoven, které poskytují většinu funkcí programovacího jazyka Java, včetně některých funkcí jazyka Java 8, které používá Java API framework.

Mnoho základních komponent a služeb systému Android, jako je ART a HAL, je vytvořeno z nativního kódu, který vyžaduje nativní knihovny napsané v C a C++. Platforma Android poskytuje Java API rozhraní, která aplikacím zpřístupňují funkce některých z těchto nativních knihoven. Lze například přistupovat k OpenGL ES prostřednictvím rozhraní Java OpenGL API a přidat podporu pro kreslení a manipulaci s 2D a 3D grafikou. Pokud aplikace, vyžaduje kód C nebo C++, je možné použít Android NDK.

Všechny funkce operačního systému Android jsou dostupné prostřednictvím API rozhraní napsaných v jazyce Java. Tato API rozhraní slouží jako základní bloky pro vytváření aplikací pro Android, což usnadňuje opětovné využití základních, modulárních systémových komponent a služeb, včetně rozšiřitelného systému zobrazení pro tvorbu uživatelských rozhraní aplikací, správce prostředků pro přístup ke zdrojům (obrázky, XML soubory atd.), správce oznámení pro zobrazování upozornění ve stavovém řádku, a správce aktivit pro správu životního cyklu aplikací. Také nabízí společný zásobník navigace nebo prostředek k odesílání dat mezi aplikacemi.

Android přináší soubor základních aplikací pro e-mail, SMS zprávy, kalendáře, procházení internetu, kontakty a další. Tyto aplikace nemají žádný speciální status oproti aplikacím, které si uživatelé sami nainstalují. Aplikace od třetích stran mohou tedy nahradit výchozí webový prohlížeč, správce SMS a hovorů nebo dokonce výchozí klávesnici. Platí některé výjimky, jako je systémová aplikace Nastavení. Systémové aplikace fungují jako běžné uživatelské aplikace a poskytují funkce, které vývojáři mohou využívat v jejich aplikacích. Pokud by měla aplikace odesílat SMS zprávy, lze využít jakoukoliv již nainstalovanou SMS aplikaci k doručování zpráv. [14]

3.1.2 Verze (OS, rozlišení)

Data ke konci března 2023. Všechny verze až na verzi 13 mají sestupnou tendenci v počtu uživatelů. Tato poslední verze Androidu vyšla 15. srpna a stabilně se zvyšuje počet uživatelů tak, jak uživatelé aktualizují své telefony. Při tvorbě aplikace je nutné dbát na to, z jakých verzí Androidu jsou funkce, které aplikace bude potřebovat ke svému běhu. Ale také při komerčním využití aplikace myslet na to, kolik uživatelů bude moci aplikaci na svém systému spustit. Porovnání, jaké verze má jaké procento uživatelů Androidu, následuje na Tabulka 1. [15]

Verze Androidu	Rok vydání	Zastoupení na trhu Androidů
13 Tiramisu	2022	17,70 %
12 Snow Cone	2021	22,20 %
11 Red Velvet Cake	2020	21,69 %
10 Q	2019	17,16 %
9 Pie	2018	8,32 %
8.1 Oreo	2017	4,72 %
7 Nougat	2016	2,00 %
6 Marshmallow	2015	1,58 %

Tabulka 1: Verze Androidu – zastoupení na trhu

Aplikace jsou naspané ve starších verzích jsou kompatibilní s novějšími verzemi Androidu. Ale aplikace na novějších verzích nemusí fungovat na starších.

Při tvorbě aplikace je nutné vzít v potaz velkou různorodost velikostí a rozlišení obrazovek telefonů s operačním systémem Android. Při tvorbě aplikace je nutné vytvořit jednotlivé *layouts* pro různá rozlišení. V opačném případě by se mohla tlačítka, texty, a další prvky uživatelského rozhraní neúmyslně prolínat a překrývat se. Nejpoužívanější rozlišení obrazovek pro mobilní telefony jsou vypsána v následující Tabulka 2. [16]

Rozlišení	Zastoupení zařízení
360x800	11,17 %
390x844	6,84 %
414x986	5,87 %
412x915	5,56 %
393x873	4,87 %
360x780	4,10 %
374x812	3,67 %
385x854	3,34 %
428x926	3,11 %

Tabulka 2: Rozlišení obrazovek mobilních zařízení

3.1.3 Práce s daty

Operační systém Android nabízí několik způsobů, jak uchovávat data a jak s nimi pracovat. Android nabízí vestavěnou relační databázi SQLite, která nabízí prostředky pro uložení a správu strukturovaných dat v aplikaci. Je ideální pro ukládání malých a středně velkých dat. Díky své kompaktní velikosti a nízkým nárokům na systémové zdroje je SQLite ideální pro mobilní aplikace, kde je efektivita a rychlost klíčová.

Další možností jsou sdílené preference (*Shared preferences*). To je jednoduchá metoda ukládání dat v párech klíč-hodnota v rámci aplikace. Sdílené preference jsou uloženy v XML formátu, který je snadno čitelný strojově i lidsky. Tato data jsou dostupná pro všechny části aplikace bez ohledu na to, kde se nachází. A zůstávají uložena i po tom, co je aplikace ukončena. Tento systém lze využít k perzistentnímu uložení malých dat nebo stavu aplikace mezi jednotlivými sezeními.

V Androidu samozřejmě nechybí souborový systém, a tedy lze využít ukládání a správu různých typů dat, ať už ve formátu textových dokumentů, obrázků, audio souborů nebo jakýkoliv jiných typů souborů. Mezi způsoby, jak s těmito soubory pracovat, patří například File API, které poskytuje základní metody pro práci se soubory, AssetManager, který umožňuje přístup k souborům uloženým v aplikaci, a MediaStore, který umožňuje přístup a manipulaci s mediálními soubory.

Důležitým nástrojem pro poskytování dat je „poskytovatel obsahu“ (*Content Provider*). Tito poskytovatelé obsahu v systému Android představují mechanismus, který umožňuje sdílení dat mezi různými aplikacemi. Ty poskytnou data na vyžádání od jiné aplikace. Díky nim mohou aplikace číst a zapisovat data do společného úložiště, kterým může být SQLite databáze, soubor, nebo i místo v síti. K dotazování poskytovatele se využívá *string* v podobě *<prefix>://<authority>/<data_type>/<id>*, kde prefix je vždy „*content*“, *authority* je název poskytovatele, *data_type* je typ zdroje, o jaký žádáme (může se jednat o název objektu, nemusí to být primitivní datový typ) a *id* je identifikátor datového objektu. Pro užívání *Content Provider* v aplikace je nutné naimplementovat několik metod (*onCreate*, *getType* a metody operací CRUD).

V poslední řadě lze data ukládat přímo na cloudové úložiště. Uživatelé mohou přistupovat ke svým datům odkudkoli, kde mají připojení k internetu. Android podporuje mnoho různých cloudových služeb, jako je například Firebase, AWS (Amazon Web Services), nebo Google Cloud.

3.1.4 Senzory

Senzory v mobilech jsou fyzické komponenty, které slouží k sběru různých dat a informací z okolí zařízení. Tyto senzory jsou schopny poskytovat nezpracovaná data s vysokou přesností a správností a jsou užitečné, pokud chcete monitorovat trojrozměrný pohyb nebo polohu zařízení nebo chcete sledovat změny okolního prostředí v blízkosti zařízení. Například lze sledovat údaje z gravitačního senzoru zařízení a odvodit tak složitá uživatelská gesta a pohyby, jako je náklon, chvění, rotace nebo houpání. Podobně může aplikace počasí používat snímač teploty a vlhkost zařízení k výpočtu a hlášení rosného bodu nebo cestovní aplikace může používat snímač geomagnetického pole a akcelerometr k hlášení azimutu kompasu.

Většina zařízení Android obsahuje pohybové senzory, které vypovídají o poloze a změně rychlosti pohybu zařízení. Mezi ně patří lineární a úhlový akcelerometr. Ty měří zrychlení zařízení v různých směrech a dle různých os. Umožňuje zjišťování pohybu, orientace a detekci otřesů. Může vracet hodnoty včetně nebo bez započtení gravitační síly Země. Další z pohybových senzorů je senzor natočení zařízení, který vrací momentální natočení zařízení.

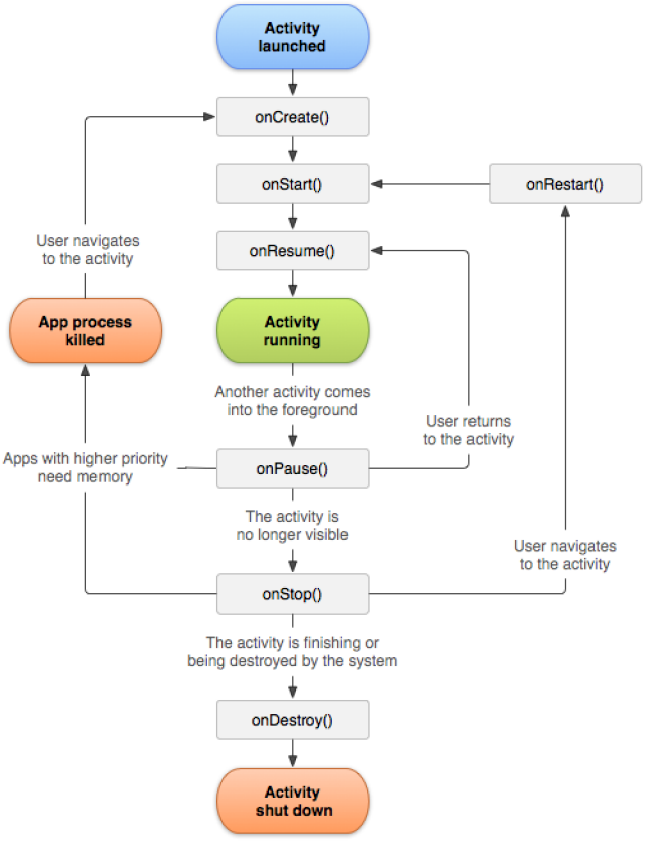
Pokud k pohybovým senzorům má zařízení i senzor magnetického pole se, dá využít společně s natočením zařízení k výpočtu směru, co se týče světových stran. K určení zeměpisné šířky a délky slouží poziční senzor. Ten využívá především signál ze satelitů GPS, ale může k určení zařízení využít i WiFi nebo mobilní data.

Méně často pak telefony mají senzor přiblížení, který bývá v horní části telefonu a slouží například k určení, zda uživatel dal telefon k uchu při hovoru, aby se zhasla obrazovka a omylem se něco nestalo. Některé telefony, nové modely velmi často, mají senzor na rozpoznání otisků prstů. Ten využívá především operační systém k odemčení telefonu a autorizaci uživatele. Zařízení také mohou mít hardwarovou podporu počítadla kroků. To by se dalo využít v aplikaci, kterou se zabývá tato práce.

Ze senzorů prostředí mají telefony nejčastěji světelný senzor, který bývá v horní části telefonu a snímá, kolik světla na něj dopadá. Výsledek vrací v luxech. Zřídka kdy telefony mají i senzory teploty, vlhkosti a tlaku. [17]

3.1.5 Vývoj aplikací

Pro vývoj aplikací v Androidu slouží několik klíčových stavebních jednotek, většina z nich se deklaruje v manifestu aplikace. První jednotkou je aktivita (*activity*). Tu musí mít aplikace, pokud chce mít uživatelské rozhraní. Aplikaci lze mít i bez aktivity, jako například službu na pozadí, ale v tom případě je doporučené, aby reagovala na nějaké akce telefonu skrze blok přijímač vysílání (*Broadcast Receiver*). Aktivita je základní stavebním kamenem aplikací, reprezentuje jednotlivé obrazovky a uživatelské rozhraní aplikace. Způsob, jakým jsou aktivity spouštěny a spojovány je základním modelem platformy. Na rozdíl od programovacích paradigmat, kde jsou programy spouštěny metodou *main*, Android spouští kód instance aktivity vyvoláním specifických metod konkrétních fází životního cyklu. Životní cyklus aplikace je na Obrázek 5. Aktivita slouží jako vstupní bod pro interakci člověka s aplikací. Každá aktivita poskytuje okno, kde vykresluje uživatelské rozhraní. To může obsahovat různé uživatelské prvky, jako jsou tlačítka, textová pole, seznamy a podobně. Jedna aktivita obvykle používá jedno rozložení (*layout*). A ačkoliv jednotlivé aktivity spolupracují k vytvoření soudržného uživatelského zážitku, jsou propojeny pouze volně s minimální závislostí. K otevření aktivity z jiné slouží úmysl (*intent*). Ten zároveň může předat data mezi aktivitami. [18]



Obrázek 5: Životní cyklus aktivity v Androidu, převzato z https://developer.android.com/guide/components/activities/activity-lifecycle

Rozložení definuje strukturu pro uživatelské rozhraní. Všechny objekty (zde nazývané *widgety*) jsou uspořádány v hierarchii kontejnerů a *widgetů*, zapsaném v XML souboru. Každý kontejner nebo *widget* obsahuje vlastnosti svého zobrazení, jako barvu, pozici, velikost, text a další. Tento systém umožňuje vytvořit flexibilní a responzivní uživatelské rozhraní. [19]

Úmysl (*intent*) je abstraktní popis operace, která se má provést. Může být využit k otevření nové aktivity, rozeslání zpráv naslouchajícím přijímačům vysílání nebo k nastartování a komunikací se službou. Ve zkratce se jedná o pasivní datovou strukturu, která obsahuje abstraktní popis akce, která se má provést. [20]

Dalšími komponenty aplikací jsou přijímače vysílání (*Broadcast Receiver*), které umožňují aplikaci reagovat na různé události, jako je například příchozí hovor nebo zpráva. *Broadcast Receivers* mohou spouštět další komponenty aplikace, jako jsou služby nebo aktivity. A poskytovatelé obsahu (*Content Provider*), které poskytují přístup k datům v rámci aplikace nebo sdílení dat mezi aplikacemi. [21] [22]

Služba je komponenta aplikace, která slouží k provádění dlouhotrvajících operací na pozadí. Neposkytuje žádné uživatelské rozhraní. Po spuštění může služba běžet určitou dobu i poté, co uživatel přejde na jinou aplikaci. Služba může například zpracovávat síťové transakce, přehrávat hudbu, provádět čtení a zápis souborů nebo komunikovat s poskytovatelem obsahu, to vše na pozadí. [23]

Posledním z důležitých komponent je fragment. Ten představuje opakovaně použitelnou část uživatelského rozhraní aplikace. Fragment definuje a spravuje své vlastní rozložení, má svůj vlastní životní cyklus a může zpracovávat své vlastní vstupní události. Fragmenty nemohou existovat samostatně, musí být součástí aktivity nebo jiného fragmentu. [24]

3.1.6 Bezpečnost

Zde je několik doporučení, jak zabezpečit aplikaci proti externím škůdcům, ale i proti nechtěné manipulaci s aplikací a daty. Tato doporučení nejsou všechna. Pro ochranu dat je doporučeno ukládat všechna soukromá uživatelská data do interního úložiště zařízení, které je v sandboxu pro každou aplikaci. Daná aplikace nemusí žádat o povolení k zobrazení těchto souborů a ostatní aplikace k nim nemají přístup. Navíc když uživatel odinstaluje aplikaci, zařízení smaže všechny soubory, které aplikace uložila do interního úložiště. Pro šetření pamětí telefonu je možné ukládat necitlivá data do externího úložiště. Pokud aplikace spolupracuje s vyměnitelným externím úložným zařízením, je nutné dbát na možnost odejmutí tohoto úložiště. Aplikace by měla obsahovat logiku k ověření dostupnosti úložného zařízení. Také by měla obsahovat logiku ke kontrole, zda data nebyla poškozena.

Pokud aplikace využívá přistupuje k nebo vytváří sdílené preference, měla by použít privátní mód. Tímto způsobem bude mít k informacím v souboru sdílených preferencí přístup pouze daná aplikace. Ke sdílení dat mezi aplikacemi by tato možnost neměla být používána. Pokud by tyto preference obsahovaly citlivá data je možné je zabezpečit šifrovanými sdílenými preferencemi, které obalují třídu *SharedPreferences*. Databázi SQLite je také možné zašifrovat.

Pokud uživatel přistupuje k citlivým datům uvnitř aplikace, nebo například něco platí, měla by aplikace ověřit, zda se jedná o oprávněnou osobu buď heslem, nebo biometrikou. Jestliže aplikace implementuje *ContentProvider* a nechce sdílet data všem aplikacím, je vhodné zakázat přístup ostatních aplikací k tomuto objektu. [25]

3.2 Cross-platform a iOS

Výhodou iOS je, že je vyvíjen a provozován pouze společností Apple, což zajišťuje jednotnost a konzistenci mezi zařízeními, co se týče hardwaru, softwaru a aktualizací. Také je obecně bezpečnější díky přísnému schvalovacímu procesu App Store a uzavřenější povaze platformy iOS. Navíc kvůli integraci hardwaru a softwaru jsou zařízení iOS často považována za optimalizovaná, což může vést k vyššímu výkonu a lepšímu výkonu baterie. Nevýhodou je omezená volba zařízení, přizpůsobení a personalizace. Apple také má striktnější omezení pro vývojáře aplikací. Přesto by bylo nevyužití této platformy na škodu, jelikož by se projekt připravil o více než čtvrtinu potenciálních uživatelů. Řešením, jak zasáhnout obě platformy je psát kód aplikace v některém z nástrojů, které umožní běh aplikace na obou platformách.

React Native je open-source rámec, který vytvořil Facebook a umožňuje vytvářet mobilní aplikace pro Android a iOS pomocí JavaScriptu a Reactu, který se přeloží do nativního kódu. Kód napsaný v React Native může být sdílen mezi oběma platformami s minimálními úpravami. Jelikož se jedná o „webovou“ aplikaci na telefonu, je zde složitější přístup například k senzorům telefonu. Tento a podobné problémy řeší anebo již vyřešila komunita programátorů. Flutter je sada nástrojů od společnosti Google, který umožňuje vytvářet nativní aplikace pro mobilní telefony, stolní počítače i vestavěná zařízení. A to vše z jednoho zdrojového kódu. Flutter používá jazyk Dart, který kompiluje do strojového kódu pro ARM nebo Intel stejně tak do Javascriptu, pro vysokou rychlost na všech zařízeních. Xamarin je rámec založený na technologii .NET od společnosti Microsoft. Xamarin umožňuje vytvářet nativní aplikace pro Android, iOS, tvOS, macOS i Windows pomocí jazyka C#. Xamarin už je nyní integrovaný přímo do sady nástrojů a knihoven .NET.

Kotlin Multiplatform Mobile (KMM) je rámec od JetBrains, který umožňuje sdílet části kódu mezi Android a iOS aplikacemi. KMM používá jazyk Kotlin a jeho platformě specifické rozšíření pro vytváření sdíleného kódu. Přesto uživatelské rozhraní a některá platformě specifická API stále vyžadují nativní implementaci. Dalšími rámci k vývoji multiplatformních aplikací je například Ionic, nebo NativeScript.

4 Komunikace

Komunikace mezi mobilním zařízením a serverem lze realizovat různými postupy a protokoly. Bezpečnost komunikace mezi serverem a mobilní aplikací zajišťuje v případě HTTP(S) vrstva TLS.

4.1 Způsoby

HTTP(S) (Hypertext Transfer Protocol (Secure)) je protokol aplikační vrstvy určený k přenosu informací mezi síťovými zařízeními a běží nad ostatními vrstvami zásobníku síťových protokolů. Funguje na principu klient pošle požadavek a server odešle odpověď. Tento protokol také umožňuje různé zlepšení rychlosti, plynulosti komunikace například využitím webové cache, proxy server a dalších. Existuje mnoho alternativ, ale tento protokol je nejrozšířenější. HTTPS je šifrovaná verze komunikace protokolu HTTP. Na serverech často bývá vystavené REST (Representational State Transfer) API, což je architektonický styl pro návrh webových aplikací zaměřen na data. Poskytuje jednotné rozhraní pro přístup k datům za pomoci standardních metod http jako jsou GET, POST, PUT, DELETE a další.

Websocket je dvousměrný komunikační protokol mezi klientem (mobilním zařízením) a serverem, který, po navázání komunikačního kanálu, umožňuje komunikaci v reálném čase mezi oběma stranami. Websocket je často používán v aplikacích, které vyžadují okamžitou synchronizaci dat mezi mobilním zařízením a serverem, například v chatovacích aplikacích nebo online hrách. Tato práce není závislá na rychlosti komunikace, a tak tento protokol nevyužije.

Push notifikace jsou způsob, jakým server může komunikovat s mobilním zařízením, aniž by bylo nutné, aby zařízení pravidelně posílalo požadavky na server. Server může poslat *push* notifikaci na zařízení, která se zobrazí na obrazovce, i když je aplikace uzavřena, a umožní tak komunikaci mezi serverem a zařízením v reálném čase. V této práci jde spíše o odeslání dat z mobilní aplikace na server.

4.2 Formáty dat

Komunikace mezi Android aplikací a webovou aplikací může využít jeden nebo více z různých formátů dat. Jedním z nejběžnějších formátů používaných v komunikaci je JavaScript Object Notation (JSON). Tento formát je znakově nenáročný (nízký počet znaků režie formátu) textový formát pro výměnu dat, který je často používán v komunikaci přes protokol HTTP(S) na vystavené REST API. JSON podporuje několik základních typů – číslo, řetězec, *boolean*, pole, objekty a *null* hodnoty. Je to velmi široce podporovaný formát, podporuje ho většina programovacích jazyků, i nesouvisejících s JavaScriptem. Při použití výchozího odsazení je snadno čitelný pro člověka. Pro jeho nízkou režii a jednoduchost zpracování je využit v komunikace mezi mobilní a webovou aplikací v této práci.

Podobným formátem je eXtensible Markup Language (XML). Jedná se o textový formát pro výměnu dat, který je široce používán v komunikaci mezi klientem a serverem. XML je obecněji použitelný než JSON, ale je obecně datově větší, což z něj činí méně populární volbu pro nové aplikace. Je rozšiřitelný, tedy umožňuje uživateli definovat vlastní struktury dat. Také podporuje vkládání metadata přímo do dokumentu. Jeho využití převažuje u architektur SOAP, případně jako formát ukládaných dat například kancelářských programů. Také existují eXtensible Stylesheet Language Transformations (XSLT) a XPath, které umožňují transformovat a dotazovat se na data respektive. Nabízí také automatickou validaci dat v dokumentu dle předem připravených schémat z XML Schema Definition (XSD). V poslední řadě podporuje i jmenné prostory, což umožňuje kombinovat definice z různých zdrojů do jednoho dokumentu bez konfliktu názvů.

Dalším způsobem jsou FormData. Toto rozhraní v JavaScriptu je cesta, jak odeslat data ve formátu párů klíč-hodnota. Formuláře jsou zpravidla používány pro odesílání dat získaných od uživatele, jako jsou textová pole, zaškrtávací políčka, tlačítka atd.

Posledním hlavním formátem je odeslání dat v podobě proudu binární dat, nebo zakódovaných binárních dat do textové podoby, jako je Base64. Ta jsou používána pro přenos binárních souborů, jako jsou obrázky, zvuky nebo videa. Vhodným způsobem přenosu dat může být protokol (SSH) File Transfer Protocol ((S)FTP), který je přímo určený k přenosu souborů, tedy i binárních dat.

5 Webová aplikace

Webové servery jsou software, který umožňuje hostování a poskytování webových stránek nebo webových aplikací na internetu. Tyto servery zpracovávají žádosti od klientů (typicky webových prohlížečů) a generují odpovídající odpovědi, které jsou zobrazeny uživatelům ve formě webových stránek. Také mohou poskytovat různé služby, jako je šifrování komunikace, správa uživatelských přístupových práv, cachování, load balancing a další. Webové servery mohou být statické, což znamená, že obsah stránek je pevně předdefinován a nemění se, nebo dynamické, což znamená, že obsah stránek je generován dynamicky na základě žádostí klientů a různých datových zdrojů.

Jedním z nejstarších a nejběžněji používaných webových serverů je Apache HTTP Server. Je open-source a nabízí širokou škálu funkcí a konfiguračních možností. Další ze serverů založených na Linuxu je NGINX. Což je populární webový server a reverse proxy server, známý pro svou vysokou rychlost, výkon a efektivitu, a je často používán pro nasazení webových aplikací s velkým provozem. Jeho součástí je load balancer. V ekosystému Microsoftu je používán webový server Internet Information Services (IIS) pro systémy Windows a Windows Server. Je především používán pro provoz webových aplikací postavených na technologii ASP.NET. Existuje mnoho různých nových architektur serverů, jako například LiteSpeed (pro vysoce zatížené aplikace) nebo Caddy (jednoduchý, rychlý s vestavěnou TLS).

5.1 Běhové prostředí

Existuje několik běhových prostředí a sad nástrojů, které může programátor použít k vytvoření webové aplikace. Běhová prostředí běžně obsahují kompilátor nebo interpret programovacího jazyka, knihovny a rámce, které poskytují běžné funkce jako je zpracování HTTP požadavků, případně připojení k různým databázím, či základní služby poskytované operačním systémem, jako vstup a výstup, správu procesů a paměti.

Mezi nejznámější prostředí patří například Node.js. To je běhové prostředí napsané v C, C++ a v JavaScriptu a umožňuje nativně spouštět JavaScript na straně serveru. Je široce používán ve vývoji webových aplikací díky své efektivitě a ekosystému balíčků (NPM). Od stejného tvůrce jako je Node.js, existuje prostředí Deno, které si zakládá na vysoké bezpečnosti. Je založen na jazyce Rust a stejně jako Node.js na V8 JavaScript engine. Node.js je běhové prostředí pro webovou aplikaci v této práci.

Dalšími známými skupinami technologií pro webové aplikace je jazyk Python s rámcem Django (zdarma, open-source, důraz na rychlý vývoj), nebo Flask (mikrorámec – malé, ale snadno rozšiřitelné jádro). Dále jazyk Ruby s Ruby on Rails, což je rámec založený na modelu model-pohled-ovladač (MVC). Obsahuje výchozí struktury pro databázi, webové služby a stránky. Oblíbenou technologií pro vývoj velkých podnikových aplikací je rámec Spring pro jazyk Java. Relativně populární je multiplatformní vývojářská platforma .NET, která kromě webových aplikací nabízí i nástroje pro tvorbu mobilních, desktopových, herních a dalších aplikací. Webovou aplikaci lze napsat pomocí dříve nejpopulárnějšího skriptovacího procesoru pro server PHP.

Rovněž pro klientskou stranu webové aplikace existují desítky rámců, mezi nejznámější patří React, Angular, Vue. Ty zjednodušují psaní kódu pro *frontend* aplikace, nabízí automatickou změnu stavů, virtuální Document Object Model (DOM) a další. Ovšem výsledný kód těchto rámců se překládá do JavaSctiptu, případně i do HTML a CSS. Místo těchto rámců lze k zobrazení stránek uživateli využít i šablonovacího *enginu*, jako například Pug nebo EJS. Ty jsou ovšem jsou vhodnější pro aplikace s nižší interaktivitou a komplexností. Jelikož se aplikace bude skládat z jedné obrazovky s mapou a maximálně ještě z registrace a přihlašování uživatelů, není nutné psát *frontendovou* aplikaci v některém z rámců.

Poslední důležitou částí je výběr databáze pro aplikaci. Zde by se dalo vybírat ze třech typů – relační (SQL) databáze (používají strukturu tabulek, které jsou propojeny klíči), dokumentové (NoSQL) databáze (ukládají data jako dokumenty, které se mohou vnořovat a obsahovat různé typy dat), nebo databáze časových řad (jejíž data jsou uspořádána v čase). Z dalších typů, které nejsou příliš vhodné pro tuto práci jsou například grafové databáze, nebo databáze ve formě klíč-hodnota. V práci jsem zvolil relační databázi MariaDB, ze dvou důvodů. První je, že jsou data v mobilní aplikaci také v relační databázi a druhým je přítomnost této databáze na serveru.

5.2 Bezpečnost

K zajištění bezpečnosti serveru od poškození dat a útočníků pomáhají tato doporučení. Hlavním prvkem ochrany serverů by měl být správně nakonfigurovaný Firewall. Ten chrání server před neoprávněným přístupem ze sítě (filtruje dle IP adres, nebo otvírá a zavírá různé porty pro různé protokoly) a brání různým útokům, jako například DDoS. Dalším důležitým prvkem ochrany serveru je pravidelná aktualizace, které přináší opravy bezpečnostních hrozeb, chyb a zranitelností. To zahrnuje aktualizace operačního systému, webového serveru, databáze, knihoven aplikací a aplikací samotných. Také je doporučené správně nastavit přístupová práva. Ty udělit pouze těm uživatelům nebo procesům, které je skutečně potřebují. V poslední řadě je dobré monitorovat a logovat server a data pravidelně zálohovat.

Bezpečnost webové aplikace se zajistí autentizací (zajištění, že pouze oprávnění uživatelé mají přístup k serveru) například skrze tokeny nebo službu OAuth, a autorizací, která omezuje přístup k různým částem API dle oprávnění uživatelů. Dále je nutné ošetřit veškerý vstup od uživatele. Vstupní data „očistit“ o prvky programovacích a dotazovacích jazyků. To brání útokům *SQL injection* nebo *cross-site scripting*. Pokud aplikace ukládá uživatelská hesla v databázi, určitě by neměla být v pouhé textové podobě, ale měla by být šifrována, například knihovnou *bcrypt*. Také celá databáze by měla mít přístup povolen pouze po zadání silného hesla.

6 GIS

Geografický informační systém (GIS) je počítačový systém (širší definici se jedná o technologii, zásady, standardy, lidské zdroje a další související činnosti) pro zachycování, ukládání, kontrolu a zobrazování dat týkajících se poloh na zemském povrchu. [26]

GIS používá jakékoli informace, které zahrnují polohu. Místo lze vyjádřit mnoha různými způsoby, jako je zeměpisná šířka a délka, adresa, PSČ a další. Pomocí GIS lze porovnávat a porovnávat mnoho různých typů informací, umožňuje lépe porozumět prostorovým vztahům a vzorcům. Systém může obsahovat údaje o lidech (jako počet obyvatel a jejich atributy), krajině (jako vodstvo, biosféra, zemědělství), budovách, sítích (silnice, inženýrské sítě). Tyto jednotlivé vrstvy lze porovnávat a určit, jak spolu souvisí (mapa průmyslu a mapa znečištění).

GIS aplikace zahrnují hardwarové i softwarové systémy. Tyto aplikace mohou zahrnovat kartografická data (mapy), fotografická a digitální data (snímky letecké, satelitní, z dronů nebo balonů), nebo data v tabulkových procesorech (demografie, statistiky). GIS umožňuje tato data propojovat do jedné mapy.

Dva hlavní typy formátů souborů GIS jsou rastrový a vektorový. Rastrové formáty jsou mřížky buněk nebo pixelů. Rastrové formáty jsou užitečné pro ukládání dat GIS, která se liší, jako je nadmořská výška nebo satelitní snímky. Vektorové formáty jsou polygony, které používají body (uzly) a čáry. Vektorové formáty jsou užitečné pro ukládání dat GIS s pevnými hranicemi, jako jsou čtvrti nebo ulice.

Technologie GIS může být použita k zobrazení prostorových vztahů a lineárních sítí. Lineární sítě, někdy nazývané geometrické sítě, jsou v GIS často reprezentovány silnicemi, případně sítěmi veřejných služeb. Čáry na mapě mohou značit i hranice ploch. Pro obecné a různorodé použití je třeba, aby údajová struktura popisující síť obsahovala všechny potřebné informace a byla přitom zcela univerzální. Těmto požadavkům vyhovuje lineární referenční systém. Základním prvkem lineárního referenčního systému je lineární systém (*linear datum*), který je zcela abstraktní strukturou a reprezentuje nejobecnější údaje o dopravní síti. Lineární systém podle definice představuje "… soubor objektů, které slouží jako základ pro umístění lineárního referenčního systému v reálném světě, definuje vztah databázové reprezentace k reálnému světu a poskytuje prostor pro transformace mezi různými referenčními systémy a geografickými reprezentacemi. Lineární systém se skládá ze základních sekcí (*anchor sections*) a základních bodů (*anchor points*), které leží v místě spojení sekcí anebo na konci sekcí. K lineárnímu systému nejsou přiřazeny žádné atributy". [27]

GIS musí zajistit, aby informace ze všech různých map a zdrojů byly zarovnány tak, aby do sebe zapadaly ve stejném měřítku. Dále musí často manipulovat s daty, protože různé mapy mají různé projekce. Všechny projekce vedou k určitému zkreslení. Přenesení zakřiveného trojrozměrného tvaru planety Země na rovný povrch nevyhnutelně vyžaduje rozšíření anebo smrštění oblastí. Mapa světa může zobrazovat buď správné velikosti, nebo správné tvary, ale neumí obojí.

Google Maps je významným příkladem aplikace GIS v praxi. Google mapy využívají GIS pro shromažďování, zpracování a zobrazování geografických dat. Tento systém je velmi obsáhlý a nabízí mnoho různých vrstev. Dvěma základními jsou základní mapa celého světa a mapa v podobě leteckých snímků. Nabízí také vrstvy na mapy, které se mění v reálném čase, jako například aktuální provoz na hlavních silnících a dopravních tazích, nebo síť hromadné dopravy ve velkých městech, včetně údajů o pravidelných linkách. Další rozsáhlou vrstvou je Street View. Ten umožňuje umístit kameru na většinu silnic a nabízí panoramatický rozhled do všech směrů. Také se lze přímo z jedné polohy přesunout na druhou a virtuálně tak procházet různé cesty. Součástí základní mapy jsou budovy, zeleň, silniční síť, vodstvo, ale i výšková mapa terénu. Mapu je možné zobrazovat jak jako plochu, tak jako kouli.

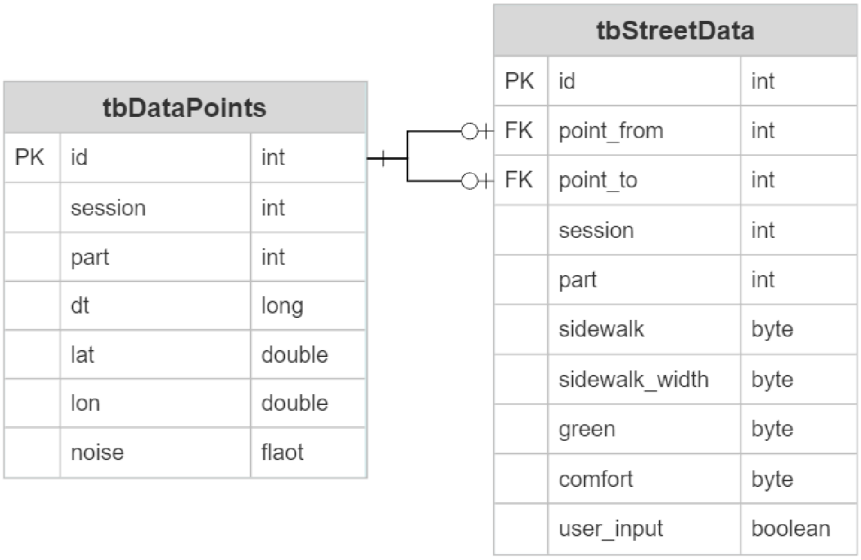
Android disponuje integrovaným API pro Google Maps, což umožňuje vývojářům snadno vytvářet aplikace s mapovými funkcemi. Toto API umožňuje zobrazování interaktivních map, georeferencování, trasování cest, vyhledávání míst a další funkce.

Aplikace pro Android mohou také využít GPS senzor, který je standardní součástí většiny moderních chytrých telefonů, pro získání přesné geografické polohy uživatele. Lze použít GIS aplikace, které poskytují specializované funkce, jako je sběr dat v terénu, geologické mapování, sledování divoké zvěře a mnoho dalšího. Viz kapitola 2.2.

7 Návrh a implementace Android aplikace

7.1 Návrh databáze

Databázi jsem navrhl jako dvě tabulky. První z nich (tbDataPoints) slouží k zaznamenávání jednotlivých měření. Záznam obsahuje pole *id* s vlastností *autoincrement*, které je primárním klíčem záznamu. Dále obsahuje pomocná data jako *session*, identifikátor relace aplikace, a *part*, který značí příslušnost k úseku cesty. Hlavní částí záznamů jsou ale pole *dt*, čas v podobě uplynulých milisekund od 1. 1. 1970 UTC, *lat* a *lon* – zeměpisná šířka a délka respektive, a *noise* – v decibelech zaznamenaná hlasitost v době pořízení záznamu.



Obrázek 6: Schéma tabulek aplikace – SQLite databáze

Druhá tabulka (*tbStreetData*) obsahuje stejně jako první tabulka pole *id*, *session* a *part*, jež mají úplně stejnou funkčnost jako výše popsané stejnojmenné pole. Dále obsahuje dva cizí klíče, které jsou reference do tabulky datových bodů a značí od jakého a do jakého bodu je daný úsek. V poslední řadě obsahuje čtyři pole *(sidewalk*, *sidewalk_width*, *green*, *comfort*), která slouží k uložení subjektivních dojmů uživatele ohledně ulic, a pole *user_input*, který značí, zda byla data uživatelem zadána. Může totiž nastat situace, že uživatel ve všem zvolí první možnost, jež má index 0.

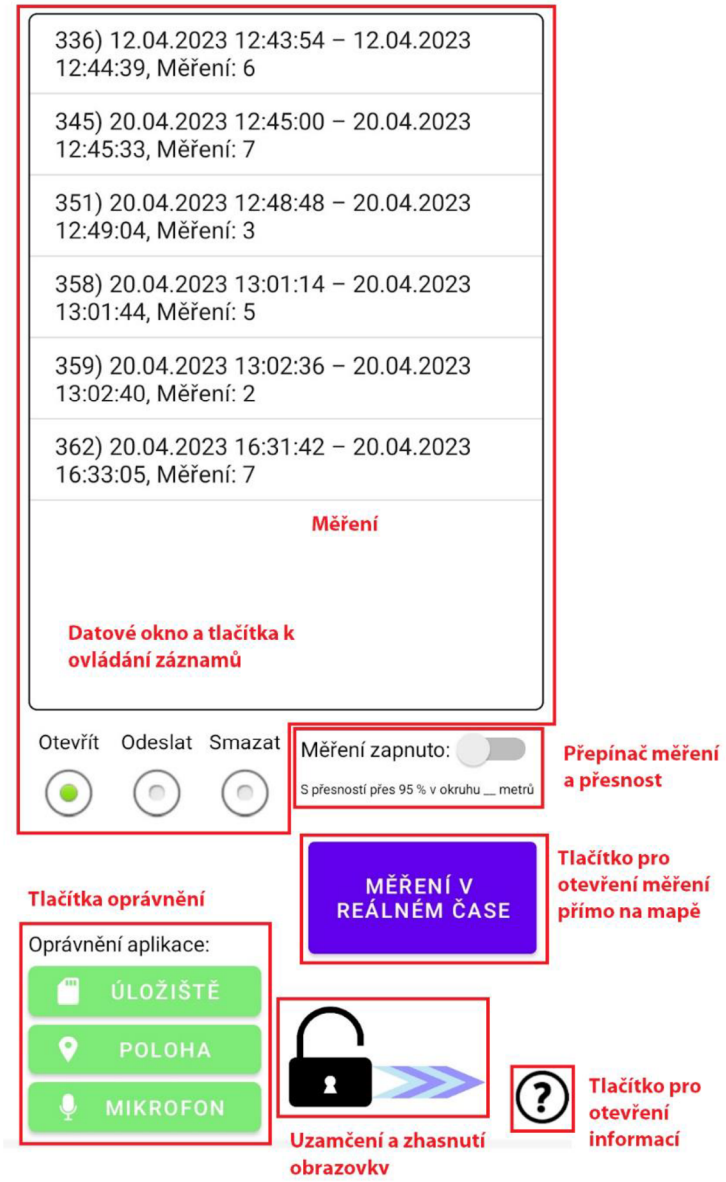
Důvod, proč tabulka ulic obsahuje identifikátor bodů od-do, na místo aby každý datový bod obsahoval referenci na identifikátor ulice, je, že se nejdříve ukládají samotné body a až po ukončení měření se celá cesta rozdělí na úseky a stačí aktualizovat pouze záznamy ulice, kterých je řádově méně. Ovšem tento důvod již není tolik platný, jelikož informace o příslušnosti k úseku byla do tabulek přidána až později a nyní se stejně aktualizuje příslušnost k úseku pro každý datový bod zpětně.

7.2 Implementace

7.2.1 Hlavní aktivita

Hlavní aktivita se skládá z několika sekcí. Jednou z nich je datové okno, kde se vypisují jednotlivá měření seskupená dle jednotlivých identifikátorů sezení. K tomuto oknu ještě patří tři rádiová tlačítka, která určují, jakým způsobem se bude aplikace chovat v případě kliknutí na záznam v datovém okně. V případě kliknutí na měření se stane jedna ze třech akcí. Pokud je zvolena možnost "Otevřít", otevře se *MapActivity* neboli obrazovka s Google Maps fragmentem, na který se zvolené měření vykreslí. K *Intentu*, který otevírá tuto aktivitu, se přidá informace o identifikátoru relace měření. Podrobněji v následující podkapitole. Další možností je "Odeslat", kde se po kliknutí zobrazí potvrzovací dialog, zda chce uživatel odeslat zkrácená data – bez subjektivních dat uživatele, i když byla už vyplněna. Poslední možností je "Smazat", což daná data (všechny datové body pro dané sezení) po potvrzovacím dialogu smaže. Sekce s přepínačem měření a s oprávněními jsou podrobněji popsány v dalších podkapitolách. Velké fialové tlačítko otevře skrze *Intent* *RealtimeMapActivity* s mapou a zadáváním subjektivních pocitů v reálném čase. Ve spodní části obrazovky je zámek, který lze přetáhnout směrem doprava, čímž lze obrazovku uzamknout a zhasnout (potemnit na nejnižší možnou svítivost). To je vhodné například po započatí měření, pokud chce uživatel šetřit baterii a zamezit dotyků omylem. Obrazovku lze opětovně odemknout potažením zámku směrem doleva. Vpravo dole už se nachází pouze otazník v kruhu, který po kliknutí otevře *InfoActivity* s informacemi o projektu.

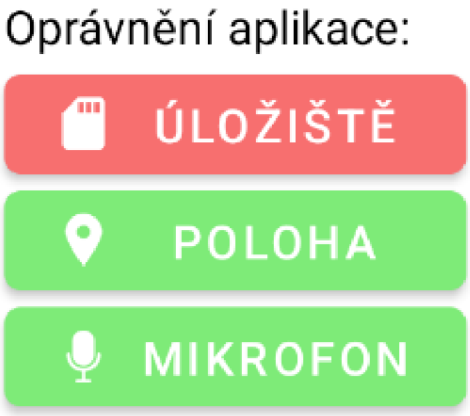
Hlavní aktivita se ještě stará o identifikátor sezení. Ten se pokusí přečíst ze *Shared Preferences* (perzistentní data aplikace ve formě klíč-hodnota) a pokud zde není, tak ho nastaví na jedna a uloží tamtéž. Tato hodnota se inkrementuje pokaždé, když je ukončeno měření a zároveň v přetížené metodě *onPause*.



Obrázek 7: Rozložení obrazovky hlavní aktivity

7.2.2 Získání oprávnění od uživatele

Jeden z prvních kroků, jak začít měřit, je získat povolení uživatele k využívání polohy a mikrofonu zařízení. Dodatečně je možné povolit i využití externího úložiště, kam je možné pak ukládat data aplikace.



Obrázek 8: Tlačítka oprávnění

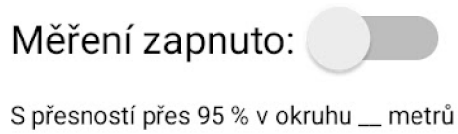
K tomu jsem vytvořil tři tlačítka v hlavní aktivitě aplikace. Ty při kliknutí od uživatele volají funkce *checkSelfPermission* a pokud není povolení uděleno, tak *requestPermissions* z knihovny *androidx*. Pokud ale uživatel rozklikl požadavek oprávnění a poté klikl mimo tabulku, nebo požadavek zamítl, už dle nastavení Androidu (od verze 11) nelze o oprávnění žádat opakovaně. [28] Toto jsem vyřešil tak, že při opakovaném pokusu o získání oprávnění se uživateli zobrazí dialog, zda chce otevřít v nastavení telefonu aplikaci a povolit oprávnění zde manuálně. Tento úkon se provede vytvořením a nastavením *Intentu*.

```
Intent intent = new Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS,
        Uri.fromParts("package", getPackageName(), null));
intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
startActivity(intent);
```

Kód 1: Vytvoření intentu k odeslání uživatele do nastavení telefonu

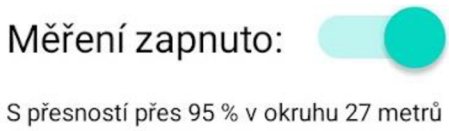
7.2.3 Automatické zaznamenání dat

Pokud jsou všechna oprávnění povolena, lze začít automatické sbírání dat. To uživatel může zapnout přímo v hlavní aktivitě klasickým přepínačem.



Obrázek 9: Přepínač zapnutí měření pozice a hlasitosti

Po zaznamenání prvního bodu se přepíše v uživatelském rozhraní přesnost na dvojnásobek vracené hodnoty v metrech. To z důvodu, že GPS senzor vrátí zeměpisnou šířku a délku a přesnost, která udává, že reálná poloha zařízení je v okruhu naměřené přesnosti s pravděpodobností 68,2 %, což odpovídá jedné směrodatné odchylce normálního rozložení. Tedy abychom získali přesnost s pravděpodobností reálného výskytu 95,4 %, musíme průměr kruhu zdvojnásobit. [29]



Obrázek 10: Přepínač měření ve stavu "zapnuto" s vyplněnou přesností lokalizace

Měření reálně započne nastavením *LocationManager* a *LocationListener* a poté jejich propojením při zažádání o aktualizace příkazem v Kód 2Kód 2: Zažádání o aktualizacích změny polohy. Kde *minTimeMs* je minimální počet milisekund mezi jednotlivými aktualizacemi pozice a *minDistanceM* je minimální posun od předchozí pozice v metrech. V aplikaci jsem nastavil tyto parametry na dva a půl tisíce milisekund a pět metrů. Tyto parametry jsem vypozoroval testováním, když jsem bral v úvahu přesnost trasy a zároveň, aby nebylo naměřených bodů mnoho.

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
minTimeMs, minDistanceM, locationListener);
```

Kód 2: Zažádání o aktualizacích změny polohy

Dále jsem vytvořil vlastní třídu *LocationChangeListener*, která implementuje *LocationListener* a jeho metody *onLocationChanged* a bez těla metody *onProviderEnabled* a *onProviderDisabled*. Do konstruktoru třídy je nutné přidat aktivitu, která požaduje zaznamenávání údajů a implementuje mé vlastní rozhraní *ILocationListenActivity*, které obsahuje jednu metodu a to *locationChanged*, ta je pak volána při každé změně polohy. Dále je vytvořen objekt *SoundMeter* a zavolána metoda *start*, která začne snímat vstup mikrofonu.

Samotná metoda *onLocationChanged*, ochuzena o komentáře a logování, je níže. Nejprve se získá a zaokrouhlí na šest desetinných míst zeměpisná šířka a délka. S přesností šesti desetinných míst lze určit místo na Zemi s přesností na přibližně jedenáct centimetrů (se zvyšující se vzdáleností od rovníku se přesnost zvyšuje). [30] V dalších krocích se zjistí počet milisekund od data 1. 1.1970 v systému a hlasitost v decibelech zaokrouhlena na dvě desetinná místa. Nakonec se zjistí, zda senzor poskytuje přesnost a ta se případně přidá do volání funkce příslušné aktivity.

```
@Override
public void onLocationChanged(Location loc) {
    double lat = Math.round(loc.getLatitude() * roundingGPS) /
roundingGPS;
    double lon = Math.round(loc.getLongitude() * roundingGPS) /
roundingGPS;
    Long ts = System.currentTimeMillis();
    double noise = Math.round(micRecording.getAmplitude() * roundingDB) /
roundingDB;
    if (activity != null) {
      if (loc.hasAccuracy()) {
        int acc = Math.round(2 * loc.getAccuracy());
        activity.locationChanged(ts, lat, lon, noise, acc);
      } else {
        activity.locationChanged(ts, lat, lon, noise, -1);
      }
    }
}
```

Kód 3: Metoda volána při změně pozice zařízení

Objekt *SoundMeter* obsahuje vyrovnávací paměť o délce osm tisíc, kde každý prvek má velikost šestnáct bitů. Délka osm tisíc je nastavena, aby se vždy uchovávala jedna vteřina zvukového záznamu a v tu chvíli, kdy si změna pozice vyžádá hlasitost, tak se vyrovnávací paměť přečte, spočítá se suma absolutních hodnot ze záznamu a z té se udělá průměr. Celá Rovnice 1 je dále. Průměr se vydělí konstantou 51805,5336, která je odvozena z předpokladu, že maximální hodnota šestnáctibitového měření x = 32767, což odpovídá 0,6325 Pa, a tedy referenční hodnota x = 1 je 0,00002 Pa. Výsledek se ještě vydělí referenční hodnotou, udělá se logaritmus o základu deset a vynásobí se dvaceti. Tak dostaneme orientačně hodnotu v decibelech v poměru k absolutnímu tichu. Jelikož telefony nejsou kalibrovány k měření absolutního ticha, není možné přesně měřit hlasitost bez prvotní kalibrace. Překvapivě ale naměřené hladiny celkem odpovídají profesionálním měření.

$$20 \cdot \log_{10}\left(\frac{\frac{\sum_{i=0}^{7999}|x_i|}{7999}}{\frac{51805.5336}{0.00002}}\right)$$

Rovnice 1: Výpočet hlasitosti v decibelech z paměti velikosti 8000

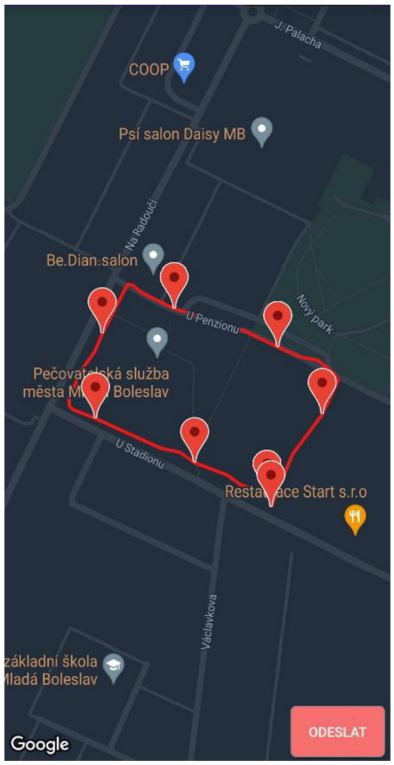
Po ukončení měření dat je volána metoda hlavní aktivity, aktualizující data v okně ve vrchní polovině obrazovky, kde s nimi může uživatel dále pracovat. Pokud ovšem cesta měla jen jeden bod, je odstraněna automaticky. Dále se body rozdělí na stometrové úseky – vytvoří se záznamy v tabulce *StreetData*. Nakonec se odstraní příznak nezhasínat obrazovku (nezamknout telefon při neaktivitě), inkrementuje se identifikátor relace a rozdělí se cesta na úseky.

Hlavní aktivita také implementuje přetíženou metodu *onPause*, kde se provede to samé, co v předchozím odstavci, a navíc se přepínač nastaví do pozice vypnuto. Toto je nutné dělat, jelikož ne vždy uživatel ukončí měření korektně přes přepínač. Může například stisknout hardwarové tlačítko zamknutí telefonu, nebo tlačítka systému pro prohlížení otevřených aplikací nebo návrat do menu. Také implementuje přetíženou metodu *onResume*, ve které je překreslení dat v datovém okně (můžou přibýt nová data v aktivitě měření v reálném čase na mapě) a také aktualizace tlačítek oprávnění, a to z důvodu změnění oprávnění v nastavení telefonu.

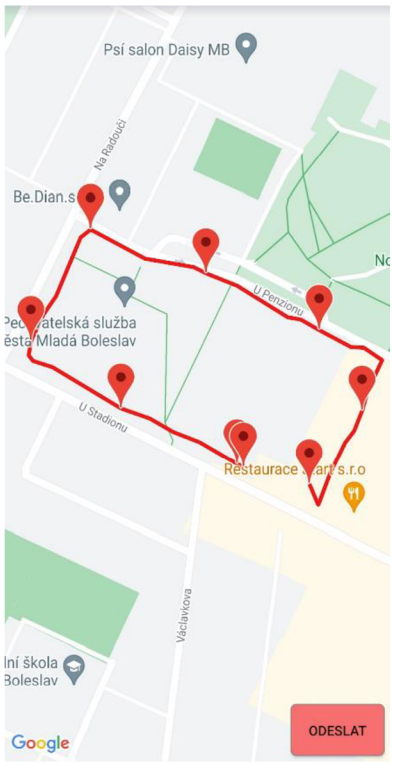
7.2.4 Google Maps aktivita

První spuštěná funkce je přetížená metoda *onCreate*, kde se získá identifikátor zvoleného měření a z databáze se přečtou data s tímto identifikátorem. Získají se jak data z tabulky *DataPoints,* tak ze *StreetData*. Poté se vytvoří fragment s Google mapami a čeká se, dokud se nenačtou. Po vyvolání události připravené mapy je zavolána přetížená metoda *onMapReady*, kde se zjistí, zda je nastaven noční režim, případně spořič baterie a podle toho se nastaví mapa v světlém, nebo tmavém zobrazení.

Google Maps poskytují možnost nadefinovat si vlastní vzhled mapy. Já jsem vzal jeden ze základních předdefinovaných s minimálními změnami. Definici vzhledu mapy lze vytvořit na *https://mapstyle.withgoogle.com/*, nebo nově na *https://console.cloud.google.com/* přímo v projektu pro danou aplikaci. Zde lze podrobně nadefinovat barvy a textury, ikony míst a mnoho dalších.



Obrázek 11: Cesta městem – noční mapa



Obrázek 12: Cesta městem – denní mapa

Následuje vykreslení bodů na mapu. Do prvního bodu se umístí značka a přidá se do seznamu pozic nově inicializované *Polyline*. Do tohoto seznamu se přidávají body, dokud se u bodu nezmění číslo úseku. V tom případě se umístí do tohoto bodu značka, přidá se do *Polyline*, ta se vykreslí a rovnou se inicializuje nová právě s touto pozicí. Vykreslovací algoritmus také kontroluje, zda daný úsek má pravdivostní hodnotu ve vlastnosti *user_input*, pokud ano, vykreslí se zeleně, jinak červeně. Takto se postupně vykreslí všechny body a úseky.

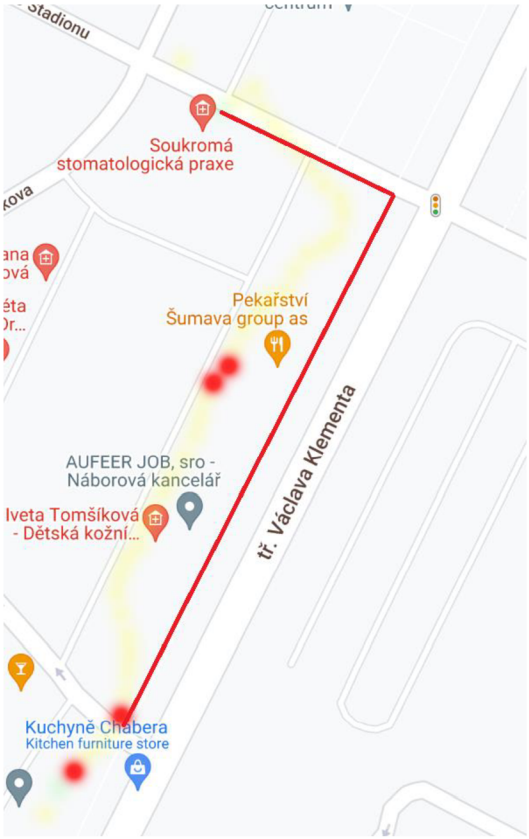
Po proběhnutí vykreslovacího algoritmu se zkontroluje, zda jsou všechny cesty zelené, v tom případě se nastaví tlačítko pro odeslání plnohodnotných dat na server také na zelenou barvu a je možné na něj kliknout. Poté se kamera přesune a přiblíží na poslední bod celé cesty. Zde by bylo možné například přesunout kameru do těžiště bodů, či poloviny maxim a zajistit, aby přiblížení bylo takové, aby se na obrazovku vešla celá cesta.

Na závěr se ještě v mapě nastaví, aby *Polyline* poslouchali na událost kliknutí. Přetížená metoda *onPolylineClick* přijímá jako vstupní parametr právě tu křivku, na kterou uživatel kliknul a ta obsahuje pouze list objektů *LatLng*, které jsou tvořeny zeměpisnou šířkou a délkou. Z tohoto listu vezmu první a poslední bod a jejich pozice. Z těch zjistím, jaké dva body to jsou, a z této informace už lze odvodit na jaký objekt *StreetData* uživatel klikl. Nyní se uživateli zobrazí okno s možnostmi zadání subjektivních pocitů pro daný úsek. Pokud pro tento úsek už byly někdy tyto hodnoty nastaveny, jsou zjištěny a dle nich předvybrány v rozevíracích seznamech dané možnosti. Toto vyskakovací okno kromě oněch čtyřech rozevíracích seznamů s možnostmi obsahuje dvě tlačítka, kde tlačítko "ULOŽIT" data uloží perzistentně do databáze a tlačítko "ZRUŠIT" dané změny v seznamech zahodí.

Obrázek 13: Vyskakovací okno k úpravě dat o daném úseku cesty

Na mapě je ještě naslouchánо události *onMapLongClick*, která je obsloužena tak, že se pokusí rozdělit úsek na dva v bodě nejblíže dlouhému kliknutí uživatele. Při dlouhém kliknutí na kamkoliv na mapu (i na obrazovku, jelikož mapový *fragment* zabírá celou plochu) se nejdříve projdou všechny body všech vykreslených křivek, které si při vykreslování ukládám do listu. Procházením těchto dat se najde nejbližší bod a příslušná křivka. Pokud byl klik od uživatele dále než sto metrů od nejbližšího bodu, vypíše se mu o tom zpráva a nic dalšího se nestane. V opačném případě se do nejbližšího bodu umístí značka a úsek se rozdělí na dva, které případně přejmou uživatelem vyplněná data z úseku původního. Také se původní křivka vymaže a vykreslí se dvě nové. Při kliknutí na značku se zobrazí vyplněný text při vytváření, v mém případě zobrazuji datum a čas pořízení tohoto datového bodu.

Vznikl požadavek (bohužel měsíc před termínem odevzdání této práce), aby šlo jednotlivými body posouvat nebo je “lepit” na ulice. Přesun by byl možný, jelikož uživatel ví, kudy šel a může opravit nepřesnosti měření GPS. Z testování bylo zjištěno, že přesnost se ve městě pohybuje od čtyř do deseti metrů, ale v případě blízkosti vysoké budovy byla přesnost nižší až k třiceti metrům. Případně tento problém řešit automatickým přesunem bodů na ulice, ale zde může být problém, že někde je chodník o deset metrů daleko od ulice v mapě. Ani jedna z těchto úprav tedy není součástí aplikace. Mám hypotézu, že s větším objemem dat od uživatelů, by se chyba naměřených dat zprůměrovala, navíc finálním výsledkem měření je teplotní mapa, kde chyba několik metrů nemá takový vliv na výsledný vzhled.

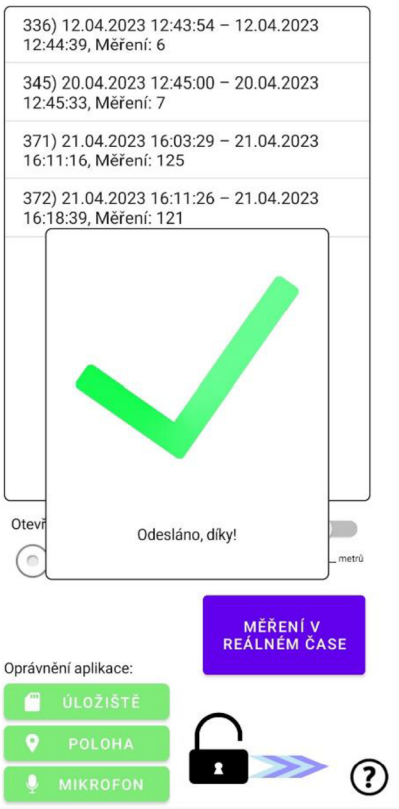


Obrázek 14: Nepřesná poziční data v okolí vysokých budov

7.2.5 Odeslání dat na server

K odesílání dat na server jsem vytvořil pomocnou třídu *HTTP*. Ta rozšiřuje třídu *AsyncTask* a umožňuje tak práci v novém vlákně nezávisle na hlavním vlákně aplikace s uživatelským rozhraním. Do konstruktoru třídy *HTTP* vstupuje URL, kam bude požadavek směřovat a rozhraní, které bude slouží pro komunikaci této asynchronní třídy s hlavním programovým vláknem. Čtení a zápis těla HTTP požadavků probíhá přes vstupní a výstupní zapisování do streamu.

Všechny aktivity (hlavní a obě aktivity s mapou), které chtějí odesílat data na server, musí implementovat rozhraní *ISendDataActivity*, které obsahuje tři definice metod: *createLoadingPopup*, *finishLoadingPopup* a *cancelLoadingPopup*. Tyto tři metody poté slouží k informování aktivity procesem na pozadí, že došlo ke změně stavu. První z nich slouží k informování, že započalo odesílání na server. Uživateli se zobrazí okno a animace načítání. Ta je v popředí obrazovky tak dlouho, dokud nepřijde od pomocné třídy *HTTP* obsluhující spojení se serverem druhá, nebo třetí z metod. Druhá metoda je pro danou aktivitu zavolána, když třída *HTTP*, úspěšně odešle data a od serveru dostane a přečte odpověď s kódem 200. Třetí pak když přijde kód 400, nebo se aplikace nedočká odpovědi do patnácti vteřin.



Obrázek 15: Úspěšné odeslání měření na server

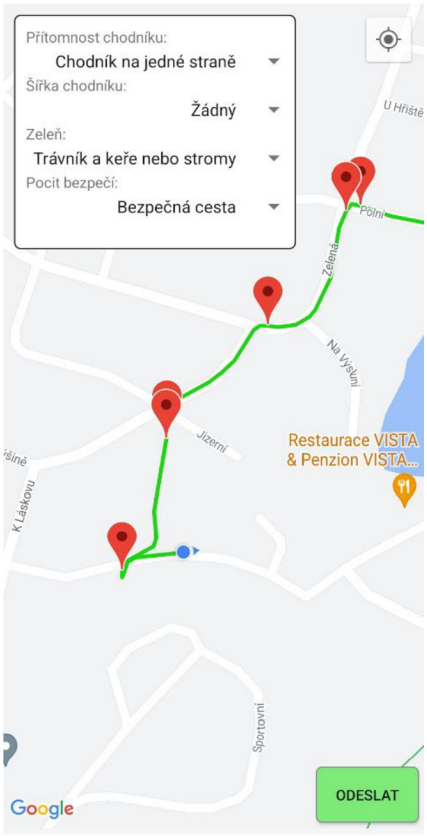
7.2.6 Realtime Google Maps aktivita

Její Hlavní účel této aktivity je v měření dat, ale zároveň zobrazování naměřené cesty uživateli přímo v mapě. Uživatel také vidí modrou tečku se šipkou, která udává jeho pozici a natočení ve světě (kam směřuje vrchní část jeho telefonu). Tato aktivita spojuje některé z funkcí hlavní aktivity a předchozí mapové aktivity. Z hlavní aktivity přebírá celý algoritmus měření, který se spouští automaticky ihned při otevření této aktivity. Podobný obsah mají přetížené metody *onPause* a *onResume*. Ty obsahují kód pro obsluhu značky ponechání nezamčeného telefonu, odregistrace naslouchávání událostem změny pozice a opětovné registrace a v poslední řadě odstranění osamocených záznamů v databázi. Přetížená metoda *onCreate* přečte a nastaví identifikátor relace, započne měření, nastaví dialog pro tlačítko odeslání a vyplní pole pro automatické vyplňování subjektivních dat o ulici.

Mapa se vytváří obdobně jako v *MapActivity*, akorát zde je navíc aktivní momentální zobrazení uživatele modrou tečkou s šipkou. Také se na mapu nastaví naslouchače událostí *onMapLongClick* a *onPolylineClick*. A stejně tak tato aktivita obsahuje metodu pro odeslání dat na server a výše zmíněné tři metody pro komunikaci s asynchronním vláknem třídy *HTTP*.

Hlavní rozdíl od dříve popisované mapové aktivity je v systému zaznamenávání hodnot a vykreslování křivek. Datové body se zaznamenávají stejně, se stejnými identifikátory a zeměpisnou šířkou a délkou, časem a hlasitostí. Ale v tabulce ulic se vytvoří záznam na začátku a s každým datovým bodem se aktualizuje jeho vlastnost *point_to.* A to proto, že uživatel může kdykoliv měření ukončit nebo odeslat. Vytvoření nového záznamu do tabulky ulic nastavena ve dvou případech a v obou se umísti značka na mapu do nově naměřeného bodu. Prvním případem je, že úsek už měří více než sto metrů. Druhým je, že uživatel změnil jakýkoliv z parametrů v okně vlastností cesty.

Další z rozdílů je také vykreslování křivek. Křivka se vykresluje ve stejný moment, jako se aktualizuje záznam ulice. Vždy, když přibude nový datový bod, se vezme seznam původní křivky, která se vymaže, přidá se k němu nová poloha a opět se vykreslí. Na novou křivku se nastaví naslouchač na kliknutí. Pokud je už záznam ulice finální a vytvoří se nový záznam, včetně umístění značky na mapě do posledního bodu, tak i poslední křivka už takto zůstane a měnit se bude křivka nová.



Obrázek 16: Obrazovka měření v reálném čase

Na obrázku je snímek obrazovky *RealtimeMapActivity*. V levém horním rohu je okno s možnými vlastnostmi cesty, které může uživatel nastavit. V pravém horním rohu je pak tlačítko, které se ve fragmentu mapy vytvoří samo, pokud je povoleno zobrazení aktuální polohy zařízení. To po stisknuté přesune kameru na uživatelovu pozici. V pravém dolním rohu je odeslání datových bodů včetně dodatečných dat ulic. Na mapě pak jsou značky pospojované křivkami. Většina značek je umístěna po dosažení úseku sto metrů, ale některé jsou přidány po změně automaticky vyplňovaných dat. Je vidět i poloha zařízení včetně směru chůze.

7.3 Bezpečnost

Obsah složek */Android/data*, kde jsou uložena data aplikací nelze v telefonu přímo zobrazit, je zobrazena hláška, že tento obsah lze zobrazit pouze v počítači. V počítači v obyčejném průzkumníku souborů, sice lze zobrazit, složky s daty aplikací, ale zde jsou viditelné pouze složky *cache*, která obsahuje binární soubory, a *files*, která je prázdná, i když reálně soubory obsahuje.

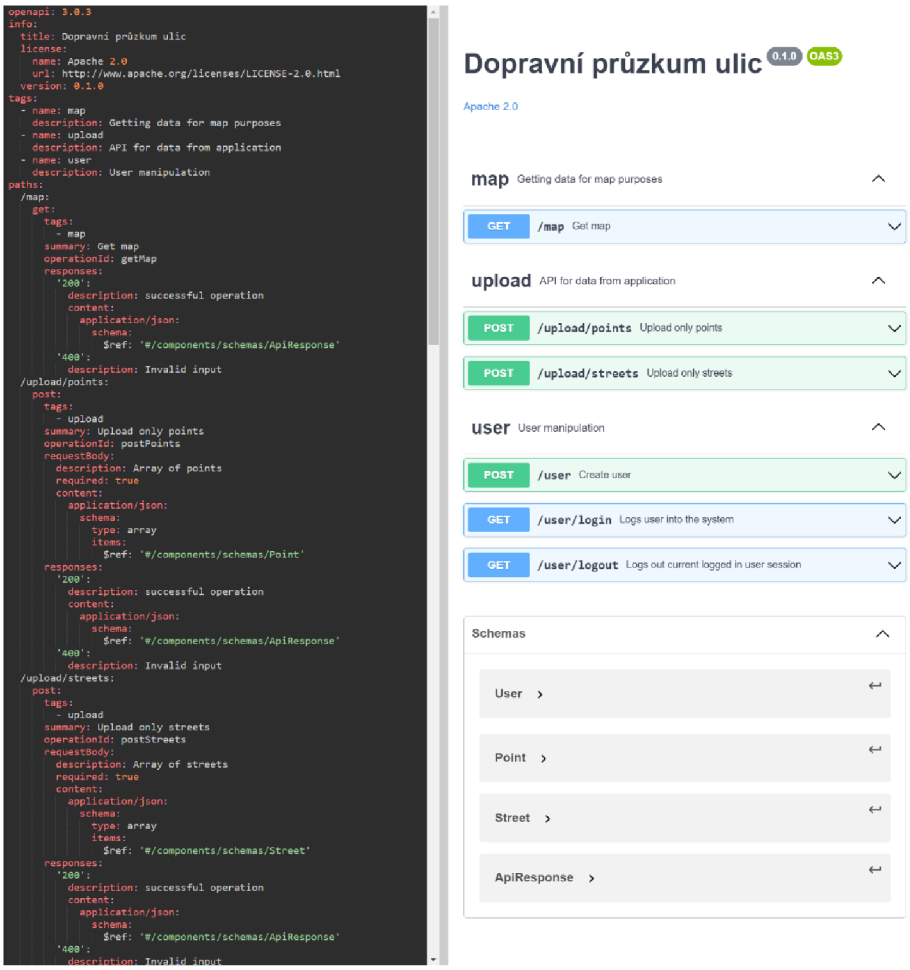
To ovšem není překážkou. Otevřít složku a číst obsah souborů lze například otevřením v Android Studiu, který otevře balíček pomocí příkazu *run-as com.packagename* a poté balíček překopírovat. Ale to pouze za předpokladu, že během sestavení aplikace byl parametr *debugable* v možnostech sestavení nastaven na hodnotu *true*. Alternativou, jak získat data, je udělat zálohu dat aplikace skrze příkaz *adb backup com.packagename*, poté je uživatel vyzván k zadání hesla, které by soubor zašifrovalo. Pokud ho ale nezadá, je vše jednodušší (i když i zašifrovaný soubor je možné rozšifrovat a rozbalit), stačí daný soubor pouze rozbalit, jelikož to jsou pouze data komprimované do souboru *tar*. Tato alternativa se ale také dá překazit, pokud je v *manifestu* aplikace nastaveno *android:allowBackup="false"*. [31]

Tyto bezpečnostní prvky lze ale obejít takzvaným "rootnutím" zařízení (*rooting device)*, což je pozměněný software (*boot.img*) telefonu tak, aby měl uživatel práva super uživatele (*super user*). Dle typu zařízení záleží na postupu, někdy je potřeba odemknout *bootloader*, kde odemčení smaže data, ty ale jdou obnovit ze zálohy. Tedy při správném postupu "rootnutí" telefonu, se data neztratí.

8 Implementace serveru Node.js a webové aplikace

Součástí implementace webové aplikaci na školní linuxový server bylo i přichystat prostředí. Po zažádání a přidělení domény, výpočetní kapacity a paměti, bylo nutné nainstalovat rámec pro běh aplikace a knihovny pro aplikaci nebo správu aplikace. Prvním krokem bylo nainstalování prostředí Node.js, který obsahuje balíčkovací systém NPM. Pomocí toho jsem nainstaloval knihovnu Express (minimální a flexibilní rámec webových aplikací Node.js) a MariaDB (knihovna ke správě připojení a dotazování do MaridDB databáze). Pro aplikaci jsem vytvořil adresář a kód sem odeslal ze svého počítače. Před spuštěním aplikace bylo ještě nutné ve Firewallu povolit port 5000, na kterém aplikace běží. Nyní se dala aplikace spustit a byla přístupná z internetu. Ovšem po ukončení spojení se serverem se ukončila i webová aplikace běžící v konzole. Řešením bylo stáhnutí a nainstalování knihovny PM2, která spustí aplikace skrze svůj manažer a umožňuje i další funkce jako monitoring stavu, zatížení aplikace a dalších informací. V poslední řadě bylo třeba nastavit, aby se webová aplikace Ulice spustila skrze PM2 ihned automaticky po zapnutí serveru.

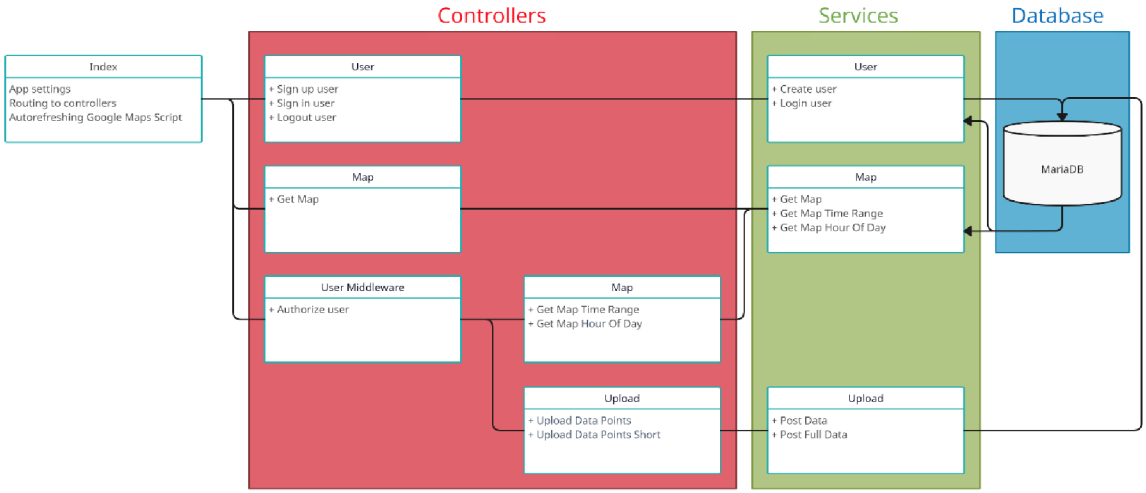
Můj postup vytvoření webové aplikace se dá rozdělit do několika kroků. Prvním bylo nadefinování si REST API, tedy jednotlivých objektů, jejich vlastností a datových typů, a cest používaných v aplikaci. K cestám zároveň doplnit informace o metodě požadavku (GET, POST a další) a o tom, zda požadavek bude obsahovat nějaké tělo a co v něm bude, případně jestli použije parametry v cestě. A v poslední řadě, co bude server vracet za odpověď. Na Obrázek 17 níže je online editor takové definice ve formě YAML souboru, společně s grafickým rozhraním, které se automaticky aktualizuje dle obsahu daného souboru. Tento obrázek také znázorňuje první verzi webové aplikace, dle které jsem vygeneroval kostru.



Obrázek 17: Webový editor REST API – editor.swagger.io

V tomto editoru se dá vytvořit z jednoho definičního souboru (formát YAML nebo i JSON) základní kostra webové aplikace pro několik různých jazyků a rámců. Patří sem mimo jiné ASP .NET Core v C#, Spring v Javě, Kotlin server, Node.js v Javascriptu, Flask v Pythonu a další. Editor je k nalezení na *https://editor.swagger.io/*. Tímto nástrojem jsem tedy vygeneroval Node.js aplikaci, která je takto funkční a pro dané cesty vrací vzorové objekty a je po nasazení možnost testovat funkčnost komunikace. Mezi vygenerovanými součástmi byly složky *Services* a *Controllers*, které obsahovaly příslušné soubory dle použitých *tags* v editoru u jednotlivých cest. Dále byl vygenerován *index.js*, který obsahuje směřování a nastavení aplikace, *writer.js*, který vytváří JSON odpověď, a *package.json*, což je soubor obsahující data o daném balíčku či aplikaci typický pro *npm*. Npm je správce balíčků pro JavaScript, výchozí správce balíčků pro prostředí Node.js.

Dalším krokem bylo upravit *index.js*, aby používal přímo server *Express* na místo *OAS3tools* a falešného (*mock*) API. V pozdější fázi projektu jsem doplnil *middleware* funkci pro autorizaci uživatele k používání určitých cest serveru. Výsledné schéma pro tok dat a požadavků je na Obrázek 18.



Obrázek 18: Schéma webové aplikace

8.1 Nahrání dat z aplikace

Pokračoval jsem doplněním ovladače pro nahrání dat a jeho příslušné služby. Tato služba si vytvoří připojení do lokální databáze MariaDB pomocí uživatelského jména, hesla, názvu hosta, názvu databáze a také limitu pro počet současných připojení. Poté v jak v metodě *PostData*, tak *PostFullData* vytvoří SQL dotaz pro vložení dat do databáze. Data jsou před vložením do dotazu "očištěna" tak, aby bylo znemožněno útoku typu *SQL injection*. Poté co proběhne korektně vložení záznamů do databáze, je spojení ukončeno a navrácen kód 200 v obou metodách tohoto ovladače. Veškeré asynchronní funkce jsou prováděny přes objekty *Promise*, ke kterému jsou připojeny *callback* jak pro korektní výstup (klíčové slovo *then*), tak pro chybu (*catch).*

8.2 Zobrazení mapy a naměřených dat

Tato funkčnost už se neskládá z pouhého ovladače a služby komunikující s databází, ale zapojuje do své funkce více komponent. Jednotlivé komponenty jsou tyto:

- Služba pro čtení dat z databáze
- Pohled se HTML šablonou
- CSS soubor se stylem
- JS soubor s kódem Google Maps
- JS soubor s vlastním kódem pro zobrazení teplotních map a uživatelského rozhraní
- Ovladače, který poskládá soubory a data dohromady
- Automatického stahování JS souboru Google Maps v *index.js*

Režie služby pro čtení dat je obdobná, jako ve službě pro vkládání dat, s tím rozdílem, že se výsledek dotazu vrací do ovladače. Samotné dotazy jsou pak ve tvaru:

```
'SELECT lat, lon, noise FROM DataPoints UNION SELECT lat, lon, noise FROM
DataPointsShort'

`SELECT lat, lon, noise FROM DataPoints WHERE dt BETWEEN ${from} AND
${to} UNION SELECT lat, lon, noise FROM DataPointsShort WHERE dt BETWEEN
${from} AND ${to}`

`SELECT lat, lon, noise FROM DataPoints WHERE (dt % 86400000) BETWEEN
${unixTime} AND ${unixTimePlusOne} UNION SELECT lat, lon, noise FROM
DataPointsShort WHERE (dt % 86400000) BETWEEN ${unixTime} AND
${unixTimePlusOne}`
```

Kód 4: Různé dotazy na datové body v serverové databázi

První z dotazů je dotaz na všechna data, kdežto druhý dotaz je pouze na data, jejichž datum a čas pořízení se nachází mezi dvěma daty s časem. Třetí dotaz vybírá pouze data v dané hodině dne, tedy data jejichž čas pořízení se nachází mezi nějakým časem *t* a *t+1 hodina*. To je řešeno pomocí operace modulo provedené na unixovém času v milisekundách. Po provedení této operace s hodnotou 86 400 000, což je počet milisekund jednoho dne, zbude hodnota v rozsahu 0 až 86 399 999. Ta se porovná s časem od uživatele převedeného na unixový čas. Pokud je zadán čas mezi jedenáctou hodinou a půlnocí, tak podmínka se porovnává, zda je hodnota po funkci modulo v rozmezí: zvolený unixový čas až 86 399 999, nebo 0 až zvolený čas plus jedna hodina mínus den. Všechny dotazy jsou sjednocením z dvou poddotazů na obě tabulky s datovými body.

Zatímco v prvním dotazu jsou na začátku i konci dotazu apostrofy značící obyčejný *String*, tak v druhém i třetím případě se jedná o Gravis [32], též tupý akcent, případně jeho anglický alias *backtick*. Tento znak v Javascriptu říká, že se jedná o *template string*, tedy šablonu textu. Do té je pak možné vložit proměnné skrze syntax dolaru a složených závorek kolem názvu proměnné. Tato šablona také podporuje text přes více řádků a nevyhodí výjimku. [33] Proměnné vkládané do dotazů, prošly kontrolou, zda odpovídají regulárnímu výrazu tvaru data a času nebo pouze času, a tedy nemůže nastat situace, že v těchto proměnných by byla nějaká forma útoku (samozřejmě nějaká forma tam být může, ale je odfiltrována už v ovladači).

Následně bylo potřeba utvořit stránku, která se zobrazí uživateli. K tomu jsem využil šablonovací *engine*, nativní pro Node.js, Pug (dříve známý jako Jade). Pro větší aplikace bych využil nějaký JavaScriptový rámec jako Angular, nebo React, ale zde mi to kvůli rozsahu aplikace nepřipadalo nutné. V hlavní složce projektu jsem vytvořil složku *views*, kde jsem přidal první pohled pod jménem *map.pug*. Také jsem vytvořil složku *resources*, kde jsou složky *stylesheets* pro CSS a *scripts* pro skripty. Tato celá šablona vypadá následovně (ochuzena pouze o skripty):

```
doctype html
html
  head
    title Dopravní průzkum ulic – Mapa
    style
      include ../resources/stylesheets/main.css
  body
    #map
    .buttons
      button#heatmap1 Nízké hladiny
      button#heatmap2 Střední hladiny
      button#heatmap3 Vysoké hladiny
      label(for='opacity_input') Průhlednost:
      input#opacity(type='range' value='60' step='10'
name='opacity_input')
      button#gradient1 Základní
      button#gradient2 Pokročilé
```

```
      button#gradient3 Zakládní odlišení červená/zelená
    .datetime_picker
      label(for='dt_from') Data od:
      input#dtfrom(type='datetime-local' name='dt_from' value="2023-03-
01T00:00"
       min="2023-03-01T00:00" max="2023-05-22T23:59")
      label(for='dt_to') Data do:
      input#dtto(type='datetime-local' name='dt_to' value="2023-05-
22T23:59"
       min="2023-03-01T00:00" max="2023-05-22T23:59")
      button#getRangeData Načti data pro daný úsek
      label(for='dt_time') Hodina dne:
      input#timepicker(type='time' name='dt_time' value="12:00")
      button#getTimeData Načti data pro hodinu dne
      button#getMap Načti nefiltrovaná data
    script.
      ...
```

Kód 5: HTML část v Pug šabloně pro zobrazení teplotních map

Za Kód 5 následuje a zde je vynechán můj vlastní Javascript kód, který mimo jiné obsahuje metodu *initMap*, která používá objekty Google (vykreslení teplotních map a funkcí s tím spojených), které se musí nejdřív načíst z jiného souboru, který je uložen na serveru. Poslední řádek šablony je znázorněn v Kód 6.

```
script(src='../resources/scripts/google_maps_script.js'
onload="initMap()")
```

Kód 6: Skript ke stažení skriptu Google Maps ze serveru

Původně bylo zamýšleno, že si uživatel stáhne kód pro vizualizaci Google map sám, ale od roku 2016 (i když ne všechny zdroje s tím už počítají) je nutné použít pro přístup API klíč, který tak musí být obsažen v kódu skriptu. [34] Ten si ovšem může kdokoliv zobrazit při využití funkce prohlížečů zobrazení si zdrojového kódu stránky. Tyto klíče lze omezit, odkud je lze využít (z jaké IP adresy, Android nebo iOS aplikace a další). [35] Já to vyřešil vlastním způsobem a to tak, že jsem kód stáhl a nechal na serveru, odkud si ho stahoval klientský kód. To ovšem fungovalo pouze několik hodin, a poté se místo načtení mapy zobrazila chyba, že kód musí být stahován přímo z Google serverů. To jsem vyřešil tak, že jsem do kódu souboru *index.js* umístil kód, který z Google serveru znovu stahuje kód Google map každé tři hodiny. A umístil jsem ho do souboru *index* právě proto, že je to vstupní bod webové aplikace, který proběhne vždy při spuštění, a proto je to vhodné místo pro umístění kódu spouštěného v pravidelných intervalech.

Jelikož jsem během programování nenašel způsob, jak dané klíče omezit a dělal jsem to po svém. Nyní bych už šel dle nalezené dokumentace a API klíč poslal na uživatelské zařízení a nechal to stáhnout odtamtud. K mé obhajobě, když jsem zveřejnil API klíč prvně, přišlo mi okamžitě upozornění od Google, že je můj klíč veřejně dostupný a obával jsem se případné částky, kdyby někdo klíč zneužil.

Nyní, když je vše připravené stačí to spojit v ovladači zobrazeném v Kód 7. Nejdůležitější části je *res.render*, což je odpověď, aby se ze šablony vytvořilo a poskytlo HTML k vrácení uživateli. *Res* je proměnná obsahující HTML odpověď, první parametr *'map'* je název šablony ve složce *views*. To, že systém *Pug* hledá tento pohled právě v této složce není náhoda, ale je to nadefinováno už při nastavování aplikace v souboru *index.js*. K nastavení stačí pouze dva řádky a ty ukazuje Kód 8.

```
const pug = require('pug');
var utils = require('../utils/writer.js');
var Map = require('../service/MapService');


module.exports.getMap = function getMap(req, res, next) {
  Map.getMap()
    .then(function (response) {
      // render the Pug view with the map data
      res.render('map', { heatmapData: response });
    })
    .catch(function (response) {
      utils.writeJson(res, response);
    });
};
```

Kód 7: Ovladač Map – spojení šablony a dat

```
app.set('views', './views');
app.set('view engine', 'pug');
```

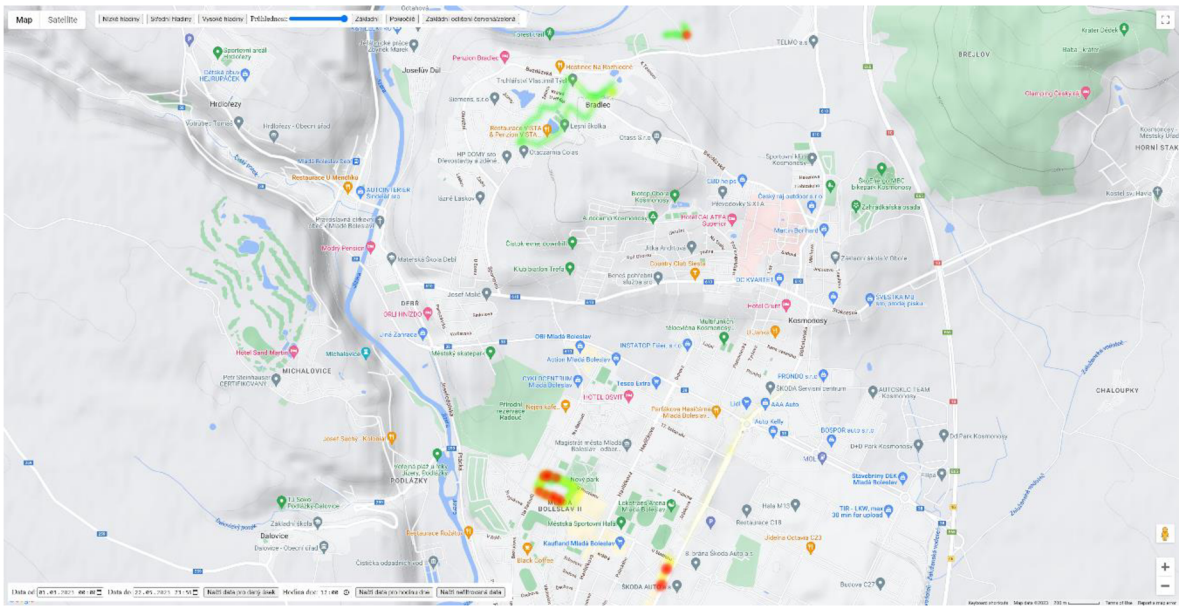
Kód 8: Nastavení šablonovacího enginu a složky s pohledy

Druhým parametrem funkce *render* je objekt obsahující lokální proměnné. Ten je pro tento příkaz volitelný. V tomto případě obsahuje proměnnou *heatmapData*, která bude součástí vrácené odpovědi uživateli. Do té nastaví to, co vrátila mapová služba. Ta vrací pole objektů s třemi vlastnostmi *lat*, *lon*, *noise*, tedy zeměpisná šířka a délka a hluk respektive. Přímo v šabloně je pak vyznačeno (Kód 9), že se budou data vkládat dynamicky z tohoto objektu.

```
const heatmapData = !{JSON.stringify(heatmapData)};
```

Kód 9: Použití obsahu proměnné vložené serverem na klientu

Finální výsledek je na Obrázek 19. V horní části obsahuje několik tlačítek a posuvník, které mu umožňují zobrazit a schovat jednotlivé teplotní mapy (nízké, střední a vysoké hladiny zvuku), nastavit průhlednost teplotních map a také měnit gradient bodů map. Ve spodní části je možnost vyfiltrovat si data od-do nějakého data a času a také zobrazit si data seskupená dle hodiny pořízení.



Obrázek 19: Výstup webové aplikace – prohlížení teplotních map

8.3 Uživatelská část

Tato část obsahuje metody pro zaregistrování uživatele, přihlášení a odhlášení uživatele a také funkci, která slouží jako *middleware* a ověřuje, že uživatel má právo vidět citlivý obsah, jako filtrace času sesbíraných dat. Dále obsahuje šablonu pro stránku s registrací s příslušným JS a CSS. Také bylo nutné rozšířit aplikaci o knihovny *bcrypt*, která „zahashuje“ heslo a také porovnává, zda daný text odpovídá těmto hashům, *jsonwebtoken* (JWT), kterou využívám pro vytvoření tokenu z uživatelského jména, tajného klíče a vypršení platnosti. Tento token je vytvořen a předán zpět uživateli při přihlášení, a *cookies-session*. Ta umožňuje uložení data relace předáním těchto dat na klienta uvnitř cookie.

Nejprve bylo nutné rozšířit serverovou databázi o tabulku, která bude shromažďovat informace o uživatelích. Databázi jsem tedy rozšířil o novou tabulku *Users*, která obsahovala *id* jako *autoincrement* a zároveň primární klíč, dále textová pole pro přezdívku, heslo, jméno, příjmení, email. Dále úroveň oprávnění ve formě celého čísla a na závěr časové razítko pro čas a datum posledního přihlášení.

S touto tabulkou operují dvě metody uživatelské služby, a to registrace a přihlášení uživatele. Na úroveň oprávnění se také doptává *middleware* funkce ověřující JWT tokeny. Ta z tokenu zjistí uživatelovu přezdívku a k té zjistí daný stupeň oprávnění, na jeho základě pak rozhodne, zda má uživatel dostatečná oprávnění nebo nikoliv. Do tabulky nové záznamy vkládá funkce *createUser*, která po obdržení dat o uživateli nejprve zkontroluje, zda už se v tabulce nenachází záznam se stejným uživatelským jménem nebo emailem, v tom případě k registraci nedojde a odešle zpět chybovou odpověď. V opačném případě heslo „zahashuje“ a společně s dalšími informacemi uloží do databáze. Oprávnění je ve výchozím stavu na nule a zatím ho lze editovat pouze ručně v databázi. V budoucnosti by mohla být aplikace rozšířena o administrátorský panel, který by mohl měnit informace o uživatelích, jako právě změna oprávnění. Funkce přihlášení pouze vyčte z tabulky přezdívku a hash hesla, dle uživatelem zadané přezdívky a poté ověří, zda se hashe hesel shodují. Podle úspěchu nebo neúspěchu se liší logika po navrácení do ovladače.

Ovladač uživatelů obsahuje čtyři metody. První pouze vrací zkompilovanou šablonu s dvěma formuláři, a to k přihlášení a registraci. Tento dokument uživatel obdrží při odeslání požadavku GET na cestu */user*. Druhá je metoda odhlášení. Ta pouze nastaví v požadavku serverem (knihovnou *cookies-session*) přidanou vlastnost *session* na *null*. A odešle zpět HTML dokument s přihlašováním a registrací. Metoda pro registraci vrací chybu, pokud se hesla neshodují (při registraci je nutné ho napsat dvakrát), nebo nastane jiná neočekávaná chyba. Pokud ale registrace proběhla úspěšně, je uživatel rovnou přihlášen a přesměrován na dokument s mapou. Poslední metodou je metoda přihlášení, které se dělí na dvě části dle návratové hodnoty z příslušné metody služby. Pokud je návratová hodnota OK, je vytvořen a přidán JWT do uživatelských cookies a uživatel je přesměrován na mapu. V opačném případě uživatel zůstává na dokumentu s přihlašováním a je mu zde vypsána chyba, které při procesu nastala.

9 Testování

Úkolem prvního testování bylo porovnat, jak se liší hlasitost v nějaké tiché a rušné oblasti a také porovnat jaký vliv na hlasitost má způsob, jakým uživatel hodnoty měří. Rušnou oblastí byla ulice 9. května v Mladé Boleslavi ve všední den kolem půl desáté hodiny. Jelikož už bylo po ranní špičce, nebyl provoz tak silný. Tichou oblastí byla zahrada mého domu o přibližně půl hodinu později. Dům se nachází relativně na okraji obce a v blízkosti je les. Počet záznamů jednotlivých měření byl od deseti do dvaceti.

Samsung M21	Ulice města, mírný provoz	Tichá oblast, ptactvo
Telefon v ruce	55,51 dB	38,20 dB
Telefon v kapse bundy	64,66 dB	56,01 dB

Tabulka 3: Porovnání hlasitostí různých oblastí a způsobů měření

Z Tabulka 3 je patrné, že telefon v kapse znatelně přidal na snímané hlasitosti. Nárůst byl znatelnější v tiché oblasti, ale je pozoruhodné, že na ulici města při telefonu v kapse je nárust o devět decibelů oproti měření v ruce. Protože kdyby byly oba zdroje hluku jak kapsy, tak města 55 dB, jejich hlasitost dohromady by byla přibližně 58 dB (nárůst o 3 dB je přibližně dvojnásobná hodnota) a ne 64 dB. Je možné, že zrovna během měření v kapse byl na ulici větší ruch.

Další hypotézou pro ověření bylo, zda všechny telefony měří stejně. Takže jsem nainstaloval aplikaci na druhé zařízení a vyrazil do lesa. Na obou telefonech jsem otevřel aplikaci a začal měřit ve stejný okamžik. Po přibližně pěti minutách jsem měření současně vypnul. Na telefonu Samsung bylo naměřeno o osm záznamů více než na telefonu VIVO, kde záznamů bylo třicet tři. Vyšší počet záznamů při stejném nastavení je pravděpodobně způsoben různou přesností dat z GPS, jelikož cesty se úplně nepřekrývaly (Obrázek 20). Telefon VIVO měl přesnost na tři až čtyři metry i bez mobilních dat (s konfidencí > 95 %), kdežto telefon Samsung M21 měl problém vůbec zaměřit svou polohu bez mobilních dat. Se zapnutými daty s přesnost pohyboval mezi sedmi až deseti metry. Telefon M21 tedy používal A-GPS (*Assisted Global Position System*), kde navíc pomáhá s určením polohy oproti běžnému GPS sítěmi Wi-Fi v okolí, nebo zjišťováním přístupových bodů mobilních dat. [36] [37] [38]

Obrázek 20: Stejná cesta s dvěma telefony

	Samsung M21	VIVO V2050
Telefon v ruce	43,39 dB	38,65 dB

Tabulka 4: Porovnání naměřených hodnot dvou různých telefonů

Na Tabulka 4 lze vidět nezanedbatelný rozdíl v průměrné hlasitosti. Kde telefon Samsung M21 naměřil o skoro pět decibelů více. Pro to se nabízí vysvětlení, že v zatáčce lesní cesty na Obrázek 20, je zvýšená hlasitost až na naměřenou hodnotu nad sedmdesát decibelů. To bylo způsobeno suchým listím po zemi a v něm rychle běžící dvojicí psů, několikrát tam a zpět. A z teplotní mapy (i když pouze ve tvaru dvou cest) je patrné, že telefon Samsung v daný kritický okamžik snímal častěji a vyšší hodnoty. To mohlo zapříčinit zvýšený průměr.

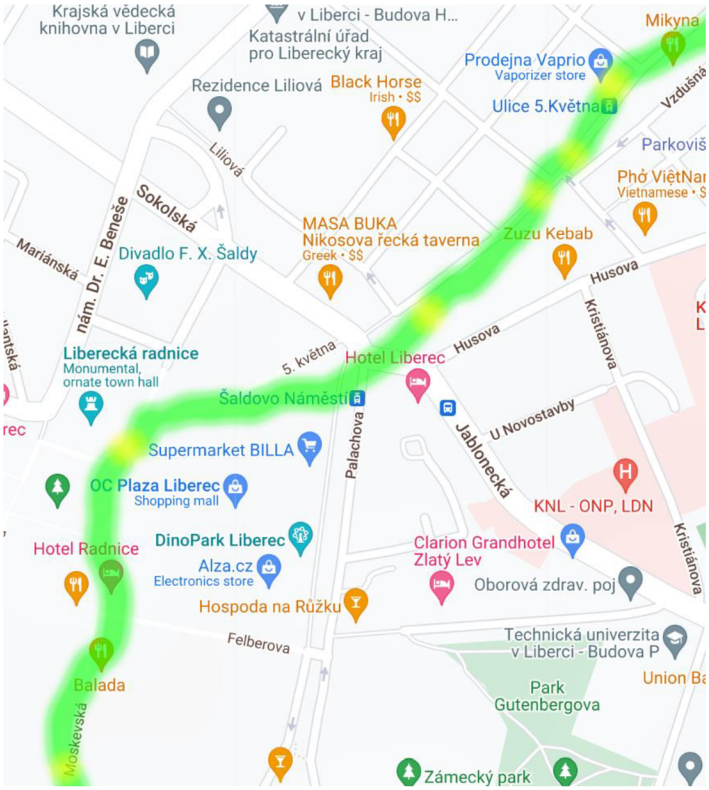
Druhou možností pak je, že každý telefon snímá tlak vzduchu jinak a je nutné je kalibrovat. A to buď navzájem od jiného již kalibrovaného telefonu, nebo za pomoci profesionálních měřičů hladiny zvuku. Další možností kalibrace je měřit hodnotu ticha (například uvnitř domu, v klidné oblasti), nebo hlasitost dopravy projíždějí několik metrů od kalibrovaného zařízení. Za předpokladu, že tiché místnosti a vozidla po světě jsou všude přibližně stejně hlasitá v podobné vzdálenosti, to může posloužit jako dostatečná kalibrační metoda. Kalibrace prozatím není součástí vytvořené aplikace, ale jeví se to jako další vhodná a nutná možnost vývoje. [39]

V pozdější fázi vývoje bylo otestováno znovu na čtyřech úsecích s podobnými podmínkami, vždy na několik minut, jak si odpovídají zařízení Samsung M21 a VIVO V2050 v naměřených hodnotách. Měření bylo započato a ukončeno ve stejný okamžik. Místem byla lesní cesta a každý úsek se měnilo natočení telefonu.

Úseky	A	B	C	D
Samsung M21	37,54 dB	47,12 dB	34,02 dB	35,61 dB
VIVO V2050	31,15 dB	44,53 dB	30,58 dB	34,38 dB

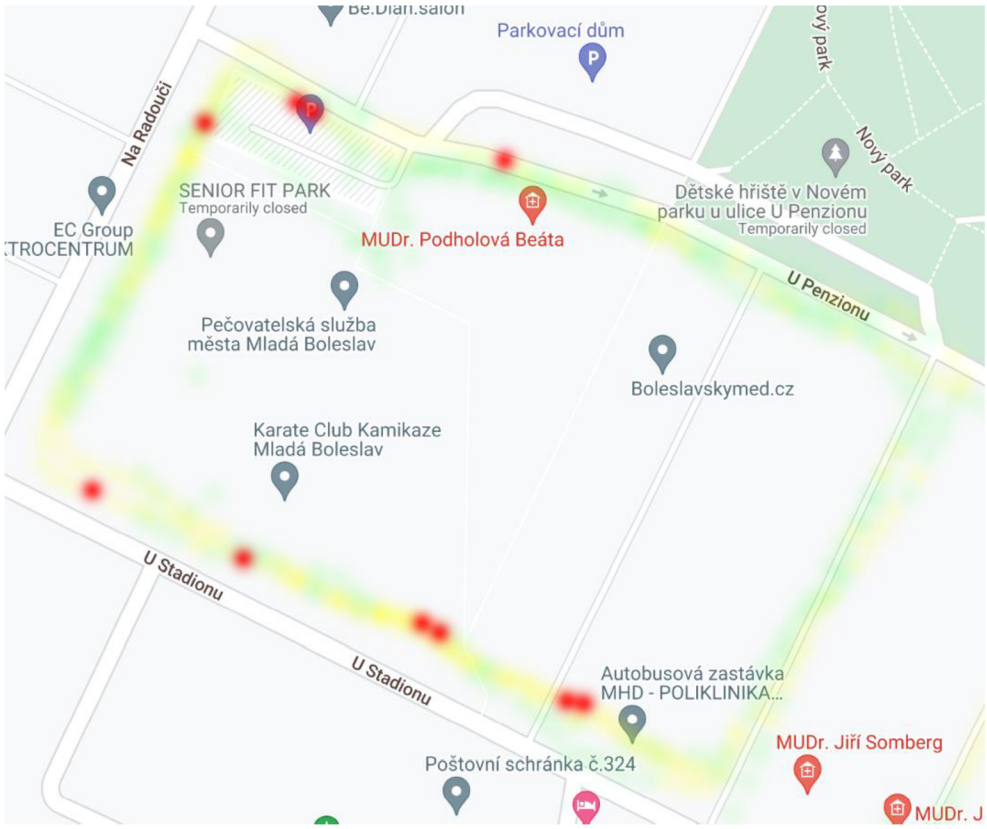
Tabulka 5: Porovnání dvou různých telefonů, 4 úseky

Z Tabulka 5 lze usoudit, že ani opakovaná měření nejsou konzistentní. Jinými slovy rozdíl mezi jednotlivými úseky není nějaké konkrétního čísla, ani hodnota vyjádřená konkrétním poměrem o kolik procent se liší. Z toho usuzuji, že přesná kalibrace bude značně složitá. Z této tabulky a z informace, že jeden z uživatelů naměřil vlastním telefonem při cestě z harcovských kolejí na Fügnerovu ulici a zpět po páté a osmé hodině večerní data s relativně nízkou hodnotou hluku. Z jeho naměřených dat bylo přibližně pouze pět procent ve střední úrovni hlučnosti (50 až 70 dB) a zbytek byl v nízké hladině (pod 50 dB). To neodpovídá mým vlastním měřením, kde cesta vcelku prázdným městem v neděli dopoledne byla tvořena převážně střední hladinou i s občasnými body s vysokou hladinou hluku (nad 70 dB) v momentech, kdy bylo projíždějící vozidlo nejblíže. Část jedné z cest uživatele je na Obrázek 21 a pro porovnání Obrázek 22 s mými měřeními. Přestože uživatel šel přes Šaldovo náměstí, nedošlo k zaznamenání významně vyšších hodnot. Autora měření se mi podařilo dohledat a vypověděl, že data měřil, když držel telefon v ruce, tudíž by tam nemělo být zkreslení hodnot překážkou mezi telefonem a zdroji hluků. To by znamenalo, že jeho zařízení neměří o jednotky decibelů rozdílně, nýbrž o nízké desítky. A v tomto případě by kalibrace (i s odchylkou jednotek decibelů) nabývala mnohem většího významu.



Obrázek 21: Cesta jednoho z uživatelů Libercem – velmi tichá

Další hypotézou bylo alespoň ověřit, když už různé telefony naměří různé hlasitosti, zda aspoň stejné modely naměří stejné nebo přibližné hodnoty. To jsem ověřoval na dvou zapůjčených zařízení Motorola Moto G41. Měření probíhalo v nedělních dopoledních hodinách v Mladé Boleslavi, cestou byla desetiminutová obchůzka dvou bloků mezi ulicemi U Stadionu, Na Radouči a U Penzionu. Telefony naměřily sto dvanáct a sto osm záznamů. Jeden z telefonů jsem omylem na chvilku zamknul, a proto má o čtyři záznamy méně.



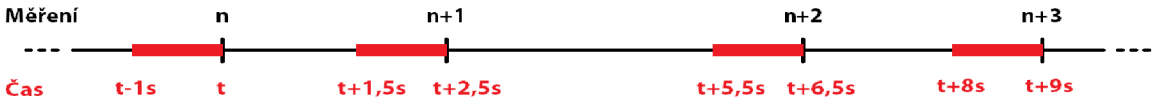
Obrázek 22: Několik cest v Mladé Boleslavi, neděle dopoledne

Na obrázku jsou celkem čtyři cesty, přičemž dvě z nich jsou právě výše zmíněné současná měření. Zároveň je dobré si všimnout, že v ulicích U Stadionu a Na Radouči je obecně větší provoz, než v ulici U Penzionu a dokládají to i data z teplotní mapy. Na mapě je i několik míst červenou barvou a ty prostorově odpovídají místu, kde bylo měřeno, když zrovna okolo projíždělo vozidlo.

Motorola Moto G41	Telefon A	Telefon B
Telefon v ruce	43,21 dB	43,65 dB

Tabulka 6: Porovnání dvou stejných modelů telefonů

Výsledky ukazují, že při použití dvou stejných modelů a měření současně, jsou výsledky téměř totožné. Malé odlišnosti mohou nastat z drobných odlišností v hardware (znečištění otvoru pro mikrofon apod.), nebo z metodiky měření (viz. Obrázek 23). Jelikož měření probíhá každé minimálně dvě a půl vteřiny, nebo při rozdílu minimálně deset metrů. Musí být splněny obě tyto podmínky. A vyrovnávací paměť si ukládá osm tisíc vzorků, které sbírá s frekvencí osm tisíc hertzů, tedy vždy obsahuje poslední vteřinu záznamu, z kterého se poté počítá průměrná hlasitost v daném okamžiku. Tudíž mohou dva stejné telefony brát vždy jakkoliv se překrývající i nepřekrývající se úseky dat.



Obrázek 23: Nákres snímání vyrovnávací paměti měření mikrofonem

10 Další úpravy webové a mobilní aplikace

Jedním ze zajímavých budoucích rozšíření aplikace by mohl být prvek gamifikace pro mapovou aktivitu v reálném čase. Například pokud by některé ulice byly procházeny uživateli méně často, aplikace by mohl zobrazit nějaký objekt na mapě (jeho zjevení by mohlo informovat uživatele vibrací nebo zvukem), který by říkal, když tudy uživatel projde, dostane na své konto body. Ty by se daly dále využít například obdržením slevy do sportovních obchodů, bazénů nebo podobných zařízení. Pokud by se tento prvek naimplementoval, bylo dobré do této aktualizace začlenit i informaci, případně vyskakovací okno, aby uživatel aplikace dbal zvýšené pozornosti a opatrnosti, jelikož se stále pohybuje v blízkosti provozu.

Mobilní aplikace by bylo dobré rozšířit o možnosti kalibrace, jelikož některé modely se skoro v měření neliší a jiné mohou být řádově mimo. Aplikace NoiseModelling z projektu Noise-Planet popsané v kapitole 2.1 nabízí několik možností, jak snížit chybu mikrofonu mobilního zařízení.

Začít měřit až po upřesnění GPS, jelikož první bod po zapnutí měření je skoro vždy nepřesný a občas i druhý ještě není na přijatelné hladině přesnosti. Zároveň pokud přijde událost změny polohy dříve než jednu vteřinu po nastartování aktivity a měření, tak vyrovnávací paměť ještě není naplněna daty a průměrná hlasitost hrubě neodpovídá skutečnosti.

Na mapě přidat funkce, které při podržení umožní uživateli přesunout body. Případně udělat automaticky a body přesouvat na nejbližší ulice, ale zde nastává problém, že někdy (zvláště poblíž vyšších budov) je GPS tak nepřesné, že zaznamenává polohu až na vedlejší ulici. Dalším problémem je, že chodníky nemusí být přímo na okraji vozovky, ale může mezi nimi být pruh zeleně, nebo jiné plochy.

Jakmile bude hodně dat (vyšší desetitisíce datových bodů), bylo by vhodné upravit logiku odesílání dat k vykreslení v teplotních mapách. Například poslat společně u požadavku GET momentální přiblížení na mapě a dle toho zažádat agregovaná data do oblastí při větším oddálení, nebo jednotlivé body k vykreslení teplotní mapy při přiblížení.

Webová aplikace by mohla být rozšířena o administrátorský panel, kde by oprávněná osoba mohla měnit uživatelům oprávnění a další vlastnosti. Záleží, jakým směrem by se další vývoj aplikace pohyboval a jaká nová data by se přidala a bylo nutné měnit.

11 Závěr

Cílem této diplomové práce bylo vytvořit mobilní aplikaci a k ní příslušnou webovou aplikaci pro sběr dat a uživatelů. V teoretické části jsem provedl rešerši již existujících řešení. Uvedl jsem projekt Noise-Planet a další aplikace a projekty, které se zabývají monitoringem hluku a jeho zobrazováním v mapách. Dále jsem se podrobně věnoval platformě Android. Popsal jsem obecnou architekturu tohoto operačního systému, a na jaké verze a rozlišení u různých zařízení je nutné dbát. Poté jsem objasnil, jak aplikace mohou pracovat s daty. Vypsal jsem, jaké senzory mohou zařízení mít a jak s nimi může Android pracovat. Vyjmenoval jsem, z jakých stavebních bloků se vyvíjí aplikace pro tento operační systém. Do jakých formátů mohou být data serializována a jakými formáty se mohou přenášet po síti jsem se věnoval v kapitole týkající se komunikace. Stručně jsem vysvětlil, jaké jsou databáze a technologie pro psaní webových aplikací. Vypsal jsem několik způsobů, jak zajistit bezpečnost mobilní aplikace, webové aplikace a jejich komunikace. Krátce jsem se věnoval objasnění, co je to geografický informační systém.

Prvním krokem praktické části bylo navrhnout formát dat a příslušnou databázi. Použil jsem vestavěnou relační databázi SQLite systému Android. Následně jsem podrobně rozebral implementaci jednotlivých částí Java Android aplikace a webové aplikace Node.js. Součástí popisu implementace byla hlavní aktivita, získání oprávnění od uživatele a poté automatické zaznamenávání dat, odeslání dat na server a dvě aktivity s mapou a cestou uživatele, přičemž pro jednotlivé úseky cesty může uživatel zadávat parametry ulic. Jedna mapová aktivita zobrazuje a umožňuje editovat cestu v reálném čase, druhá zpětně. Naimplementoval jsem bezpečnostní prvky popsané v teoretické části.

Celý systém jsem testoval v terénu. Využil jsem několik zařízení od různých výrobců. Dva rozdílné telefony mohly naměřit relativně podobná data nebo i kompletně odlišná. Stáří, kvalita a stav telefonu mohou hrát roli v přesnosti a citlivosti mikrofonu a GPS. Porovnával jsem vliv překážky mezi mikrofonem a zdrojem hluku, jako je například uložení telefonu v kapse, nebo držení v ruce. Výsledkem byl značný rozdíl, neboť telefon uložený v kapse nebyl utlumen překážkou, ale naopak samotná kapsa, respektive její šustění, bylo zdrojem hluku. Otestoval jsem, zda dva stejné modely mobilních telefonů zaznamenají stejnou hladinu hlasitosti během současného měření.

Dalším vylepšením a novým funkcím jak mobilní, tak webové aplikace je věnována kapitola na konci praktické části. Změny by se mohly týkat především kalibrace mikrofonu, gamifikace mobilní aplikace, případně další vylepšení uživatelského rozhraní. Jedním z návrhů je cesty automaticky připnout k ulicím ze systému GIS. Funkčnost se mi zdá momentálně nereálná, ať už kvůli přesnosti měření – zvlášť ve městech u vysokých budov – nebo kvůli umístění chodníku, kdy nezřídka bývá mezi chodníkem a vozovkou nemalá plocha. Řešením by bylo mít telefony s přesnější GPS a mít GIS, který bude zaměřen na chodníky namísto silnic.

Přínos práce vidím ve vizi samotné neziskové organizace, a to zvýšit pěší mobilitu na krátké vzdálenosti. Na základě naměřených dat je zjištěna hlasitost lokality, tudíž hustota dopravy a tím pádem i znečištění ovzduší emisemi. Zároveň kdyby se aplikace rozšířila o herní prvky, jako například virtuální sběr předmětů za body, mohlo by to mít kladný vliv na přirozený pohyb dětí a mládeže.

Literatura

[1] „Homepage – Pěšky městem,“ Pěšky městem, z.s., [Online]. Available: https://peskymestem.cz/. [Přístup získán 13 květen 2023].

[2] „Mobile Operating System Market Share Worldwide | Statcounter Global Stats,“ StatCounter, [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide. [Přístup získán 23 duben 2023].

[3] „Noise-Planet - NoiseCapture,“ CNRS & Université Gustave Eiffel, [Online]. Available: https://noise-planet.org/noisecapture.html. [Přístup získán 4 květen 2023].

[4] „Noise-Planet - NoiseModelling,“ CNRS & Université Gustave Eiffel, [Online]. Available: https://noise-planet.org/noisemodelling.html. [Přístup získán 4 květen 2023].

[5] „Noise-Planet - Onomap,“ CNRS & Université Gustave Eiffel, [Online]. Available: https://noise-planet.org/onomap.html. [Přístup získán 4 květen 2023].

[6] „Hlukové mapy 2017,“ Zdravotní ústav se sídlem v Ostravě, 12 únor 2019. [Online]. Available: https://www.zuova.cz/Home/Novinka?id=291. [Přístup získán 5 květen 2023].

[7] P. Taylor, „Mobile network subscriptions worldwide 2028 | Statista,“ Statista, 30 březen 2023. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Přístup získán 21 duben 2023].

[8] A. Turner, „How Many People Have Smartphones Worldwide (May 2023),“ BankMyCell, [Online]. Available: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world. [Přístup získán 2023 duben 22].

[9] S. Kemp, „Digital 2023: Global Overview Report — DataReportal – Global Digital Insights,“ Kepios Pte. Ltd., 26 leden 2023. [Online]. Available: https://datareportal.com/reports/digital-2023-global-overview-report. [Přístup získán 22 duben 2023].

[10] J. Howarth, „Internet Traffic from Mobile Devices (May 2023),“ Exploding Topics, 5 květen 2023. [Online]. Available: https://explodingtopics.com/blog/mobile-internet-traffic. [Přístup získán 9 květen 2023].

[11] P. Taylor, „Czechia: smartphone penetration rate 2019-2028 | Statista,“ Statista, 5 květen 2023. [Online]. Available: https://www.statista.com/statistics/568081/predicted-smartphone-user-penetration-rate-in-czech-republic/. [Přístup získán 9 květen 2023].

[12] L. Ceci, „Biggest app stores in the world 2022 | Statista,“ Statista, 8 listopad 2022. [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. [Přístup získán 2023 duben 23].

[13] „Application Sandbox | Android Open Source Project,“ Google LLC, [Online]. Available: https://source.android.com/docs/security/app-sandbox. [Přístup získán 30 duben 2023].

[14] „Platform architecture | Android Developers,“ Google LLC, 4 květen 2023. [Online]. Available: https://developer.android.com/guide/platform. [Přístup získán 2023 květen 15].

[15] „Android Version Market Share Worldwide | Statcounter Global Stats,“ Stat Counter, [Online]. Available: https://gs.statcounter.com/os-version-market-share/android. [Přístup získán 23 duben 2023].

[16] „Mobile Screen Resolution Stats Worldwide | Statcounter Global Stats,“ StatCounter, [Online]. Available: https://gs.statcounter.com/screen-resolution-stats/mobile/worldwide. [Přístup získán 23 duben 2023].

[17] „Sensors Overview | Android Developers,“ Google LLC, [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview. [Přístup získán 2022 duben 24].

[18] „Introduction to activities | Android developers,“ Google LLC, [Online]. Available: https://developer.android.com/guide/components/activities/intro-activities. [Přístup získán 7 květen 2023].

[19] „Layout | Android developers,“ Google LLC, [Online]. Available: https://developer.android.com/develop/ui/views/layout/declaring-layout. [Přístup získán 7 květen 2023].

[20] „Intents | Android developers,“ Google LLC, [Online]. Available: https://developer.android.com/reference/android/content/Intent. [Přístup získán 7 květen 2023].

[21] „Broadcasts overview | Android developers,“ Google LLC, [Online]. Available: https://developer.android.com/guide/components/broadcasts. [Přístup získán 7 květen 2023].

[22] „Intent | Android Developers,“ Google LLC, [Online]. Available: https://developer.android.com/reference/android/content/Intent. [Přístup získán 7 květen 2023].

[23] „Services overview | Android Developers,“ Google LLC, [Online]. Available: https://developer.android.com/guide/components/services. [Přístup získán 7 květen 2023].

[24] „Fragments | Android Developers,“ Google LLC, [Online]. Available: https://developer.android.com/guide/fragments. [Přístup získán 7 květen 2023].

[25] „App security best practices | Android Developers,“ Google LLC, [Online]. Available: https://developer.android.com/topic/security/best-practices. [Přístup získán 14 květen 2023].

[26] J. Evers, „GIS (Geographic Information System),“ Natural Geographic Society, 19 prosinec 2022. [Online]. Available: https://education.nationalgeographic.org/resource/geographic-information-system-gis/. [Přístup získán 14 květen 2023].

[27] P. Cenek, „GIS Otrava 2000 - Cenek, P.: Použití lineárního referenčního systému pro popis dopravních sítí,“ 2000. [Online]. Available: http://gisak.vsb.cz/GIS_Ostrava/GIS_Ova_2000/Sbornik/Cenek/Referat.htm. [Přístup získán 14 květen 2023].

[28] „Permissions updates in Android 11 | Android Developers,“ Google LLC, [Online]. Available: https://developer.android.com/about/versions/11/privacy/permissions. [Přístup získán 25 duben 2023].

[29] „Location | Android Developers,“ Google LLC, [Online]. Available: https://developer.android.com/reference/android/location/Location. [Přístup získán 25 duben 2023].

[30] „Decimal degrees - GIS Wiki | The GIS Encyclopedia,“ wiki.GIS.com, 24 květen 2011. [Online]. Available: http://wiki.gis.com/wiki/index.php/Decimal_degrees. [Přístup získán 25 duben 2023].

[31] Anonymní, „Unpacking Android backups,“ 12 červen 2012. [Online]. Available: https://nelenkov.blogspot.com/2012/06/unpacking-android-backups.html. [Přístup získán 29 duben 2023].

[32] „Gravis – Wikipedie,“ Wikipedie, 28 prosinec 2021. [Online]. Available: https://cs.wikipedia.org/wiki/Gravis. [Přístup získán 25 duben 2023].

[33] „Template literals (Template strings) - JavaScript | MDN,“ Mozilla Foundation, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals. [Přístup získán 26 duben 2023].

[34] „Building for Scale: Updates to Google Maps APIs Standard Plan | Google Cloud Blog,“ Google LLC, 2016 červen 2016. [Online]. Available: https://cloud.google.com/blog/products/maps-platform/building-for-scale-updates-to-google. [Přístup získán 26 duben 2023].

[35] „Authenticate using API keys | Authentication | Google Cloud,“ Google LLC, [Online]. Available: https://cloud.google.com/docs/authentication/api-keys?visit_id=638184160056141018-194174466#api_key_restrictions. [Přístup získán 26 duben 2023].

[36] P. A. Zandbergen a S. Barbeau, „Positional Accuracy of Assisted GPS Data from High-Sensitivity GPS-enabled Mobile Phones,“ červenec 2011. [Online]. Available: https://www.researchgate.net/publication/231849997_Positional_Accuracy_of_Assisted_GPS_Data_from_High-Sensitivity_GPS-enabled_Mobile_Phones. [Přístup získán 2023 duben 28].

[37] P. A. Zandbergen, „Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning,“ 2009. [Online]. Available: https://www.globe.gov/documents/2631933/6965868/42534991.pdf. [Přístup získán 28 duben 2023].

[38] F. Zahradnik, „How Assisted GPS Works in Cellphones,“ Dotdash Meredith, 6 září 2021. [Online]. Available: https://www.lifewire.com/assisted-gps-1683306. [Přístup získán 28 duben 2023].

[39] „Noise-Planet - NoiseCapture calibration,“ CNRS & Université Gustave Eiffel, [Online]. Available: https://noise-planet.org/noisecapture_calibration.html. [Přístup získán 25 duben 2023].