FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Mobilní aplikace pro vzájemnou komunikaci mezi vozidly pomocí 5G
Student:	Bohdan Poberezhnyi
Vedoucí:	RNDr. Zuzana Kosová
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Hlavním cílem této práce je vytvoření bezpečnostního nástroje pro vozidla na silnicích, který předem upozorní na přibližující se dopravní prostředky (například vozidla IZS). V rámci práce budou vytvořeny dvě mobilní aplikace pro komunikaci mezi vozidly pomocí 5G. Tyto aplikace budou určeny k využití na mobilních zařízeních s operačním systémem Android. První aplikace bude sloužit k signalizování ostatním mobilním zařízením v okolí, že jsou blízko nich. Druhá aplikace bude zobrazovat notifikace o poloze blížících se aktivních zařízení. Tím se předejde situacím, kdy například řidič neslyší přijíždějící vozidla IZS. Řešení této problematiky může mít významný dopad na bezpečnost v dopravě.

Postupujte v následujících krocích:

1. Specifikujte funkční a nefunkční požadavky na aplikace.
2. Proveďte rešerši podobných řešení.
3. Vyberte vhodné technologie a navrhněte aplikace.
4. Implementujte prototypy obou aplikací.
5. Prototypy řádně otestujte a zdokumentujte.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 18. listopadu 2023 v Praze.

Bakalářská práce

MOBILNÍ APLIKACE PRO VZÁJEMNOU KOMUNIKACI MEZI VOZIDLY POMOCÍ 5G

Bohdan Poberezhnyi

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: RNDr. Zuzana Kosová
16. května 2024

Obsah

Seznam obrázků

Seznam výpisů kódu

Chtěl bych poděkovat především své vedoucí práce RNDr. Zuzaně Kosové za vedení a pomoc při tvorbě bakalářské práce. Děkuji také svým rodičům a bratrovi za podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 16. května 2024

Abstrakt

Tato bakalářská práce se věnuje využití mobilních aplikací k zlepšení bezpečnosti v dopravě. Zaměřuje se na identifikaci blížících se dopravních prostředků, zejména vozidel integrovaného záchranného systému, a vytváří bezpečnostní nástroj pro vozidla, který umožňuje předem upozornit na přibližující se dopravní prostředky.

V rámci práce probíhá analýza existujících aplikací s podobným zaměřením, návrh vlastního řešení, jeho implementace a testování. Výsledkem práce jsou dva prototypy mobilních aplikací a backend. První aplikace slouží k signalizování přítomnosti vozidla a informování o jeho poloze, zatímco druhá aplikace zobrazuje notifikace o blížících se vozidlech.

Řešení tohoto problému je určené především pro hendikepované uživatele, jako jsou lidé se sluchovými problémy, a může přinést značný přínos pro bezpečnost silničního provozu.

Klíčová slova mobilní aplikace, Android, 5G, integrovaný záchranný systém, dopravní bezpečnost, Kotlin, Ktor

Abstract

This bachelor thesis focuses on the use of mobile applications to improve traffic safety. It targets the identification of approaching vehicles, particularly those of the integrated rescue system, and develops a safety tool that can provide early warnings about approaching traffic.

The thesis includes an analysis of existing applications with similar objectives, the design of a solution, its implementation, and testing. The result of the thesis is two mobile app prototypes and a backend. The purpose of the first app is to signal the presence of a vehicle and inform about its location, while the second app displays notifications about approaching vehicles.

The solution is primarily intended for users with disabilities, such as those with hearing loss, and can significantly contribute to traffic safety.

Keywords mobile applications, Android, 5G, integrated rescue system, traffic safety, Kotlin, Ktor

Seznam zkratek

API	Application Programming Interface
CRUD	Create Read Update Delete
ER	Entity Relationship
DAO	Data Access Object
DSRC	Dedicated Short Range Communications
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IZS	Integrovaný záchranný systém
JSON	JavaScript Object Notation
JWT	JSON Web Token
MIMO	Multiple-Input and Multiple-Output
MVC	Model View Controller
MVP	Model View Presenter
MVVM	Model View ViewModel
ORM	Object Relational Mapping
REST	Representational State Transfer
SDK	Software Development Kit
SQL	Structured Query Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Uniform Resource Locator

Úvod

V současné době díky technologickému pokroku a rychlému internetovému připojení, jako je například technologie 5G, se otevírají nové možnosti pro využití mobilních aplikací s cílem zlepšit bezpečnost silničního provozu.

Pro uživatele s poruchami sluchu může být identifikace blížících se dopravních prostředků, zejména vozidel integrovaného záchranného systému (IZS), náročná. Cílem této bakalářské práce je proto vytvoření bezpečnostního nástroje pro vozidla, který umožní předem upozornit na přibližující se dopravní prostředky. Pro dosažení tohoto cíle budou v rámci práce vyvinuty dvě mobilní aplikace.

První aplikace bude sloužit k signalizování ostatním vozidlům v okolí přítomnost vozidla a informování o jeho poloze. Druhá aplikace pak bude zobrazovat notifikace o blížících se vozidlech, což pomůže řidičům lépe reagovat na příchozí dopravní situace, včetně případů, kdy je třeba dát přednost vozidlům IZS. Obě tyto aplikace budou vyvíjeny pro mobilní zařízení s operačním systémem Android.

Tento úvodní text dává pouze obecný nástin tématu a cílů práce. V následujících kapitolách budou detailněji probírány funkční a nefunkční požadavky na aplikace, provedena rešerše podobných řešení, vybrány vhodné technologie a provedena implementace a testování prototypů aplikací.

Kapitola 1
Analýza

Tato kapitola slouží k provedení potřebné analýzy pro další návrh a implementaci aplikací. Za tímto účelem je nutné provést rešerši podobných řešení a jejich analýzu. Poté by bylo snazší specifikovat požadavky na aplikace a vybrat, jaké technologie budou potřebné pro jejich implementaci.

1.1 Analýza podobných řešení

Tato sekce se zabývá rešerší a analýzou podobných řešení.

Hledání aplikací je prováděno s ohledem na to, že tyto aplikace by měly být schopny zobrazovat na mapě různá rizika a nebezpečí, například uzavřené silnice, příjezd záchranných vozidel nebo nehodu.

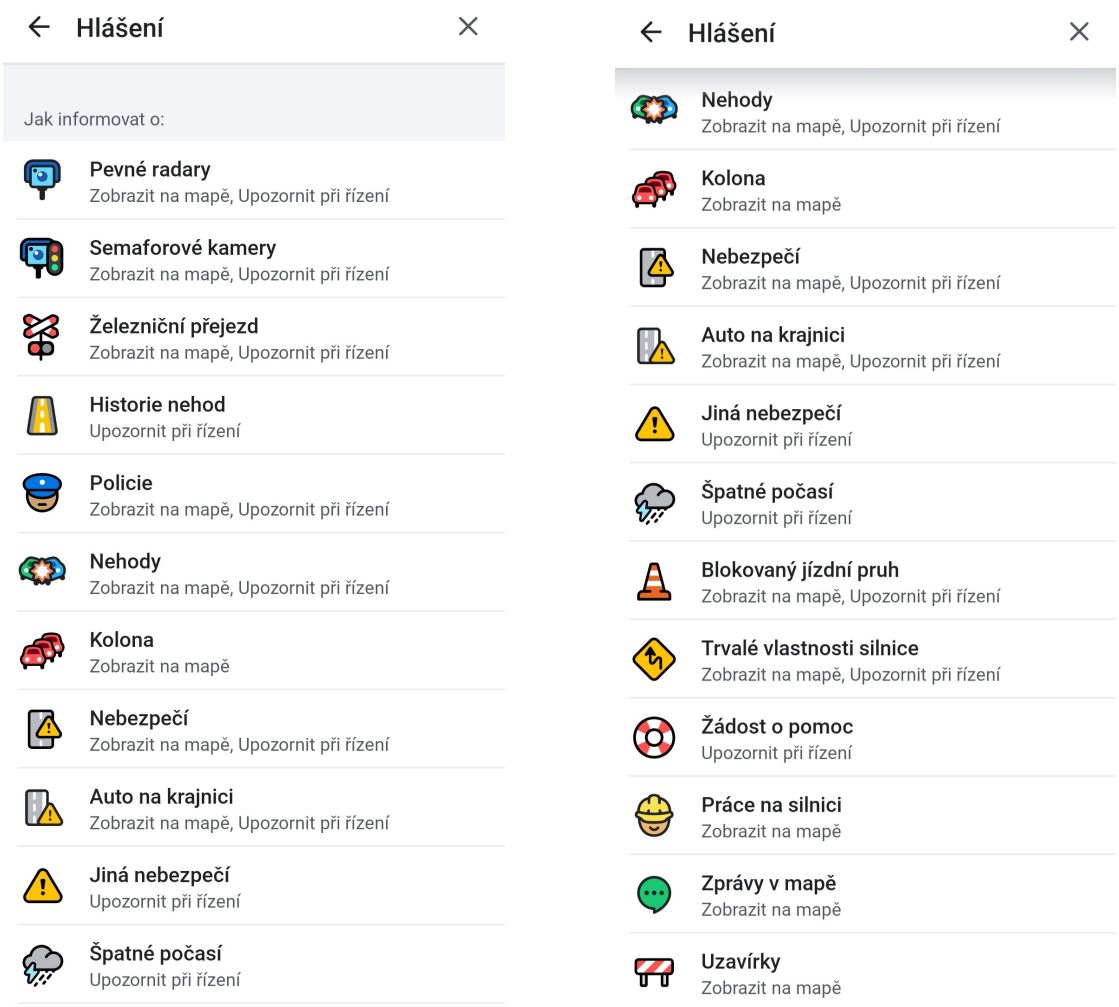
Preferována jsou taková řešení, která umožňují ostatním uživatelům na silnici v reálném čase zjistit polohu záchranných a hasičských služeb, což jim umožní předvídat a reagovat na jejich přítomnost a pohyb na silnici.

1.1.1 Waze

Aplikace Waze je mobilní navigační aplikace, která poskytuje uživatelům informace o dopravní situaci v reálném čase, navigaci, upozornění na dopravní nehody, zácpy a jiné události na silnicích [1]. Hlavním cílem aplikace je umožnit uživatelům rychlejší a efektivnější cestování pomocí nejaktuálnějších informací o provozu na silnicích.

1.1.1.1 Hlavní funkce

- Navigace s živými aktualizacemi: Aplikace poskytuje živé aktualizace dopravní situace, což umožňuje uživatelům plánovat svou trasu na základě aktuálního provozu.
- Sdílení informací: Uživatelé mohou sdílet informace o dopravní situaci, nehodách, zácpách a jiných událostech na silnicích s ostatními uživateli. Na obrázku 1.1 je vidět, které události jsou k dispozici pro hlášení.
- Alternativní trasy: Aplikace nabízí alternativní trasy v případě, že na hlavní trase se prodlouží čas jízdy kvůli neočekávané události.

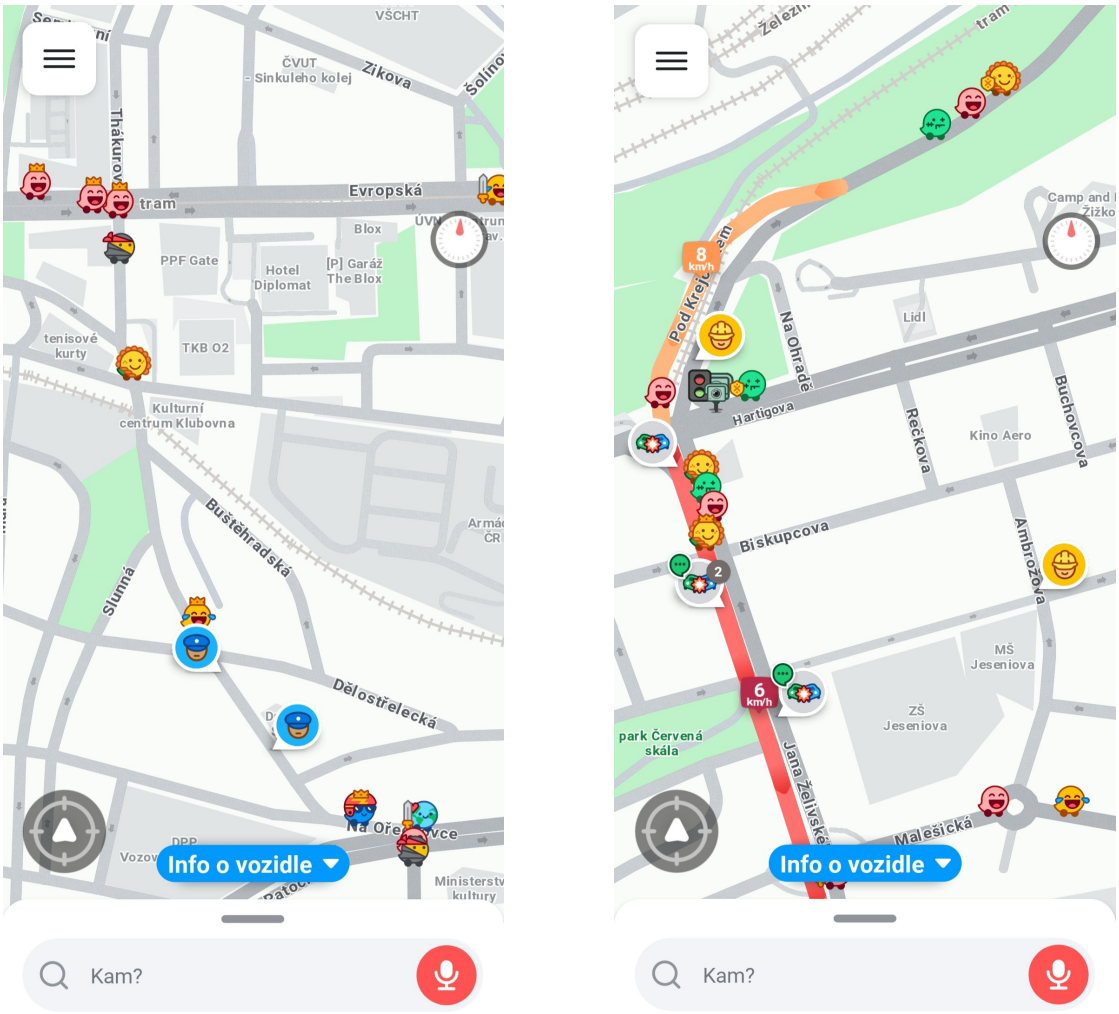


Obrázek 1.1 Hlášení událostí v aplikaci Waze [1]

1.1.1.2 Výhody a nevýhody

- **Výhody:** Aplikace má poměrně hezké, pohodlné a intuitivní uživatelské rozhraní. Informace přichází v reálném čase. Každý účastník silničního provozu má možnost nahlásit událost, například nehodu na silnici. Aplikaci lze stáhnout zdarma z Google Play [2].

- **Nevýhody:** Informace o událostech mohou být nepřesné, protože každý účastník může nahlásit problém, ale není zaručeno že poskytne správné a přesné informace. Navíc nebylo zjištěno sledování vozidel IZS v reálném čase. Lze nahlásit nehodu na silnici a oznamovatel může uvést, že na místě je policie, což vám umožní předem na situaci reagovat, ale nedochází k sledování těchto vozidel na cestě k místu události. Také na místech, kudy neprojíždí auta, nejsou k dispozici žádné informace.



Obrázek 1.2 Mapa v aplikaci Waze [1]

1.1.1.3 Shrnutí

Tato aplikace je efektivní při varování před statickými událostmi na silnici, jako jsou nehody či jiná nebezpečí, která se nacházejí na určitém místě. Nicméně není vhodné pro sledování událostí, které jsou v pohybu. Také zde je dobrý přístup, kdy uživatelé sami aktivně přispívají k bezpečnosti na silnicích tím, že mají možnost nahlásit nebezpečí, ale to je také i nevýhoda, protože informace v takovém případě může být nepřesná nebo nepravdivá.

1.1.2 HAAS Alert

HAAS Alert je technologická společnost specializující se na vývoj a implementaci systémů včasného varování před blížícími se nebezpečnými situacemi na silnicích [3]. Nabízejí řešení pro vozidla záchranných služeb a další dopravní prostředky, která umožňují odesílat v reálném čase upozornění o své poloze ostatním účastníkům provozu [4]. To

umožňuje ostatním řidičům snížit riziko vzniku nehody.

1.1.2.1 Hlavní funkce

- Sledování pohybujících se vozidel: Zajišťuje sběr informací o poloze a stavu varování všech vozidel vybavených tímto systémem. Analyzují se historická data, aby se získaly statistiky incidentů a zlepšila se bezpečnost
- Snížení rizika pro záchranáře: Snižuje nebezpečí srážek s vozidly tím, že informuje řidiče o záchranných vozidlech nebo místech nouze.
- Poskytování hardwaru: Tato společnost poskytuje kromě softwarového vybavení také určitý hardware, jako jsou transpondéry a specializované sady světel pro správný provoz jejich služby.

1.1.2.2 Výhody a nevýhody

- **Výhody:** Dobré řešení této problematiky. Sleduje určitá dopravní vozidla a předává informace do cloudu.
- **Nevýhody:** Tato služba je placená, cena se počítá za každé sledované dopravní vozidlo. Navíc vyžaduje instalaci specializovaného hardwaru.

1.1.2.3 Shrnutí

Tato služba nabízí dobré řešení této problematiky, ale hlavní nevýhodou je, že vyžaduje specializovaný hardware pro sledovaná vozidla. Tato potřeba však může být obejita použitím komunikace přes síť 5G, protože hlavní problémy spočívají v spolehlivosti, rychlosti přenosu a stabilitě přenosu dat.

1.1.3 Glance TravelSafely

Společnost Applied Information se specializuje na zlepšování bezpečnosti ve veřejné dopravě, zaměřuje se na optimalizaci dopravních křižovatek, parkovacích systémů, integrovaných záchranných služeb a řízení semaforů [5]. Poskytují aplikaci Glance TravelSafely, kterou lze zdarma stáhnout z Google Play [6]. Je určená nejen pro vozidla, ale také pro cyklisty a chodce.

Pro pochopení principu fungování celého systému je nutné provést analýzu jejich dvou produktu „Connected Vehicle Solutions“ [7] a „Emergency Vehicle & Preemption System“ [8], které zaručují komunikaci pomocí sítě 4G a DSRC[1].

1.1.3.1 Hlavní funkce

- Samotné produkty „Connected Vehicle Solutions“ a „Emergency Vehicle & Preemption System“ zahrnují instalace speciálního zařízení pro příjem signálu a jeho

[1]Dedicated short-range communications (DSRC) je technologie pro přímou bezdrátovou výměnu dat mezi vozidly a ostatními účastníky silničního provozu (např. chodci, cyklisté) a silniční infrastrukturou (např. dopravní signály, elektronické informační tabule).

přesměrování do cloudu. Nejčastěji tato zařízení jsou umístěny na semaforech, které jsou rozmístěny na křižovatkách a dalších rizikových místech. Kromě semaforů mohou být tato zařízení instalována i na dalších částech dopravní infrastruktury, jako dopravní značky nebo informační tabule.

- Mobilní aplikace Glance TravelSafely: Tato mobilní aplikace slouží k bezpečné navigaci účastníků silničního provozu. Uživatelé aplikace jsou aktivně informováni o přiblížení a pohybu vozidel, kterým by měli například dát přednost na silnici.
- Zobrazování informační zpráv ve formě textu a šipek: Informace o dopravních událostech a situacích jsou uživatelům prezentovány ve formě textových zpráv a vizuálních šipek. Tímto způsobem je uživatelům umožněno reagovat na aktuální situace na silnici a minimalizovat riziko dopravních nehod.

1.1.3.2 Výhody a nevýhody

- **Výhody:** Poskytuje komplexní řešení pro zlepšení bezpečnosti ve veřejné dopravě. Zaměřuje se nejen na vozidla, ale také na bezpečnost cyklistů a chodců. Také společnost informuje, že brzy jejich hardware bude schopen komunikovat nejen v síti 4G, ale také v síti 5G.
- **Nevýhody:** Zobrazení nebezpečí není příliš pohodlné, protože chybí mapa, na které by bylo možné přesněji vidět polohu ostatních dopravních prostředků. Připojení zařízení pouze na určitých úsecích silnice s omezeným pokrytím rovněž není optimálním řešením, protože je žádoucí mít pokrytí na celé silnici, což vyžaduje značné investice do zařízení.

1.1.3.3 Shrnutí

I když společnost Applied Information přináší inovativní přístup k zlepšení bezpečnosti ve veřejné dopravě prostřednictvím svých produktů, existují určité nedostatky, které by měly být zohledněny. Jedním z hlavních problémů je omezená vizualizace nebezpečí pomocí mobilní aplikace Glance TravelSafely, která ztěžuje identifikaci a reakci na potenciální rizika. Dalším problémem je potřeba rozsáhlého pokrytí zařízení, což představuje značné náklady.

1.1.4 Závěr

Během analýzy byly prozkoumány různé aplikace zaměřené na zlepšení bezpečnosti ve veřejné dopravě. Každá z těchto aplikací má své vlastní výhody a nevýhody, avšak žádná z nich nenabízí ideální řešení. Některé z nich neumožňují sledovat pohyb vozidel v reálném čase, což může být způsobeno omezenými možnostmi mobilního připojení pomocí sítě 4G, která nemá dostatečnou rychlost a spolehlivost pro tento účel. Jiné aplikace řeší tuto problematiku pomocí speciálního hardwaru, což může být neoptimální řešení z důvodu nákladů a složitosti s vývojem takového zařízení. Proto v rámci této práce předpokládáme, že řidiči budou využívat mobilní telefony, které komunikují v síti 5G, což bude podrobněji popsáno v následující sekci „5G“ 1.2. Specifikace požadavků na tyto mobilní aplikace bude dále popsána v sekci „Specifikace požadavků“ 1.3.

1.2 5G

5G (pátá generace bezdrátových systémů) představuje nový telekomunikační standard mobilní sítě, který navazuje na standard 4G. Hlavním přínosem této nové technologie je významné zvýšení přenosové rychlosti a podstatné snížení doby odezvy oproti standardu 4G.

1.2.1 Výhody 5G

Nový standard má umožnit zvýšení teoretické přenosové rychlosti až na 20 Gbit/s a snížení odezvy na jednotky milisekund. To otevírá možnosti pro rychlejší internet v mobilních telefonech, což umožní rychle a efektivně přenášet data mezi mobilními zařízeními a serverem oběma směry, a mělo by být dostatečné pro rychlost komunikace, která bude potřebná v rámci této práce.

Technické řešení 5G je založeno na několika klíčových technologiích, jako jsou milimetrové vlny, malé buňky, massive MIMO a plně duplexní komunikace. Tyto technologie umožňují efektivnější využití frekvenčního pásma, zvýšení hustoty sítě a zlepšení rychlosti a spolehlivosti přenosu dat [9].

- Výhody 5G zahrnují:
  - Vyšší kapacitu a datovou rychlost: 5G nabízí vyšší kapacitu a datovou rychlost, což umožňuje rychlejší přenos dat a lepší výkon pro uživatele.
  - Nižší latence: Díky nižší latenci umožňuje 5G rychlejší odezvu a zlepšuje výkon aplikací vyžadujících rychlou reakci.
  - Masivní propojení zařízení: 5G umožňuje propojení velkého množství zařízení.
  - Nižší náklady: 5G může snížit náklady na provoz a údržbu díky efektivnějšímu využití zdrojů a možnosti automatizace.
  - Konzistentní kvalitu služeb: Díky lepšímu řízení sítě a větší kapacitě může 5G poskytnout konzistentní kvalitu služeb pro uživatele, což je důležité pro aplikace vyžadující spolehlivé připojení.

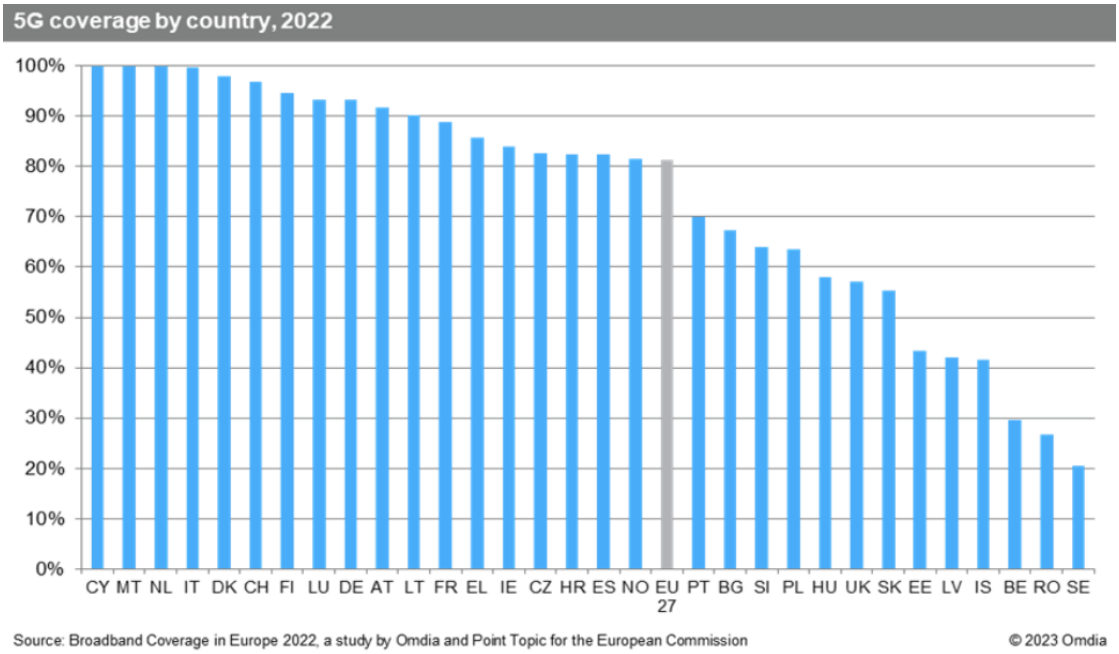
1.2.2 Pokrytí 5G sítí v České republice

Předpokládá se, že řidiči budou využívat mobilní telefony, které komunikují v síti 5G 1.1.4, ale k tomu je nutné znát pokrytí sítě 5G v České republice.

V roce 2022 byl proveden průzkum pokrytí širokopásmovým připojením v Evropě, jehož cílem bylo monitorovat pokrok členských států EU směrem k pokrytí gigabitovým připojením a sítěmi 5G [10]. Z jejich publikace vyplývá, že na konci června 2022 bylo 82,6 % území České republiky pokryto sítí 5G:

„Dostupnost LTE služeb je univerzální po celé České republice a více než osm z deseti domácností (82,6 %) je pokryto sítěmi 5G po zaznamenání nárůstu o 33,2 procentního bodu oproti předchozímu roku.“ [10].

Také na diagramu 1.3 je vidět, že podle tohoto ukazatele je Česká republika nad evropským průměrem. Z těchto informací můžeme usoudit, že Česká republika má poměrně dobrou dostupnost sítě 5G.



Obrázek 1.3 Pokrytí 5G podle státu EU, 2022 [10]

.

1.3 Specifikace požadavků

V rámci bakalářské práce budou vyvinuté dvě mobilní aplikace, LocShare a LocTracker. LocShare bude sloužit k sdílení informací o poloze vozidel IZS, jako jsou hasičská, záchranná a policejní vozidla apod. LocTracker, jakožto druhá aplikace, bude sloužit k přijímání informací o poloze těchto vozidel, která jsou vysílána pomocí aplikace LocShare. Bude využíváno uživateli, kteří chtějí být informováni o blízkosti a pohybu vozidel IZS.

Dále budou podrobně rozebrány funkční a nefunkční požadavky na obě aplikace. Funkční požadavky jsou specifikace toho, co systém má dělat. Tedy jaké funkce, úkoly nebo operace by měl systém provádět, aby splňoval požadavky uživatelů. Nefunkční požadavky se naopak zaměřují na vlastnosti systému, které nejsou přímo spojeny s jeho funkcemi. Tyto požadavky popisují charakteristiky systému, jeho obecné vlastnosti a chování, bezpečnost, spolehlivost.

1.3.1 LocShare

1.3.1.1 Funkční požadavky

- **F1: Registrace.**

  Aplikace obsahuje jednoduchou registraci pro další identifikaci uživatele.

- **F2: Přihlášení.**

  Uživatel se přihlásí pomocí uživatelského jména a hesla.

- **F3: Odhlášení.**

  Uživatel se odhlásí a bude mít možnost změnit uživatelský účet.

- **F4: Zobrazení mapy.**

  Uživatel vidí mapu a svou polohu v reálném čase.

- **F5: Odesílání polohy.**

  Uživatel má možnost aktivovat a deaktivovat odesílání své vlastní polohy pomocí tlačítka.

- **F6: Výběr typu vozidla.**

  Uživatel má možnost vybrat typ vozidla a vložit poznámky do textového pole.

- **F7: Připojení k síti 5G.**

  Aplikace zobrazuje, kdy je telefon připojený k síti 5G.

1.3.1.2 Nefunkční požadavky

- N1: Aplikace je kompatibilní s telefony, které mají minimální verzi operačního systému Android 8.1 a podporují připojení k síti 5G.
- N2: Zajištěná bezpečnost dat uživatelů, zejména ochrana hesel.
- N3: Není možné sdílet polohu z jednoho uživatelského účtu a zároveň z různých zařízení.
- N4: Není možné sdílet polohu uživatele bez jeho souhlasu.

1.3.2 LocTracker

1.3.2.1 Funkční požadavky

- **F8: Zobrazení mapy.**

  Uživatel vidí mapu a svou polohu v reálném čase.

- **F9: Připojení k síti 5G.**

  Aplikace zobrazuje, kdy je telefon připojený k síti 5G

- **F10: Zobrazení nebezpečí.**

  Aplikace zobrazuje polohu vozidel, kterou sdílí aplikace LocShare.

1.3.2.2 Nefunkční požadavky

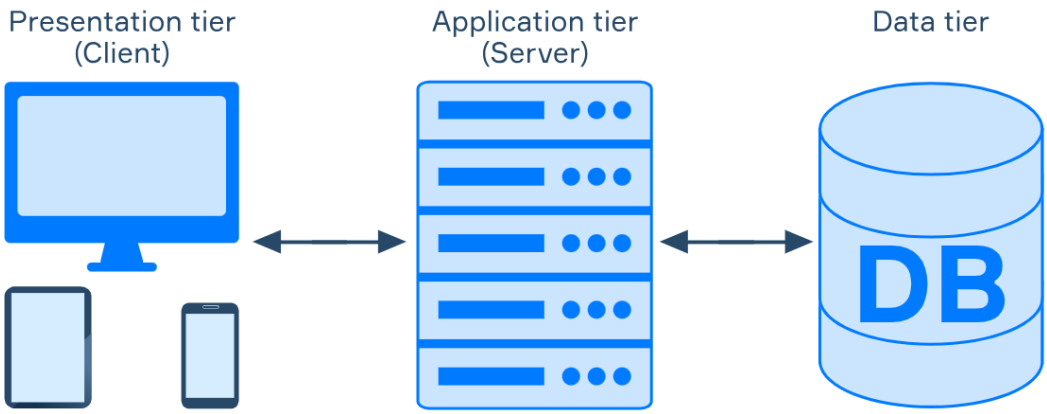
- N5: Aplikace je kompatibilní s telefony, které mají minimální verzi operačního systému Android 8.1 a podporují připojení k síti 5G.
- N6: Není možné sdílet polohu uživatele bez jeho souhlasu.
- F7: Aplikace je k dispozici k použití bez nutnosti registrace a přihlášení do uživatelského účtu.

Kapitola 2
Návrh

V této kapitole jsou představeny hlavní aspekty návrhu vyvíjeného řešení. Je zde popsána architektura systému a hlavní technologická řešení, která jsou použita k dosažení stanovených cílů.

2.1 Architektura

Na základě požadavků a představy o fungování softwarového systému jako celku byla vybrána pro tuto práci, s ohledem na efektivní organizaci procesu zpracování a ukládání dat, třívrstvová architektura z několika důvodů. Umožňuje efektivní oddělení složitých business logik a funkcí aplikace, což činí systém více modulárním a usnadňuje jeho správu. Tímto přístupem se zajišťuje lepší škálovatelnost, flexibilita, bezpečnost a jednoduchá modifikace jednotlivých komponent systému, což urychluje proces tvorby nových funkcí, testování a usnadňuje údržbu systému.



Obrázek 2.1 Třívrstvá architektura [11]

- Vrstvy architektury:
  - **Prezentační vrstva:** Tato část aplikace je viditelná pro uživatele a zajišťuje přijímání vstupních požadavků a prezentaci výsledků. Je specifická pro každou platformu (např. webová aplikace, aplikace pro Android atd.) a může se lišit podle zařízení či platformy.

    Pro tento systém jsou prezentační vrstvou dvě mobilní aplikace, které poskytují uživatelům rozhraní pro interakci s aplikací.
  - **Aplikační vrstva:** Střední vrstva modelu, která zajišťuje výpočty a operace mezi vstupně-výstupními požadavky a daty. Jejím úkolem je provádět aplikační logiku.

    V rámci tohoto systému bude tato vrstva obsažena v backendové aplikaci, která provádí aplikační logiku. Tato backendová aplikace by následně měla být nasazena a spuštěna na serveru, aby byla dostupná pro komunikaci s prezentačními vrstvami, tj. mobilními aplikacemi.
  - **Datová vrstva:** Nejnižší vrstva architektury, která se zabývá prací s daty, tj. správou databázového systému a základními datově-funkčními operacemi, jako je ukládání, výběr, agregace, předzpracování a audit dat.

    V této architektuře je datová vrstva reprezentována databázovým systémem, do kterého backendová aplikace ukládá a z něhož čte data.

Taková architektura umožňuje efektivní a robustní zpracování dat o polohách dopravních prostředků a umožňuje snadnou komunikaci mezi mobilními aplikacemi a backendem [12].

2.2 Programovací jazyk

2.2.1 Pro vývoj mobilních aplikací

Existuje několik programovacích jazyků, které lze použít pro vývoj mobilních aplikací pro platformu Android. Mezi nejznámější patří Java a Kotlin. Vzhledem k mnoha faktorům, včetně moderní syntaxe, interoperability s Javou, široké podpory pro vývojářské nástroje, je Kotlin považován za jednoznačně lepší volbu pro vývoj mobilních aplikací pro platformu Android. Kotlin přináší významný přínos v prevenci běžného problému *null pointer exceptions* díky svému typovému systému, který rozlišuje mezi referenčními a nereferenčními typy, umožňujícím odhalení potenciálních chyb již v době kompilace. Tento jazyk byl schválen Googlem jako preferovaný jazyk pro vývoj Android aplikací a má stále rostoucí komunitu vývojářů. Použití Kotlinu pro vývoj mobilních aplikací umožňuje snazší správu a údržbu systému díky jeho jasnější syntaxi a bezpečnosti typů, což přispívá k rychlejšímu vývoji a nižším nákladům na údržbu [13].

2.2.2 Pro vývoj backendu

Stejně jako pro vývoj mobilních aplikací, Kotlin je také vynikající volbou pro vývoj backendových aplikací, díky jeho bezpečnosti typů a moderním funkcím. Použití jednoho jazyka pro psaní jak mobilních aplikací, tak i backendových aplikací má řadu výhod, protože lze využít stejné vývojářské dovednosti a nástroje jako při vývoji mobilních aplikací.

2.3 Backend

V této sekci bude proveden návrh backendové části.

2.3.1 Framework

Výběr vhodného frameworku pro implementaci backendové části aplikace je klíčovým rozhodnutím, které ovlivní efektivitu, flexibilitu a udržitelnost celého systému. Existuje několik populárních frameworků pro tvorbu webových aplikací v jazyku Kotlin, z nichž každý má své vlastní výhody a nevýhody.

- **Spring Boot:** Spring Boot je výkonný framework založený na jazyce Java, který nabízí širokou škálu funkcí pro tvorbu robustních a škálovatelných webových aplikací. Spring Boot má velkou komunitu uživatelů a bohatou dokumentaci, což z něj činí populární volbu pro rozsáhlé projekty [14].
- **Vert.x:** Vert.x je framework navržený pro tvorbu vysokovýkonných webových aplikací. Je vhodný pro situace, kdy je potřeba zpracovávat velké množství požadavků s minimální reakční dobou. Vert.x podporuje asynchronní programování a umožňuje psát kód ve více jazycích, včetně Kotlinu [15].
- **Jooby:** Jooby je moderní framework pro tvorbu webových aplikací v jazyce Kotlin. Nabízí jednoduché a intuitivní API pro tvorbu webových aplikací. Jooby je známý svou modularitou a flexibilitou, což umožňuje snadnou integraci s různými technologiemi a knihovnami [16].
- **Http4k:** Http4k je lehký a funkcionální framework pro tvorbu webových aplikací v jazyce Kotlin. Je navržen tak, aby byl jednoduchý na použití a snadno rozšiřitelný pomocí funkcionálního programování. Http4k je ideální volbou pro malé a střední projekty [17].
- **Ktor:** Ktor je jedním z nejpopulárnějších frameworků pro tvorbu webových aplikací v jazyce Kotlin. Nabízí vysokou efektivitu v asynchronním zpracování požadavků, jednoduché a intuitivně použitelné API a podporu pro korutiny pro asynchronní programování. Ktor je aktivně podporován komunitou vývojářů a je zaměřen na jednoduchost a rychlost vývoje [18].

Proto byl pro tento projekt zvolen framework Ktor. Důvodem pro tuto volbu je kombinace několika faktorů, které z Ktoru dělají ideálního kandidáta pro backendovou část aplikace.

První výhodou Ktoru je jeho jednoduchost použití a minimalismus. Ktor se zaměřuje na poskytnutí elegantního a intuitivního API pro tvorbu webových aplikací, což umožňuje rychlý vývoj a snadnou údržbu kódu. Díky tomu je Ktor vhodným řešením pro projekty s menšími týmy vývojářů nebo s omezenými zdroji.

Další výhodou Ktoru je jeho integrace s jazykem Kotlin. Ktor je plně kompatibilní s Kotlinem a využívá jeho moderních vlastností, jako jsou rozšíření funkcí, lambdy a korutiny. To umožňuje psát čistý a konzistentní kód, což zvyšuje produktivitu a snižuje riziko chyb.

Dalším důležitým faktorem je vysoká výkonnost a efektivita Ktoru. Framework je navržen tak, aby poskytoval rychlou odezvu a vysoký výkon i při zpracování velkého počtu požadavků. To je zásadní pro aplikace, které se zaměřují na poskytování reálných časových dat nebo interakci s velkým počtem uživatelů současně.

Kromě toho má Ktor aktivní a rozvíjející se komunitu vývojářů, která poskytuje podporu, aktualizace a nové funkce. To zajišťuje, že framework zůstává aktuální a spolehlivý i v dlouhodobém horizontu.

Celkově lze tedy říci, že volba frameworku Ktor byla motivována jeho jednoduchostí, integrací s jazykem Kotlin, vysokou výkonností a aktivní komunitou vývojářů. Tyto faktory společně přispívají k efektivnímu a spolehlivému vývoji backendové části aplikace a poskytují pevný základ pro budoucí rozvoj a údržbu systému.

2.3.2 Architektonický styl a API

2.3.2.1 REST

REST (Representational State Transfer) je architektonický styl, který definuje sadu pravidel pro vytváření webových služeb, které jsou snadno čitelné a pochopitelné pro lidi i stroje. Jeho základním principem je práce se stavem prostřednictvím standardních operací HTTP. RESTová architektura preferuje jednoduchost, modularitu a škálovatelnost, což je ideální pro vývoj moderních webových aplikací.

Jeden z důvodů, proč byl REST API vybrán v této konkrétní aplikace, je jeho schopnost poskytnout standardní rozhraní pro komunikaci mezi klienty a serverem pomocí běžných HTTP metod, jako jsou GET, POST, PUT a DELETE. Tato jednotná rozhraní umožňují snadnou implementaci API.

- Základní HTTP metody a jejich využití:
  - **GET:** Tato metoda je určena k získání dat ze serveru. Klient pomocí metody GET posílá požadavek na server a server odpovídá zaslaním požadovaných dat. GET by mělo být idempotentní, což znamená, že opakované požadavky na stejný zdroj by neměly mít žádný vedlejší efekt.
  - **POST:** Metoda POST je používána pro odesílání dat na server k vytvoření nového prostředku nebo provádění nějakého akce na serveru. Může být použita k vytvoření nového záznamu v databázi, odeslání formuláře nebo provádění jakékoliv jiné akce, která má za následek změnu stavu na serveru.
  - **PUT:** Metoda PUT se používá pro aktualizaci existujících dat na serveru. Klient pošle kompletní reprezentaci zdroje na server a server buď vytvoří nový zdroj, pokud neexistuje, nebo aktualizuje existující zdroj podle poslaných dat.
  - **DELETE:** Tato metoda slouží k odstranění určitého zdroje na serveru. Klient pošle požadavek DELETE na specifický zdroj a server poté odstraní tento zdroj ze svého úložiště.

2.3.2.2 API

Byly vytvořeny různé API endpoints, které slouží k předávání informací o aktuální poloze uživatelů a následně k umožnění ostatním uživatelům provádět dotazy a získávat tuto

polohu. Tyto endpoints jsou jednoduše přístupné pomocí standardních HTTP metod. Tímto způsobem je dosaženo snadného a efektivního propojení klientů s backendem a poskytnutí potřebných dat a funkcí pro uživatele aplikace.

- Zde jsou konkrétní API endpoints pro komunikaci s backendem:
  - **@POST: /auth/register** - Slouží k registraci nového uživatele.
  - **@POST: /auth/login** - Slouží k přihlášení uživatele.
  - **@POST: /auth/refresh** - Slouží k obnovení access a refresh tokenu.
  - **@GET: /api/v1/users/me** - Slouží k získání informací o aktuálně přihlášeném uživateli.
  - **@POST: /api/v1/incidents** - Slouží k vytvoření nového incidentu [1].
  - **@GET: /api/v1/incidents/?latitude={val}&longitude={val}** - Slouží k získání všech incidentů v okolí. Parametry `latitude` a `longitude` určují polohu uživatele, který provádí dotaz. To znamená, že incidenty budou hledány v blízkosti této pozice.
  - **@DELETE: /api/v1/incidents/{id}** - Slouží k odstranění konkrétního incidentu z aktivních incidentů na základě jeho identifikátoru, čímž dochází k deaktivaci sdílení polohy.
  - **@POST: /api/v1/incidents/locations** - Slouží k zaznamenání nové polohy pro incident.

2.3.3 Databáze

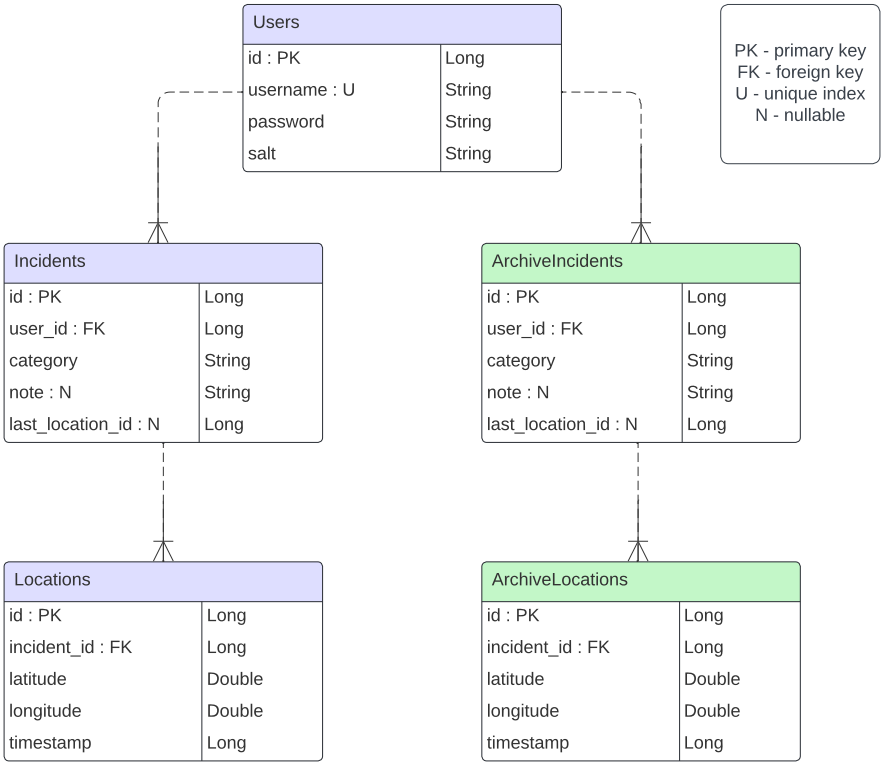
2.3.3.1 Výběr databázového systému

V jazyce Kotlin existuje vynikající řešení pro práci s databází. Jedná se o ORM framework nazývaný Exposed. Tento framework může pracovat s databázemi jako je MariaDb, MySQL, Oracle, PostgreSQL, SqLite, Microsoft SQL Server [19].

PostgreSQL byl zvolen pro backendovou aplikaci z několika důvodů. Je open-source, což znamená, že je zdarma dostupný a jeho zdrojový kód je veřejně přístupný, což usnadňuje jeho úpravu a přizpůsobení. Tento databázový systém je také jedním z nejoblíbenějších na trhu, což znamená, že má rozsáhlou komunitu uživatelů a dostatek zdrojů a dokumentace pro řešení potenciálních problémů. PostgreSQL je multiplatformní a podporuje mnoho operačních systémů, což umožňuje jeho nasazení na různých platformách a infrastrukturních prostředích. Dále poskytuje efektivní zpracování, indexaci a optimalizaci dotazů, což je důležité pro rychlou a spolehlivou práci s daty v aplikaci. Bezpečnost dat je také klíčovým faktorem při výběru databázového systému. PostgreSQL klade důraz na bezpečnost pomocí funkcí jako je správa přístupu založená na rolích a SSL šifrování, což zajišťuje ochranu citlivých informací a zabraňuje neoprávněnému přístupu k datům [20].

[1] V rámci této práce bude použit termín „incident“ pro událost, která začíná svůj životní cyklus, když uživatel v aplikaci LocShare začne sdílet svou polohu. V této fázi se vytvoří „incident“ a poloha uživatele bude pravidelně odesílána na server.

2.3.3.2 Databázový návrh



Obrázek 2.2 ER diagram

Tabulky v databázi 2.2 :

1. **Users** (uživatelé): Tato tabulka obsahuje informace o registrovaných uživatelích, včetně jejich identifikátoru, uživatelského jména, hashu hesla a soli, která se používá k bezpečnému vytváření hashe hesla.
2. **Incidents** (události): Tabulka událostí obsahuje informace o událostech, které začínají svůj životní cyklus, když uživatel začne sdílet svou polohu v aplikaci. Obsahuje data, jako je identifikátor události, identifikátor uživatele, kategorie vozidla, poznámka a identifikátor poslední polohy.
3. **Locations** (polohy): Tabulka pro záznamy poloh. Obsahuje data, jako je identifikátor místa, identifikátor události, souřadnice polohy a čas záznamu.

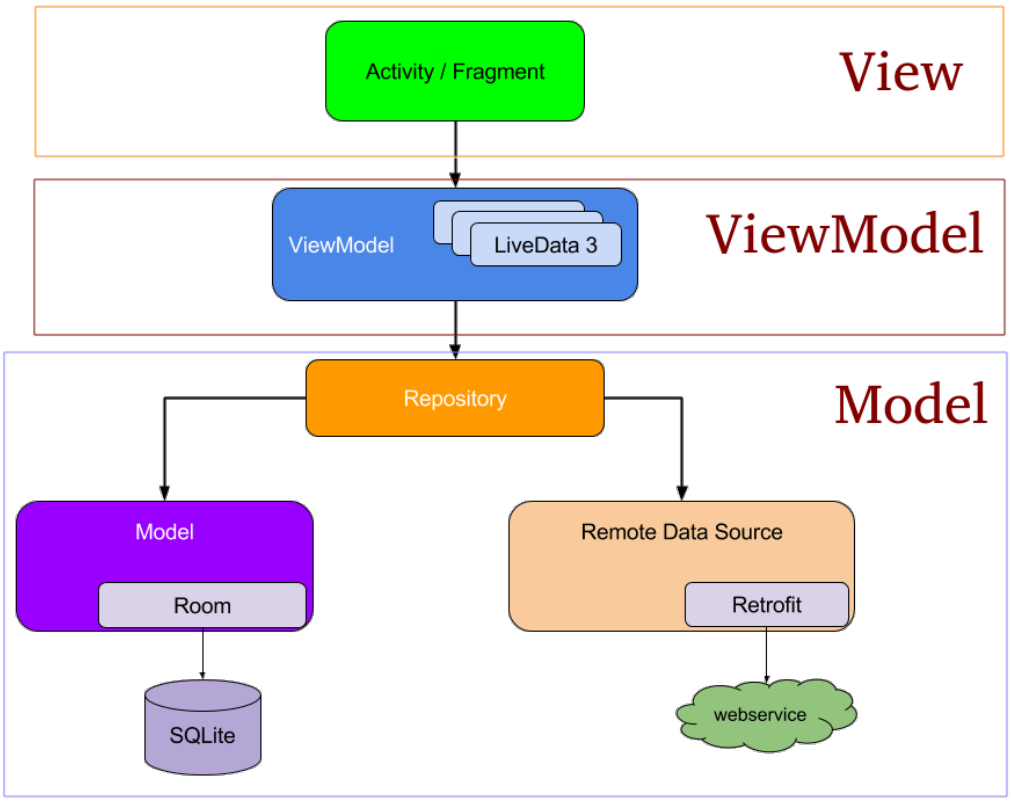
Nyní bude podrobně prozkoumáno poslední dvě tabulky. Když událost přestane být aktivní (tj. uživatel aplikace LocShare přestane sdílet svou polohu), záznamy v tabulkách Incidents a Locations spojené s konkrétní ukončenou událostí jsou přesunuty do tabulek ArchiveIncidents a ArchiveLocations, kde jsou uloženy jako archivní záznamy. Toto se provádí s cílem nezatěžovat tabulky Incidents a Locations a umožnit rychlé vyhledávání v budoucnu, kdy bude potřeba hledat aktivní události.

4. **ArchiveIncidents** (archivní události): Tato tabulka slouží k archivaci starých dat o událostech, které již byly deaktivovány. Je to obdobná tabulka jako tabulka událostí, ale pro archivované záznamy.
5. **ArchiveLocations** (archivní polohy): Tabulka pro záznamy archivních poloh. Je to obdobná tabulka jako tabulka poloh, ale obsahuje údaje o polohách souvisejících s archivovanými událostmi.

2.4 Mobilní aplikace

V této sekci bude proveden návrh dvou mobilních aplikací:
LocShare a LocTracker [2].

2.4.1 Architektura



Obrázek 2.3 Architektura MVVM [21]

Při rozhodování o architektuře pro mobilní aplikace LocShare a LocTracker bylo zapotřebí zvážit několik možností, které by nejen poskytly efektivní a udržitelné řešení,

[2] Připomenutí: Mobilní aplikace LocShare bude sloužit k odesílání vlastní polohy (např. prostřednictvím mobilního telefonu v vozidle IZS). Mobilní aplikace LocTracker bude sloužit k přijímání a zobrazování polohy zařízení LocShare, která se nacházejí v blízkosti a momentálně sdílí svou polohu.

ale také usnadnily vývoj a údržbu kódu. Mezi běžně používané architektonické vzory patří například MVC (Model-View-Controller), MVP (Model-View-Presenter) a MVVM (Model-View-ViewModel).

Po důkladném zhodnocení všech možností byla jako optimální volba vybrána architektura MVVM (Model-View-ViewModel) 2.3. Tento výběr byl opřen o několik klíčových faktorů, které přispívají k její přednosti nad ostatními architekturami.

MVVM poskytuje jasnou separaci mezi daty (Model), uživatelským rozhraním (View) a logikou aplikace (ViewModel). Tato struktura umožňuje snadnější sledování a úpravu každé části aplikace nezávisle na ostatních. Oproti tomu u MVC a MVP může být kód značně provázaný, což může vést k obtížím při úpravě a testování.

Dalším klíčovým faktorem pro volbu MVVM je jeho podpora pro databinding. Tato funkce umožňuje automatickou synchronizaci dat mezi modelem a uživatelským rozhraním, což eliminuje potřebu ruční aktualizace zobrazení. To zjednodušuje vývoj uživatelského rozhraní a snižuje možnost chyb.

Celkově lze tedy říci, že MVVM architektura nabízí přehlednou a flexibilní strukturu, která usnadňuje správu a rozšíření aplikace v budoucnosti.

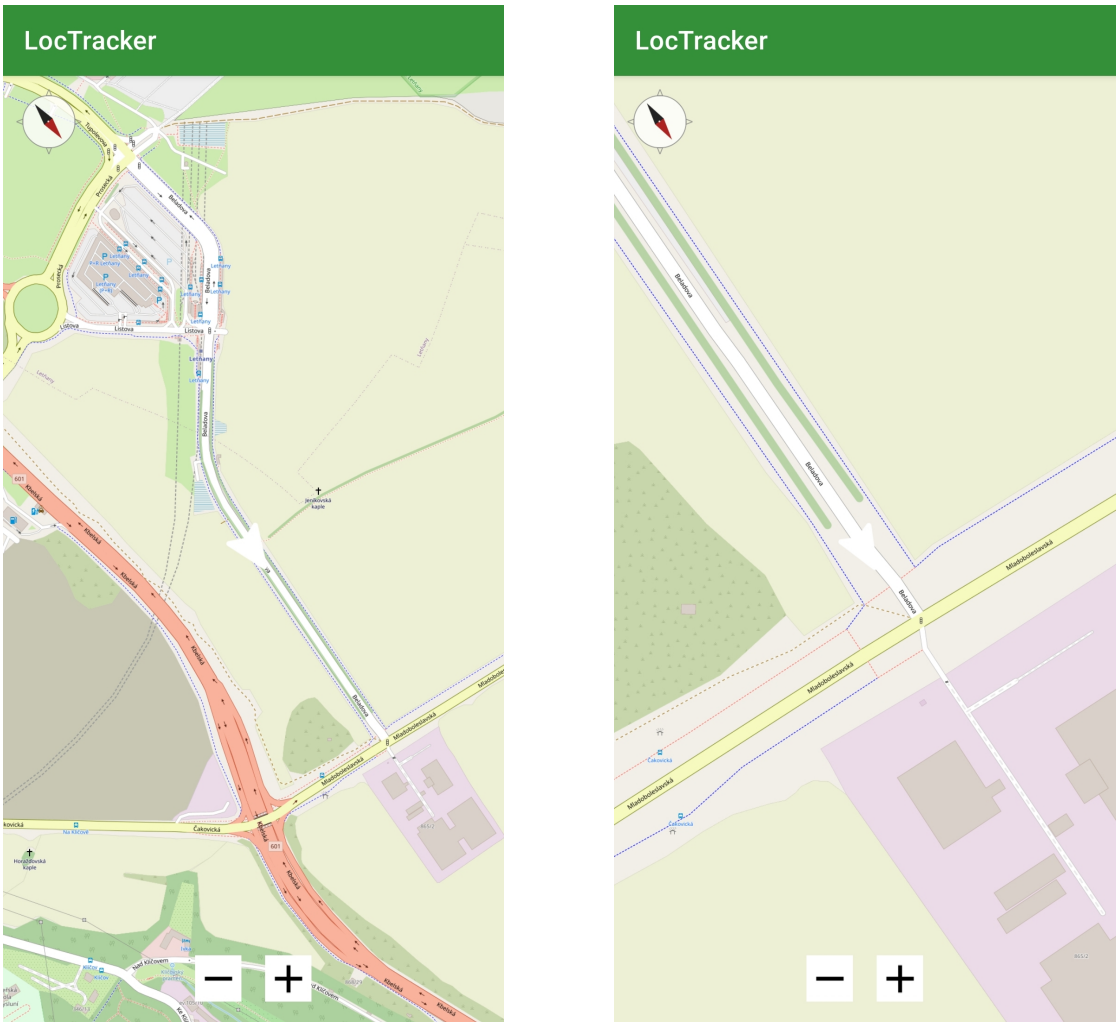
2.4.2 Nástroje pro zobrazení mapy

Zobrazení mapy hraje klíčovou roli v mobilních aplikacích LocShare a LocTracker, neboť přesné a aktuální informace o poloze vozidel jsou zásadní pro tento systém. V obou aplikacích je uživatelům poskytováno zobrazení mapy.

Při výběru vhodného řešení pro zobrazení mapy bylo zapotřebí zvážit několik faktorů, včetně integrace s ostatními funkcemi aplikace, uživatelské přívětivosti a efektivity Mezi populární možnosti patří Google Maps SDK, Mapbox SDK, HERE SDK a OsmAnd SDK.

- **Google Maps SDK:** Jedním z nejpopulárnějších nástrojů pro zobrazení mapy je Google Maps SDK. Tento nástroj poskytuje bohaté možnosti zobrazení mapy včetně navigace, vyhledávání a interakce s uživatelem. Google Maps SDK je velmi snadno integrovatelný do aplikací a poskytuje širokou škálu funkcí, ale je placený [22].
- **Mapbox SDK:** Mapbox SDK je další možností pro zobrazení interaktivních map v mobilních aplikacích. Tento nástroj poskytuje možnosti úpravy vzhledu mapy a rozšířené funkce pro geolokaci. Mapbox SDK je dostupný v placené verzi s rozšířenými funkcemi, avšak nabízí i bezplatnou verzi pro vývojáře s omezeným rozsahem funkcí [23].
- **Here SDK:** Here SDK je další možností pro zobrazení mapy v aplikacích. Tento nástroj poskytuje širokou škálu funkcí, včetně navigace, geolokace a vyhledávání. Here SDK nabízí placené verze s rozšířenými funkcemi pro komerční použití, ale také poskytuje bezplatnou verzi s omezenými možnostmi pro vývojáře [24].
- **OsmAnd SDK:** OsmAnd SDK nabízí několik výhod pro vývojáře mobilních aplikací. Poskytuje možnost přizpůsobení, což umožňuje vývojářům upravovat vzhled a chování mapy podle specifických požadavků jejich aplikace. Jako open-source projekt má OsmAnd SDK otevřený zdrojový kód, což umožňuje vývojářům přístup k veškerému kódu a možnost jeho upravování podle potřeb. Široká škála funkcí, jako jsou

různé vrstvy mapy, navigace, vyhledávání míst a geolokace, poskytuje vývojářům rozmanité možnosti pro tvorbu bohatých a interaktivních mapových aplikací. Díky aktivní komunitě uživatelů a vývojářů je OsmAnd SDK neustále podporováno a zdokonalováno prostřednictvím komunitních fór, což zajišťuje stabilní prostředí a možnost spolupráce na dalším rozvoji nástroje [25] [26].



Obrázek 2.4 Ukázka zobrazení mapy v aplikaci LocTracker. Auto je v pohybu

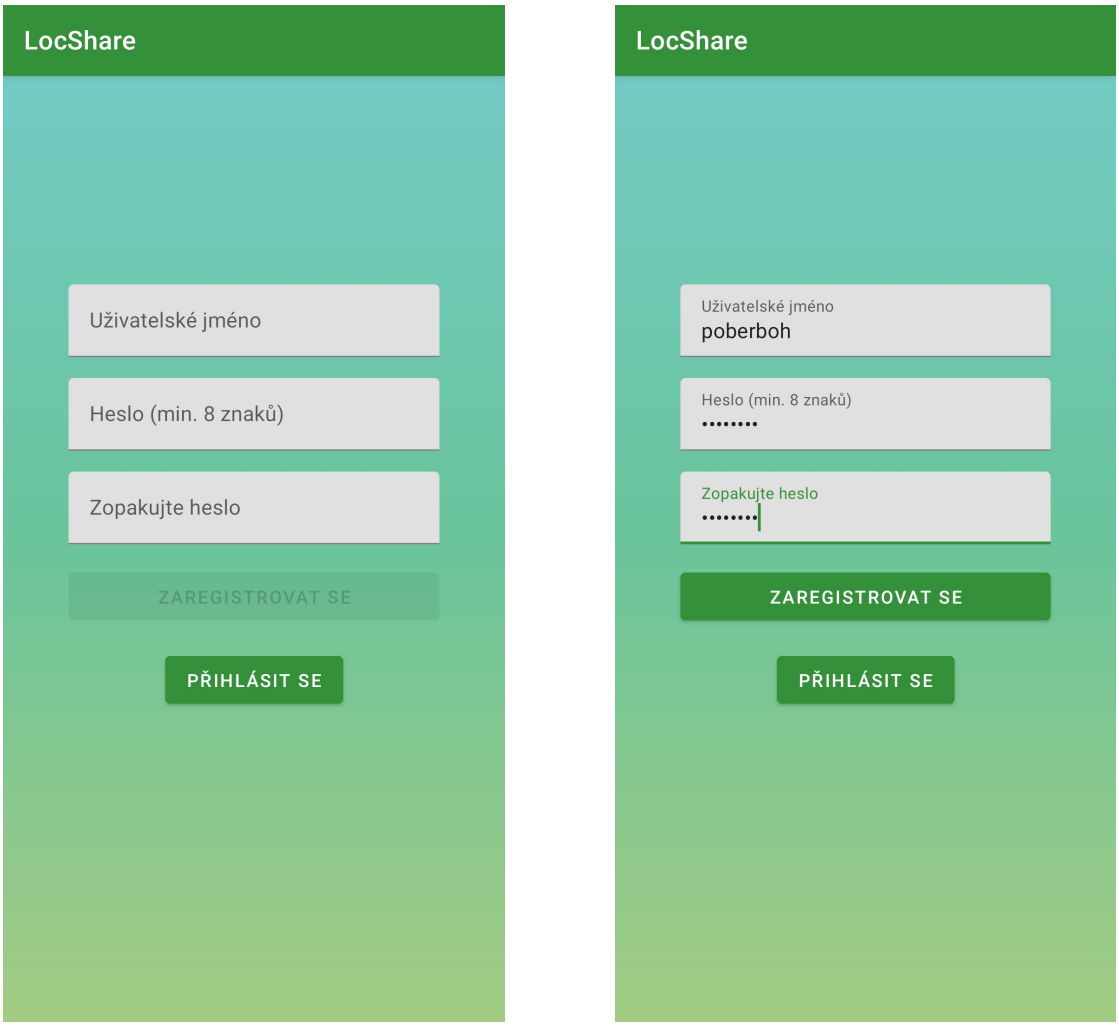
Po důkladném zvážení všech možností bylo rozhodnuto použít OsmAnd SDK, především kvůli jeho open-source povaze a dostupnosti zdarma. Tato volba umožní snadnou úpravu a přizpůsobení mapového rozhraní specifickým potřebám aplikací LocShare a LocTracker 2.4.

2.4.3 Design

Při návrhu designu pro aplikace LocShare a LocTracker je důležité vzít v úvahu jejich funkční a nefunkční požadavky a zajistit uživatelskou přívětivost a atraktivní uživatelské rozhraní.

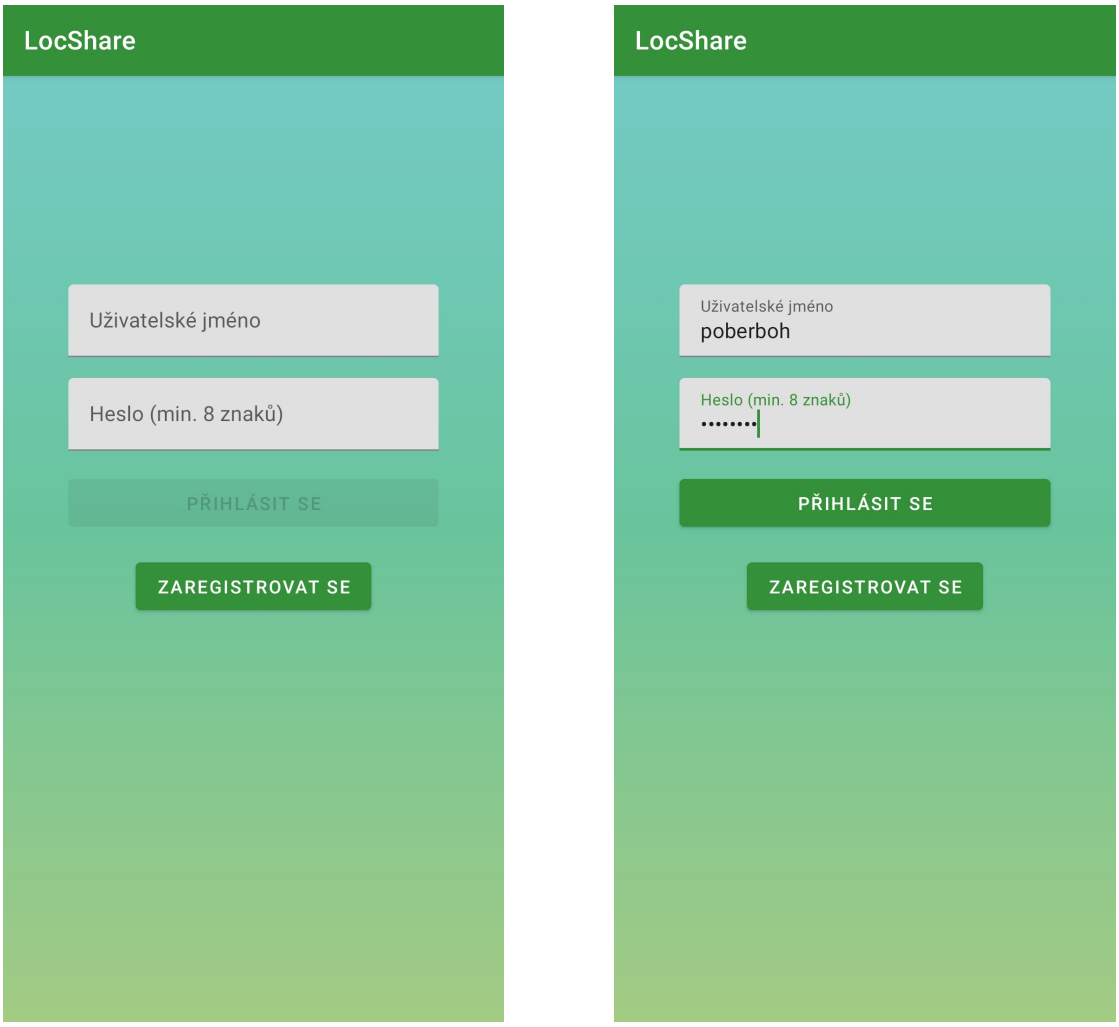
2.4.3.1 LocShare

Pro aplikaci LocShare, která umožňuje sdílení polohy a typu dopravního prostředku, je důležité vytvořit intuitivně srozumitelné rozhraní, které uživatelům umožní snadno se registrovat, přihlašovat a používat základní funkce:



Obrázek 2.5 Registrace v aplikaci LocShare

- Rozhraní pro registraci a přihlášení by mělo umožňovat uživatelům zadat své uživatelské jméno a heslo, aby se mohli registrovat nebo přihlásit do aplikace (požadavky:

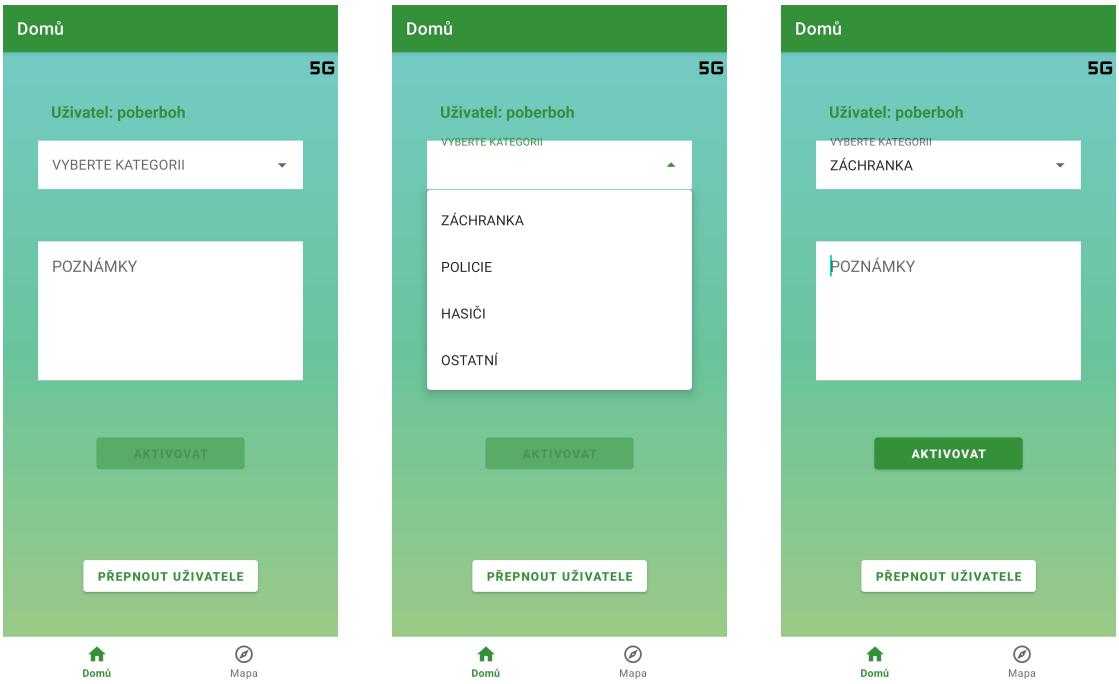


Obrázek 2.6 Přihlášení v aplikaci LocShare

F1 1.3.1.1, F2 1.3.1.1). Dále by mělo poskytovat možnost odhlášení (požadavek: F3 1.3.1.1).

- Aplikace by měla zobrazovat mapu s označením aktuální polohy uživatele (požadavek : F4 1.3.1.1), stejně jako tlačítka pro aktivaci a deaktivaci odesílání polohy a výběr typu dopravního prostředku (požadavek: F6 1.3.1.1).
- Uživatel může snadno vybrat typ dopravního prostředku a přidat k němu poznámky prostřednictvím uživatelského rozhraní (požadavek: F6 1.3.1.1).
- Zobrazení stavu připojení k síti 5G by mělo být přehledné, umožňující uživateli snadno rozpoznat, kdy je jeho telefon připojen k síti 5G (požadavek: F7 1.3.1.1).

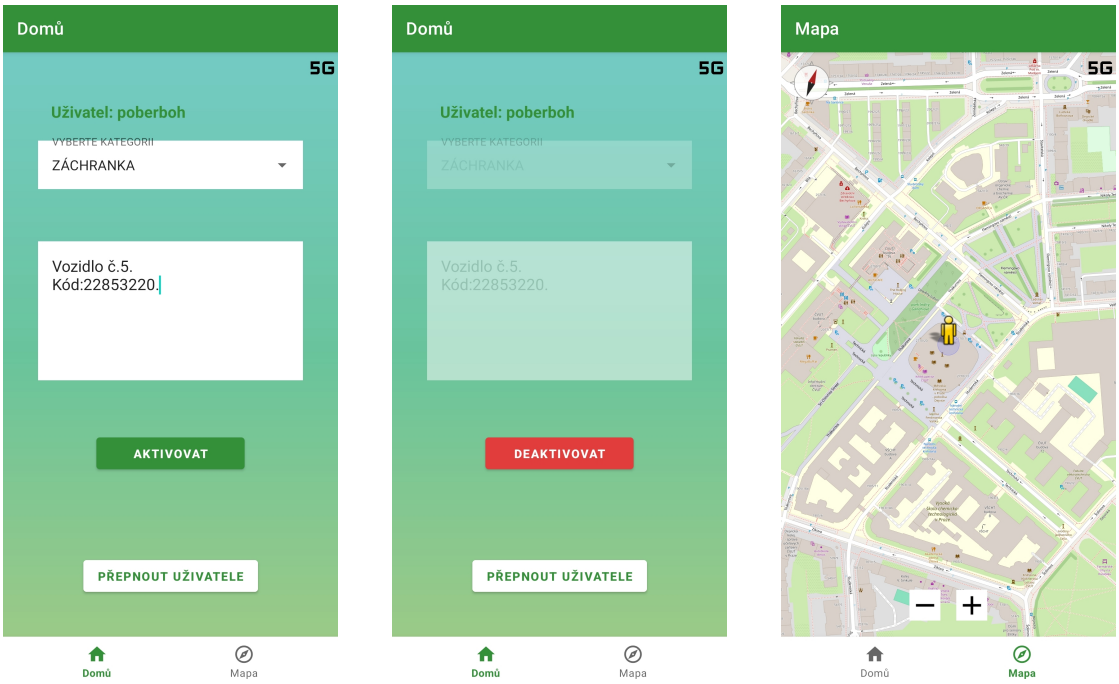
Po spuštění aplikace LocShare uživatel se dostane na stránku přihlášení. 2.6, kde se může přihlásit ke svému účtu pomocí uživatelského jména a hesla. Uživatel nemůže kliknout na tlačítko (`Přihlásit se`), dokud nevyplní všechna povinná pole. Na této



Obrázek 2.7 Hlavní obrazovka v aplikaci LocShare. Výběr kategorie vozidla

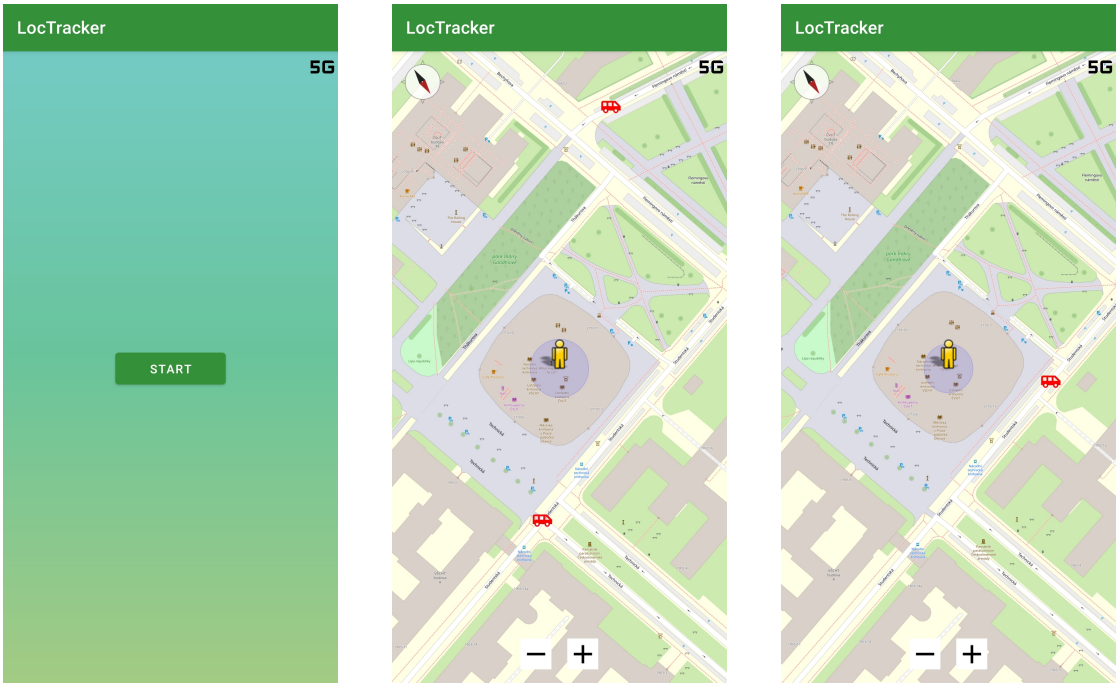
obrazovce je také přítomno tlačítko (`Zaregistrovat se`), které uživatele přesměruje na stránku registrace 2.5, kde stačí vyplnit uživatelské jméno, heslo a zopakovat heslo. Uživatel nemůže kliknout na tlačítko (`Zaregistrovat se`), dokud nevyplní všechna povinná pole. Registrace je zjednodušena, protože slouží pouze k identifikaci uživatele. Po úspěšné registraci je uživatele přesměrován na stránku přihlášení.

Úspěšné přihlášení přesměruje uživatele na hlavní stránku aplikace LocShare , kde jsou k dispozici pole pro výběr kategorie dopravního prostředku 2.7. V současné době jsou definovány čtyři typy: „ZÁCHRANKA“, „POLICIE“, „HASIČI“, „OSTATNÍ“. Dokud uživatel nevybere typ dopravního prostředku, nemůže kliknout na tlačítko (`Aktivovat`) pro spuštění sdílení své polohy. Uživatel také nemůže měnit typ vozidla a poznámky, když je aktivní funkce sdílení polohy. Pole pro poznámky je nepovinné pole, kam lze zapsat libovolné informace, které je potřeba uložit. Dále je zde tlačítko (`Přepnout uživatele`), které umožňuje uživateli odhlásit se a vrací ho na obrazovku přihlášení. Na dolní části obrazovky se nachází navigační lišta, pomocí které lze přepínat mezi hlavním a mapovým zobrazením 2.8. Na mapové obrazovce uživatel uvidí svou polohu. Když uživatel nepohybuje, jeho poloha bude zobrazena pomocí ikonky žlutého mužíčka, ale pokud se pohybuje, ikonka bude bílá šipka ukazující směr pohybu.



Obrázek 2.8 Hlavní obrazovka a mapa v aplikaci LocShare. Vyplnění poznámky a aktivace odesílání své polohy

2.4.3.2 LocTracker



Obrázek 2.9 Ukázka aplikace LocTracker. Úvodní obrazovka a mapa

Pro aplikaci LocTracker, která slouží k sledování polohy dopravních prostředků, je důležité vytvořit rozhraní, které uživatelům umožní snadno sledovat polohu dopravních prostředků:

- Obrazovka zobrazující mapu by měla být srozumitelná, s možností zobrazení polohy všech sledovaných dopravních prostředků (požadavky: F8 1.3.2.1, F10 1.3.2.1).
- Zobrazení stavu připojení k síti 5G by mělo být přehledné, umožňující uživateli snadno rozpoznat, kdy je jeho telefon připojen k síti 5G (požadavek: F9 1.3.2.1).

Po spuštění aplikace se uživatel dostane na úvodní obrazovku, kde bude k dispozici tlačítko (`Start`). Po jeho stisknutí začne aplikace zobrazovat mapu, na které bude aktuální poloha uživatele. Pokud je uživatel v klidu, jeho poloha bude zobrazena pomocí ikonky žlutého mužíčka, ale pokud se pohybuje, ikonka bude bílá šipka ukazující směr pohybu. Umístění zařízení LocShare, která jsou v daný okamžik v blízkosti a sdílí svou polohu, bude zobrazeno pomocí ikony červené dodávky. Vše je viditelné na následujících obrázcích 2.9 a 2.4.

2.5 Automatizace sestavení programu

Automatizace sestavení programu je proces kompilace zdrojového kódu, následující zabalení binárního kódu a spuštění automatizovaných testů. Tento proces je klíčovým prvkem moderního vývoje softwaru, protože umožňuje vývojářům efektivně spravovat a distribuovat svůj kód bez zbytečného manuálního zásahu.

V rámci vývoje aplikace v jazyce Kotlin existují různé nástroje pro automatizaci sestavení, mezi které patří Maven, Gradle Kotlin a Gradle Groovy.

- **Maven:** Maven je nástroj pro řízení softwarových projektů, který se používá k automatickému sestavení, dokumentaci a správě závislostí. Používá XML formát pro konfiguraci a je populární zejména v prostředí jazyka Java [27].
- **Gradle Kotlin:** Gradle Kotlin je další nástroj pro automatizaci sestavení programu, který umožňuje psát skripty sestavení pomocí jazyka Kotlin. Tím se získává výhoda v přirozenosti zápisu a možnosti využití výhod jazyka Kotlin, jako jsou lambda výrazy a rozšíření [28].
- **Gradle Groovy:** Kromě Gradle Kotlin je možné použít také standardní Gradle s Groovy skripty. Gradle Groovy je široce používaný nástroj sestavení s bohatou funkcionalitou a rozsáhlou dokumentací [29].

Pro automatizaci sestavení programu byl vybrán Gradle Kotlin. Tento nástroj poskytuje výhody v čitelnosti, stabilitě a integraci, což bylo rozhodujícím faktorem pro jeho výběr jak pro backend, tak pro mobilní aplikace. Každá aplikace bude mít svůj vlastní sestavovací skript napsaný v Gradle Kotlin.

2.6 Autentizace

Tato sekce se zabývá autentizací uživatelů v aplikaci a zajištěním bezpečného přístupu k jejím prostředkům. Pro tyto účely využívá aplikace JWT (JSON Web Token).

2.6.1 JWT

JWT (JSON Web Token) je používán pro autentizaci uživatelů a zajištění bezpečné komunikace mezi klientem a serverem. Tento mechanismus umožňuje serveru vytvořit token obsahující informace o uživateli a jeho oprávněních, který je posléze klientovi posílán a využíván pro ověření jeho identity při každém požadavku na chráněné prostředky [30].

- **Princip fungování JWT:**
  1. Uživatel úspěšně projde autentizací pomocí svých přihlašovacích údajů (uživatelské jméno a heslo).
  2. Server vytvoří JWT obsahující unikátní údaje uživatele (například identifikátor uživatele) a další informace o platnosti tokenu.
  3. JWT je odeslán klientovi, který jej uloží (například do lokálního úložiště).
  4. Při každém požadavku na chráněné prostředky klient zahrnuje JWT do hlavičky požadavku.
  5. Server ověří platnost JWT, autentizuje uživatele a povolí nebo zamítne požadavek v závislosti na výsledku ověření.

- **Výhody použití JWT:**
  - Bezpečnost: JWT je podepsán serverem, což zajišťuje jeho integritu a autentičnost.
  - Nezávislost na stavu: JWT obsahuje všechny potřebné informace, takže není třeba ukládat stav autentizace na serveru.
  - Škálovatelnost: JWT může být předán mezi různými komponentami systému bez nutnosti přístupu k centralizovanému úložišti.

2.6.2 Refresh token

Problém, se kterým se může setkat aplikace při používání JWT, spočívá v možnosti, že útočník získá platný token, což by mohlo vést k ohrožení bezpečnosti aplikace. Aby se tento problém vyřešil, je nutné omezit dobu platnosti JWT. Pokud je však doba platnosti krátká, může to vést k neustálému opakovanému přihlašování uživatele. Proto byl navržen mechanismus refresh tokenu.

Refresh token je dlouhodobě platný token, který slouží k obnovení platného JWT bez nutnosti zadávat přihlašovací údaje znovu. Když platnost JWT vyprší, klient použije refresh token k získání nového JWT od autorizačního serveru. Tím se zajistí, že uživatel zůstane přihlášený, aniž by musel opakovaně zadávat své přihlašovací údaje, a zároveň je zajištěna bezpečnost aplikace pomocí omezené doby platnosti JWT [31].

Tento mechanismus poskytuje vyvážený přístup mezi bezpečností a uživatelským pohodlím, přičemž chrání aplikaci před možnými útoky na zneužití dlouhodobě platného JWT. Proto pro zajištění stabilního a plynulého fungování autentizačního systému bude v aplikaci použit nejenom samotný JWT jako access token, ale také refresh token.

Kapitola 3

Implementace

V této kapitole je proveden popis implementace navržených řešení. Nejprve je zaměřeno na backendovou část aplikace. Poté je věnována pozornost implementaci mobilních aplikací.

3.1 Backend

3.1.1 Struktura backendu

V této části je popsána struktura backendu. Byla navržena s důrazem na snadnou rozšiřitelnost, testovatelnost a zajištění vysokého výkonu a bezpečnosti.

Obecná struktura:

- `/loc_backend` – Kořenová složka projektu
  - `build` – Složka obsahující sestavené soubory a dokumentaci
  - `gradle/wrapper` – Složka obsahující konfiguraci Gradle wrapperu
  - `src` – Zdrojové kódy projektu
    - `main` – Hlavní zdrojové soubory
    - `test` – Testy
  - `build.gradle.kts` – Soubor s konfigurací Gradle
  - `Dockerfile` – Konfigurační soubor pro Docker
  - `gradle.properties` – Konfigurační soubor Gradle
  - `gradlew` – Spustitelný soubor Gradle wrapperu
  - `README.md` – Soubor obsahující základní informace o projektu
  - `settings.gradle.kts` – Soubor s nastavením projektu pro Gradle

Struktura /src/main:

```
/src/main ..... Hlavní zdrojové soubory
├── koltin/cz/cvut/fit/poberboh/loc_backend .....
│   ├── dao ..... Složka obsahující Data Access Objects (DAO)
│   │   ├── archive ..... DAO pro správu archivních dat
│   │   │   ├── incidents ..... DAO pro správu archivních incidentů
│   │   │   └── locations ..... DAO pro správu archivních lokací
│   │   ├── incidents ..... DAO pro správu incidentů
│   │   ├── locations ..... DAO pro správu lokací
│   │   └── users ..... DAO pro správu uživatelů
│   ├── database ..... Složka pro konfiguraci databáze
│   │   └── tables ..... Složka pro definice tabulek v databázi
│   ├── di ..... Složka obsahující DAO provider
│   ├── models ..... Složka s datovými modely
│   ├── network ..... Definované API cesty (routes)
│   │   ├── auth ..... API endpoints pro autentizaci uživatelů
│   │   ├── incident ..... API endpoints pro správu incidentů
│   │   └── user ..... API endpoints pro správu uživatelů
│   ├── plugins ..... Plugins balíček pro konfiguraci
│   ├── security ..... Složka pro zabezpečení
│   │   ├── hashing ..... Service pro hashování hesel
│   │   └── token ..... Service pro vytváření access a refresh tokenu
│   └── Application.kt ..... Hlavní vstupní bod aplikace
└── resources ..... Resource soubory
    ├── application.conf ..... Konfigurační soubor aplikace
    └── logback.xml ..... Konfigurační soubor pro logování
```

Celý backend je rozdělen do několika částí, které mají specifické úkoly a odpovědnosti.

- **Hlavní vstupní bod a konfigurace aplikace:** Kód začíná v hlavním vstupním bodě aplikace (main) v souboru `Application.kt` , který spouští server Ktor pomocí Netty. Důležitou součástí je funkce `module`, která konfiguruje různé komponenty aplikace, jako jsou konfigurace tokenů, databáze, zabezpečení, směrování (routing), logování a serializace. Tato část je zodpovědná za inicializaci serveru a jeho spuštění.

- **Směrování (routing):** V rámci směrování jsou definovány cesty (routes) pro různé části aplikace, které poskytují API endpointy pro komunikaci s klientem. Například máme API endpoint pro autentizaci (`auth`), pro manipulaci s incidenty (`incidents`). Každá cesta je definována jako samostatná funkce, která zpracovává požadavky klienta.

- **Data Access Objects (DAO):** DAO jsou zodpovědné za manipulaci s daty v databázi. Každý DAO poskytuje sadu metod pro práci s konkrétním typem dat. Například máme DAO pro uživatele, incidenty, lokace.

- **Inicializace databáze:** Objekt `DatabaseSingleton` se používá k připojení k databázi a případně k vytvoření tabulek.

- **Hashování hesel:** Třída `SHA256HashingService` poskytuje službu pro hashování hesel pomocí SHA-256 algoritmu. Tato služba generuje náhodnou sůl, která je připojena k heslu před hashováním.
- **Generování JWT tokenů:** Třída `JwtTokenService` poskytuje službu pro generování JWT tokenů. Tato služba vytváří access token a refresh tokeny.

Celkově backend aplikace funguje tak, že přijímá požadavky od klientů prostřednictvím definovaných API endpointů, zpracovává tyto požadavky, provádí potřebné operace v databázi pomocí DAO a poté vrátí odpověď klientovi. Tímto způsobem je dosaženo oddělení logiky aplikace od prezentační vrstvy a zajištěna snadná rozšiřitelnost, testovatelnost a udržovatelnost aplikace.

Toto je obecné shrnutí struktury backendu. V následujících částech budou jednotlivé části popsány podrobněji.

3.1.2 API

API bylo specifikováno v předchozí kapitole v sekci 2.3.2.2, nyní bude popsána jeho implementace.

3.1.2.1 Serializace dat

Pro výměnu dat mezi mobilními aplikacemi a backendem se používá formát JSON, který zajišťuje jednoduchost a efektivitu přenosu dat po síti. Pro podporu serializace a deserializace dat ve formátu JSON ve backendu se používá knihovna `kotlinx.serialization`, která umožňuje převádět objekty Kotlin do JSON a zpět. Konfigurace této serializace je demonstrována v ukázce kódu 3.1.

```kotlin
// /plugins/Serialization.kt
fun Application.configureSerialization() {
    install(ContentNegotiation) {
        json()
    }
}
```

Výpis kódu 3.1 Konfigurace serializace JSON

3.1.2.2 Zpracování požadavků

Pro zpracování požadavků od mobilních aplikací a odpovědí na ně se využívají `Route` v frameworku Ktor. Routes jsou definovány v rámci konfigurace směrování aplikace a určují, jakým způsobem jsou jednotlivé URL adresy obsluhovány.

V ukázce kódu 3.2 je definována konfigurace směrování aplikace, která obsahuje dvě hlavní části. První z nich obsluhuje autentikační API a druhá zpracovává veškeré ostatní požadavky na API, jako je například práce s incidenty.

Tímto způsobem je v Ktoru definováno směrování aplikace a je zajištěno, že každý příchozí požadavek je předán správné části backendu k zpracování. Dále Route komunikují s DAO pro provedení změn v databázi.

```
// /plugins/Routing.kt
fun Application.configureRouting(tokenConfig: TokenConfig) {
    routing {
        route("auth") {
            configureAuthApi(tokenConfig)
        }

        route("api/v1") {
            configureUserApi()

            route("incidents") {
                configureIncidentApi()
            }
        }
    }
}
```

Výpis kódu 3.2 Konfigurace routování aplikace

Tyto hlavní komponenty vzájemně spolupracují, aby zajistily fungování API backendu a umožnily klientům komunikovat s aplikací pomocí standardních HTTP požadavků.

3.1.3 Autentizace

V této sekci budou popsány základní principy implementace autentizace, včetně procesů přihlašování a registrace uživatelů do systému, a bude proveden popis konfigurace, generování a ověřování JWT tokenů.

3.1.3.1 Přihlášení a registrace

Tato sekce se zabývá procesem přihlašování a registrace uživatelů do systému.

Při registraci nového uživatele do systému se provádí několik kroků, které zajišťují bezpečné uložení jeho údajů do databáze a následný přístup do systému.

1. **Ověření platnosti údajů**: Při registraci uživatel odesílá na server požadavek obsahující své uživatelské jméno a heslo. Nejprve je nezbytné ověřit platnost těchto údajů a zkontrolovat, zda splňují požadavky systému, jako je minimální délka hesla a použití pouze povolených znaků.
2. **Kontrola unikátnosti uživatelského jména**: Pro zajištění unikátnosti je třeba zkontrolovat, zda zadané uživatelské jméno je již obsazeno v databázi. Pokud uživatelské jméno již existuje, registrace se neprovede a uživateli bude odeslána odpovídající chybová zpráva.

```kotlin
// /secutiry/hashing/SHA256HashingService.kt
class SHA256HashingService : HashingService {
    override fun generateSaltedHash
        (value: String, saltedLength: Int): SaltedHash {
            val salt = SecureRandom
                        .getInstance("SHA1PRNG")
                        .generateSeed(saltedLength)
            val saltAsHex = Hex.encodeHexString(salt)
            val hash = DigestUtils.sha256Hex("$saltAsHex$value")
            return SaltedHash(hash, saltAsHex)
    }

    override fun verify
        (value: String, saltedHash: SaltedHash): Boolean {
            return DigestUtils
            .sha256Hex(saltedHash.salt + value)
             == saltedHash.hash
    }
}
```

Výpis kódu 3.3 Implementace funkcí hashování pomocí SHA256

3. **Hashování hesla s použitím soli**: Pro zvýšení bezpečnosti je nezbytné hashovat heslo uživatele. Používá se metoda hashování `SHA-256`. Bez použití soli by stejný textový řetězec vždy poskytoval stejný hash, což by umožnilo útočníkům vytvářet a používat předpočítané tabulky jako `rainbow tables` k prolomení hesel. Abychom tomu zabránili, před hashováním hesla je generována náhodná a unikátní sůl, která je poté spojena s heslem uživatele a teprve poté je celý řetězec zahashován. Tímto způsobem zajišťujeme, že i pro stejná hesla budou v databázi uloženy odlišné hashe. Ukázka kódu pro generování a ověření hashe 3.3.

4. **Uložení údajů do databáze**: Po ověření platnosti údajů a unikátnosti uživatelského jména jsou všechny potřebné údaje (uživatelské jméno, hash hesla a sůl) uloženy do databáze.

Po úspěšné registraci je uživatel připraven k přihlášení do systému. Proces přihlášení je následující:

1. **Odeslání přihlašovacích údajů**: Uživatel odesílá na server požadavek obsahující své uživatelské jméno a heslo.

2. **Ověření údajů**: Server ověřuje, zda zadané přihlašovací údaje odpovídají údajům uloženým v databázi. Provádí se porovnání zahashovaného hesla z databáze se zahashovaným heslem, které uživatel poskytl.

3. **Generování JWT**: Pokud jsou přihlašovací údaje platné a odpovídají údajům v databázi, server generuje a poskytuje uživateli access a refresh tokeny. Pokud jsou po-

skytnuté přihlašovací údaje neplatné nebo se neshodují s údaji v databázi, uživateli je vrácena chybová zpráva a nepovoluje se mu přístup do systému.

3.1.3.2 JWT

V předchozí kapitole v sekci 2.6 byl popsán princip použití JWT. Nyní bude popsána jeho implementace v backendové části.

Pro konfiguraci JWT v backendové části aplikace je nejprve nutné nastavit parametry JWT v konfiguračním souboru `application.conf`. Tyto parametry zahrnují: `issuer`, `audience`, `expiresIn`, `refreshIn`, `realm`, `secret`. Je důležité, aby tajný klíč `secret` byl pečlivě chráněn, protože na něm závisí integrita a autentičnost JWT. Pokud by tajný klíč unikl, útočník by mohl podepisovat falešné JWT a získávat neoprávněný přístup k aplikaci.

Standardně je doba platnosti access tokenu `expiresIn` nastavena na 1 hodinu a doba platnosti refresh tokenu `refreshIn` na 24 hodin. Tato konfigurace umožňuje uživatelům udržet svou přihlášenou identitu po dlouhou dobu a zároveň minimalizuje riziko zneužití dlouhodobě platného JWT. Také v aplikaci je implementována služba pro generování a ověřování access a refresh tokenů tokenů.

Když uživatel provádí požadavky na server, předává JWT v hlavičce požadavku. Server pak ověřuje platnost tohoto tokenu a pokud je platný, umožňuje přístup k chráněným zdrojům systému. Pokud je JWT neplatný nebo vypršela jeho platnost, server odmítne požadavek.

Pro obnovení access a refresh tokenu je třeba odeslat platný refresh token na API endpoint `@POST /auth/refresh`.

Tímto způsobem je implementováno použití JWT v backendové části aplikace, což umožňuje bezpečnou autentizaci uživatelů a zajištění přístupu k chráněným prostředkům systému.

3.1.4 Databáze

V této sekci bude podrobněji prozkoumáno připojení a práce s daty v databázi. Jak již bylo dříve zmíněno, bude použito PostgreSQL s frameworkem Exposed.

3.1.4.1 Připojování k databázi

Pro připojení k databázi je nutné provést konfiguraci v souboru `application.conf`. Nastavení databáze zahrnuje následující parametry: `name`, `host`, `port`, `user`, `password`. Tato data mohou být uložena jako proměnné prostředí na serveru, což je bezpečnější alternativa než pevně zapsat do konfiguračního souboru. V konfiguračním souboru lze pak použít zástupné proměnné, které odkazují na hodnoty těchto proměnných prostředí. Například:

```
database {  name = ${?DB_NAME}
            host = ${?DB_HOST}
            port = ${?DB_PORT}
            user = ${?DB_USER}
            password = ${?DB_PASSWORD}  }
```

Pro připojení k databázi a vytvoření potřebných tabulek je použít následující kód 3.4.

```
object DatabaseSingleton {
    fun init(databaseConfig: DatabaseConfig) {

        val database = Database.connect(
        url = databaseConfig.url,
        driver = databaseConfig.driver,
        user = databaseConfig.user,
        password = databaseConfig.password
        )

        transaction(database) {
            SchemaUtils.create(Users, Incidents,
            Locations, ArchiveIncidents, ArchiveLocations)
        }
    }

    suspend fun <T> dbQuery(block: suspend () -> T): T
    = newSuspendedTransaction(Dispatchers.IO) { block() }
}
```

Výpis kódu 3.4 Připojování k databázi

3.1.4.2 DAO

Data Access Object (DAO) slouží k oddělení kódu, který pracuje s databází, od zbytku aplikace. V aplikaci jsou implementována následující DAO pro práci s daty:

1. **UserDao:** Obsahuje metody pro manipulaci s tabulkou Users, umožňuje provádět operace CRUD (create, read, update, delete) nad uživateli.

2. **IncidentDao:** Poskytuje funkce pro manipulaci s tabulkou Incidents, umožňuje číst všechny incidenty, incidenty podle uživatelského ID, vytvářet nové incidenty a aktualizovat poslední umístění incidentu.

3. **LocationDao:** Obsahuje metody pro manipulaci s tabulkou Locations, umožňuje provádět operace CRUD nad lokalitami, a také vyhledávání incidentů.

4. **ArchiveIncidentDao:** Poskytuje funkce pro manipulaci s archivovanými daty incidentů v tabulce ArchiveIncidents. Toto DAO využito pro archivaci ukončených incidentů a poskytuje podobné funkce jako IncidentDao.

5. **ArchiveLocationDao:** Podobně jako ArchiveIncidentDao, tento DAO poskytuje funkce pro manipulaci s archivovanými daty lokalit v tabulce ArchiveLocations.

Funkce v rozhraních DAO jsou deklarovány s klíčovým slovem `suspend`, což indikuje, že jsou připraveny pracovat s asynchronními operacemi. Například funkce `suspend fun readAll(): List<User>` v rozhraní UserDao značí, že metoda `readAll()` může být volána asynchronně a vrátí seznam uživatelů. Funkce `suspend fun <T> dbQuery(block: suspend () -> T): T}` z ukázky kodu 3.4, je pomocná funkce, která obaluje kód provádějící dotaz do databáze a zajišťuje, že je spuštěn v asynchronním kontextu.

Ukázka rozhraní UserDao pro manipulaci s databází 3.5.

```
interface UserDao {
    suspend fun readAll(): List<User>
    suspend fun read(id: Long): User?
    suspend fun readByUsername(username: String): User?
    suspend fun delete(id: Long): Boolean
    suspend fun create(
        username: String, password: String, salt: String
    ): User?
}
```

Výpis kódu 3.5 Ukázka rozhraní UserDao pro manipulaci s databází

3.1.5 Výpočet vzdálenosti mezi vozidly

Endpoint `@GET /api/v1/incidents/?latitude={val}&longitude={val}` slouží k získání všech incidentů v okolí určené polohy, definované pomocí parametrů `latitude` a `longitude`. Avšak, je důležité odesílat pouze incidenty, které jsou skutečně v blízkosti, aby se zabránilo zbytečné zátěži při vykreslování v aplikaci LocTracker a snížila se velikost přenášených dat. Je proto nezbytné určit, jaká vzdálenost je pro tyto účely vhodná.

Vozidla nejrychleji přibližují, když jedou proti sobě. Proto lze přibližně vypočítat vzdálenost, aby řidič měl alespoň 10 sekund na reakci.

- Vzorec pro výpočet (vozidla jedou proti sobě):

$$\text{Vzdálenost} = (\text{rychlost 1. vozidla} + \text{rychlost 2. vozidla}) \times \frac{10s}{3600s} \tag{3.1}$$

Vzhledem k tomu, že nejvyšší povolená rychlost na silnicích v České republice činí 130 km/h, tato hodnota byla použita pro obě vozidla. Pak vychází z toho, že požadovaná vzdálenost mezi nimi je přibližně 0,72 km. Pro aplikaci by však bylo vhodné zvolit vzdálenost mírně vyšší než 0,72 km, například 1 km, aby byla zajištěna dostatečná rezerva.

Také je potřeba zvolit vhodný vzorec pro výpočet vzdálenosti mezi dvěma body na mapě. Jelikož Země má kulatý tvar, přímý výpočet vzdálenosti mezi dvěma body na mapě není možný. Proto je nezbytné použít speciální vzorec, který bere v úvahu zakřivení Země, jako je například goniometrická Haversinova formule. Zde je vidět, jak

```kotlin
// /dao/locations/LocationDaoImpl.kt
private fun calculateDistance(
    latitude1: Double, longitude1: Double,
    latitude2: Double, longitude2: Double
): Double {
    val earthRadius = 6371
    val latDistanceRadians =
        Math.toRadians(latitude2 - latitude1)
    val lonDistanceRadians =
        Math.toRadians(longitude2 - longitude1)
    val a = sin(latDistanceRadians / 2) *
            sin(latDistanceRadians / 2) +
            cos(Math.toRadians(latitude1)) *
            cos(Math.toRadians(latitude2)) *
            sin(lonDistanceRadians / 2) *
            sin(lonDistanceRadians / 2)
    val centralAngle = 2 * atan2(sqrt(a), sqrt(1 - a))
    return earthRadius * centralAngle
}
```

Výpis kódu 3.6 Implementace výpočtu vzdálenosti mezi dvěma body na Zemi pomocí formule Haversine

vypadá výpočet v kódu: 3.6. Tato metoda vrací vzdálenost mezi dvěma body v kilometrech, pak endpoint, který vrací všechny incidenty v okolí, bude vracet všechny incidenty v poloměru 1 km.

3.2 Mobilní aplikace

3.2.1 Struktura

3.2.1.1 Základ

Při vytváření nových fragmentů, viewmodelů a repozitáře je vhodné, aby nové třídy dědily od odpovídajících základních abstraktních tříd. To umožní udržet konzistenci v implementaci a správě dat v celé aplikaci a zajistí dodržení správné architektury aplikace. Proto v aplikaci jsou definovány následující klíčové prvky:

- **BaseFragment**: Tato abstraktní třída slouží jako základ pro fragmenty v aplikaci. Poskytuje společnou funkcionalitu, jako je manipulace s ViewBinding pro správu uživatelského rozhraní, inicializace ViewModel pro logiku a data fragmentu.
- **BaseViewModel**: Abstraktní třída pro ViewModels v aplikaci. Tato třída poskytuje společný základ pro všechny ViewModels a drží odkaz na BaseRepository.
- **BaseRepository**: Abstraktní třída funguje jako prostředník mezi aplikací a datovou vrstvou, zahrnující jak síťová volání, tak i manipulaci s lokálními daty na mobilním

zařízení. Tímto způsobem poskytuje rozhraní pro získávání a ukládání dat, ať už ze vzdálených serverů nebo z lokálního úložiště zařízení.

Obecná struktura aplikací LocShare a LocTracker:

- `/loc_{share/tracker}` — Kořenová složka projektu
  - `app` — Modul app
    - `build` — Složka obsahující sestavené soubory a dokumentaci
    - `src` — Zdrojové kódy projektu
      - `androidTest` — Instrumentační testy
      - `main` — Hlavní zdrojové soubory
        - `java` — Zdrojové kódy Java
        - `res` — Resource soubory
        - `AndroidManifest.xml` — Manifest soubor
      - `test` — Testy
    - `build.gradle.kts` — Soubor s konfigurací Gradle pro modul app
  - `gradle` — Složka s konfigurací Gradle
    - `wrapper` — Složka obsahující konfiguraci Gradle wrapperu
    - `libs.versions.toml` — Konfigurační soubor verzí
  - `build.gradle.kts` — Soubor s konfigurací Gradle pro projekt
  - `gradle.properties` — Konfigurační soubor Gradle
  - `gradlew` — Spustitelný soubor Gradle wrapperu pro Unix
  - `gradlew.bat` — Spustitelný soubor Gradle wrapperu pro Windows
  - `README.md` — Soubor obsahující základní informace o projektu
  - `settings.gradle.kts` — Soubor s nastavením projektu pro Gradle

3.2.1.2 LocTracker

Struktura /src/main:

- `/src/main` — Hlavní zdrojové soubory
  - `java/cz/cvut/fit/poberboh/loc_tracker`
    - `network` — Složka pro síťovou komunikaci
      - `api` — Složka obsahující MapApi rozhraní
      - `responses` — Složka s odpověďmi z API
    - `repository` — Složka obsahující repozitáře pro práci s daty
    - `ui` — Složka obsahující uživatelské rozhraní
      - `base` — Abstraktní třídy BaseFragment a BaseViewModel
      - `basic` — Fragmenty, viewmodely pro start. obrazovku a mapu
    - `MainActivity.kt` — Hlavní aktivita aplikace LocTracker
  - `res` — Resource soubory
    - `drawable` — Obsahuje obrázky, pozadí a různé grafické prvky
    - `layout` — XML soubory s popisem rozložení obrazovek
    - `mipmap` — Ikony různých velikostí
    - `navigation` — Nastavení navigace v aplikaci
    - `values` — Různé hodnoty, například textové řetězce, barvy
  - `AndroidManifest.xml` — Soubor obsahující popis aplikace

Hlavním vstupním bodem aplikace LocTracker je MainActivity, kde se vyžaduje udělení oprávnění k přístupu k poloze a typu sítě. Pokud uživatel neposkytne oprávnění k poloze, aplikace se uzavře. Hlavními fragmenty v aplikaci jsou StartFragment a MapFragment. Aplikace začíná na StartFragmentu, kde po stisknutí tlačítka `Start` proběhne přepnutí na fragment MapFragment. Na mapě je zobrazeno pohybování uživatele a aktivní incidenty. V následujících částech této kapitoly bude podrobněji zkoumáno odesílání požadavků a celkové chování aplikace.

3.2.1.3 LocShare

Struktura /src/main:

- `/src/main` — Hlavní zdrojové soubory
  - `java/cz/cvut/fit/poberboh/loc_share`
    - `data/AppPreferences.kt` — Třída pro správu nastavení aplikace
    - `network` — Složka pro síťovou komunikaci
      - `api` — Složka obsahující API rozhraní
      - `requests` — Složka obsahující definice API požadavků
      - `responses` — Složka s odpověďmi z API
    - `repository` — Složka obsahující repozitáře pro práci s daty
    - `ui` — Složka obsahující uživatelské rozhraní
      - `auth` — Fragmenty, viewmodely pro registraci a přihlašení
      - `base` — Abstraktní třídy BaseFragment a BaseViewModel
      - `home` — Fragmenty, viewmodely pro hlavní obrazovku aplikace
    - `utils` — Složka obsahující pomocné utility pro aplikaci
    - `AuthActivity.kt` — Aktivita pro autentizaci uživatelů
    - `HomeActivity.kt` — Aktivita pro zobrazení hlavní obrazovky aplikace
    - `MainActivity.kt` — Hlavní vstupní bod do aplikace
  - `res` — Resource soubory
    - `drawable` — Obsahuje obrázky, pozadí a různé grafické prvky
    - `layout` — XML soubory s popisem rozložení obrazovek
    - `menu` — Menu s definicemi přepínání mezi fragmenty
    - `mipmap` — Ikony různých velikostí
    - `navigation` — Nastavení navigace v aplikaci
    - `values` — Různé hodnoty, například textové řetězce, barvy
  - `AndroidManifest.xml` — Soubor obsahující popis aplikace

Tato aplikace se skládá ze tří hlavních aktivit. První z nich je MainActivity, která je vstupním bodem do aplikace a vyžaduje povolení k přístupu k poloze. Pokud uživatel toto povolení neposkytne, aplikace se uzavře. Poté je kontrolováno, zda je uložený access token k dispozici. Pokud ano, otevře se HomeActivity, pokud ne, otevře se AuthActivity. V AuthActivity je nejprve zobrazen LoginFragment, kde uživatel může buď přihlásit se do hlavní části aplikace, nebo přepnout na RegisterFragment a provést registraci. Přepínání mezi těmito fragmenty se provádí pomocí tlačítek `Zaregistrovat se` a `Přihlásit se`. Po úspěšném přihlášení se zobrazí HomeActivity, kde je zpočátku zobrazen HomeFragment, který slouží k aktivaci odesílání polohy na server. Na dolní části obrazovky je navigační panel, pomocí kterého můžeme přepínat mezi HomeFrag-

ment a MapFragment, tedy mezi zobrazením domovské obrazovky a mapy, na mapě je zobrazen pouze pohyb uživatele. Detailní popis funkcionality jednotlivých částí aplikace bude poskytnuto v dalších částech této kapitoly.

3.2.2 Autentizace v aplikaci LocShare

V aplikaci LocShare je autentizace zajištěna pomocí následujících prvků:

- **AppInterceptor:** Tento interceptor přidává hlavičku autorizace s access tokenem ke každému požadavku. Používá se k autentizaci uživatele při provádění požadavků na API.
- **AppAuthenticator:** Tato třída se používá k obnovení access tokenu, pokud je aktuální token neplatný. Automaticky obnovuje token, pokud server vrátí chybu 401 (Unauthorized).
- **AppPreferences:** Tato třída je zodpovědná za správu nastavení aplikace, včetně ukládání, odstraňování access a refresh tokenů.
- **AuthApi**: Toto rozhraní poskytuje metody pro autentizaci uživatele, jako je registrace nového uživatele a přihlášení uživatele.
- **RefreshApi:** Toto rozhraní definuje metodu pro obnovení access a refresh tokenů.

3.2.2.1 Přihlášení a registrace

- **Registrace:** Uživatel vyplní registrační formulář, který obsahuje požadované informace, jako je uživatelské jméno a heslo. Tyto údaje jsou odeslány na server, kde je nový účet vytvořen a uložen do databáze.
- **Přihlášení:** Uživatel vyplní přihlašovací formulář s registračními údaji. Tyto údaje jsou odeslány na server, kde jsou ověřeny. Pokud jsou údaje platné, server vrátí platný access a refresh tokeny, který identifikuje uživatele a poskytuje jim přístup k chráněným prostředkům na serveru.

Při spuštění aplikace AuthActivity zajišťuje inicializaci uživatelského rozhraní pro přihlašování a registraci. Toto rozhraní zahrnuje dvě hlavní části: LoginFragment pro přihlašování uživatele a RegisterFragment pro registraci nových uživatelů.

- **LoginFragment:** Tato část obsahuje formulář pro zadání uživatelského jména a hesla. Po zadání údajů může uživatel kliknout na tlačítko `Přihlásit se`, což spustí proces ověřování. LoginFragment komunikuje s AuthViewModel, který je zodpovědný za provedení ověření a komunikaci s repozitářem pro přístup k datům. Po úspěšném ověření AuthViewModel obdrží access a refresh tokeny a tyto tokeny jsou uloženy do úložiště. Poté je uživatel přesměrován na hlavní obrazovku aplikace.
- **RegisterFragment:** Tato část umožňuje novým uživatelům zaregistrovat se do aplikace. Uživatel vyplní formulář s požadovanými údaji a klikne na tlačítko `Zaregistrovat se`. Fragment komunikuje s AuthViewModel, který ověřuje zadané údaje a předává je repozitáři pro uložení do databáze. Po úspěšné registraci se uživatel přesměruje na přihlašovací obrazovku, kde se může přihlásit se svým nově vytvořeným účtem.

Tímto způsobem spolupracují AuthActivity, LoginFragment, RegisterFragment a AuthViewModel k umožnění uživatelům registrace a přihlašování do aplikace.

3.2.3 API

Pro komunikaci s backendem byla zvolena knihovna Retrofit. Retrofit poskytuje jednoduché a efektivní možnosti definování API rozhraní pomocí anotací, což usnadňuje práci vývojářům a zvyšuje čitelnost kódu. Navíc Retrofit automaticky konvertuje JSON data na odpovídající objekty v jazyce Kotlin [32]. Dobře dokumentovaná knihovna s širokou podporou v komunitě.

Následující komponenty jsou používány jak v aplikaci LocShare, tak v aplikaci LocTracker:

- **RemoteDataSource:** Třída RemoteDataSource je zodpovědná za vytváření a odesílání požadavků na API pomocí Retrofitu. Obsahuje metody pro vytvoření instance API. Její účel spočívá v abstrakci komunikace s backendem a poskytování jednotného rozhraní pro práci s API.
- **Resource:** Třída Resource představuje výsledek síťového volání. Může obsahovat úspěšný výsledek nebo síťovou chybu. Tato třída usnadňuje zpracování odpovědí ze serveru. Jejím cílem je umožnit konzistentní zpracování dat v celé aplikaci.

Níže jsou uvedeny hlavní prvky, které tvoří systém komunikace v aplikaci LocShare:

- **BasicApi:** Toto rozhraní definuje základní metody pro komunikaci s API, jako je získání jména uživatele, vytvoření incidentu, záznam polohy uživatele a zastavení sdílení polohy.
- **RefreshApi:** Toto rozhraní definuje metodu pro obnovení access a refresh tokenů.
- **AuthApi**: Toto rozhraní poskytuje metody pro autentizaci uživatele, jako je registrace nového uživatele a přihlášení uživatele.

Když klientská aplikace odesílá požadavek na API, RemoteDataSource používá Retrofit k vytvoření a odeslání požadavku s veškerými potřebnými nastaveními, jako jsou URL, hlavičky a parametry požadavku. AppInterceptor přidává hlavičku autorizace ke každému požadavku, používá aktuální access token z AppPreferences. Pokud server vrátí chybu 401 (Unauthorized), Authenticator automaticky obnoví access token a znovu odešle požadavek s aktualizovaným tokenem. V důsledku toho klientská aplikace může komunikovat s API bez potřeby manuálně obnovovat access token.

Níže je uvedena hlavní komponenta API v aplikaci LocTracker:

- **MapApi**: Toto rozhraní definuje metody pro získání dat o incidentech v blízkosti zadaných souřadnic zeměpisné šířky a délky.

Stejně jako v aplikaci LocShare při provádění dotazů na API třída RemoteDataSource používá Retrofit k vytvoření a odeslání požadavků s nezbytnými parametry, jako jsou URL a parametry dotazu.

3.2.4 Mapy

V obou aplikacích používáme OsmAnd SDK pro práci s mapami.
Pomocí `GpsMyLocationProvider` a `MyLocationNewOverlay` z knihovny OsmAnd zobrazuje se pohyb uživatele na mapě.

`GpsMyLocationProvider`: Tento poskytovatel lokalizace je zodpovědný za získávání aktuální polohy uživatele pomocí GPS signálu. Jeho hlavním účelem je poskytovat přesnou polohu uživatele na základě signálů z GPS zařízení.

`MyLocationNewOverlay`: Tento overlay slouží k zobrazení aktuální polohy uživatele na mapě. V podstatě se jedná o grafickou reprezentaci polohy, která se zobrazuje na mapě v reálném čase. `MyLocationNewOverlay` umožňuje zobrazit ikonku, která reprezentuje polohu uživatele na mapě a sleduje jeho pohyb.

Kromě současných funkcí aplikací by bylo možné zvážit implementaci funkcionality pro přednačítání mapy pro určitý region, čímž by se zkrátil čas načítání mapy při spuštění aplikace. Nicméně, tato funkcionalita by mohla zvýšit velikost aplikace na mobilním zařízení, protože by vyžadovala ukládání přednačtených mapových dat. Je důležité poznamenat, že i bez této funkcionality mapa stále funguje a uživatelé mohou interagovat s aplikací.

3.2.4.1 Mapa v aplikaci LocTracker

V aplikaci LocTracker slouží `MainActivity` jako vstupní bod do aplikace. Po spuštění se vytváří instance `MapFragment`, která zobrazuje mapu a interaguje s ní. `MapFragment` načítá incidenty pomocí třídy `MapViewModel` a zobrazuje je na mapě metodou `displayIncidentsOnMap()`. Aktuální polohy incidentů jsou získávány každou sekundu dotazem na server a následně zobrazovány na mapě ve formě červených vozidel. Tento proces umožňuje uživatelům sledovat směr a vzdálenost incidentů od jejich aktuální pozice v aplikaci LocTracker.

3.2.4.2 Mapa v aplikaci LocShare

V aplikaci LocShare slouží `HomeActivity` jako vstupní bod do aplikace.
`LocationFragment` je abstraktní fragment, který obsahuje mapu a zajišťuje sledování polohy uživatele. Mapa je načtena na úrovni tohoto fragmentu, aby bylo možné sledovat pohyb uživatele na všech fragmentech aplikace, nejen na `MapFragment`. `HomeFragment` je fragment, který zobrazuje hlavní obsah aplikace. V tomto fragmentu je mapa skryta, aby uživatelé mohli interagovat s ostatními funkcemi aplikace. `MapFragment` je fragment, který zobrazuje mapu a umožňuje uživatelům procházet mapu a interagovat s ní. V aplikaci LocShare data o poloze uživatele odesílají na server každou sekundu poté, co je aktivována funkce sdílení polohy stisknutím tlačítka `Aktivovat`.

Pomocí navigačního panelu v dolní části obrazovky je možné snadno přepínat mezi jednotlivými fragmenty aplikace, což umožňuje pohodlnou navigaci a používání funkcí aplikace LocShare.

3.2.5 Uživatelské rozhraní

V rámci této práce byl návrh uživatelského rozhraní proveden pouze pro vertikální orientaci mobilních zařízení. V dalším vývoji může být rozšířena implementace o design vhodný pro horizontální orientaci, což přispěje k lepší uživatelské zkušenosti.

Zde jsou soubory, které jsou definovány v `res/drawable` a `res/layout` a jsou použity pro tvorbu uživatelského rozhraní v aktivitách a fragmentech:

1. **LocShare:**
   - **button_background.xml:** Tento soubor obsahuje popis selektoru, který určuje vzhled pozadí pro tlačítka, stanovující hlavní barvu pro tlačítka v aplikaci.
   - **gradient_background.xml:** Tento soubor obsahuje popis vrstvy, která vytváří gradientní pozadí. Gradientní pozadí má plynulý přechod mezi několika barvami.
   - **twotone_5g.xml:** Tento soubor obsahuje vektorovou grafiku symbolizující technologii 5G.
   - **activity_home.xml:** Tento layout je používán pro hlavní obrazovku aplikace. Obsahuje mapu a dolní navigační panel.
   - **fragment_home.xml:** Tento layout slouží pro zobrazení fragmentu na hlavní obrazovce. Obsahuje prvky uživatelského rozhraní, jako jsou textová pole, tlačítka.
   - **activity_auth.xml:** Tento layout slouží pro obrazovku autentizace uživatele. Obsahuje prvky uživatelského rozhraní, jako jsou textová pole a tlačítka, pro zadávání přihlašovacích údajů.
   - **fragment_login.xml:** Tento layout je používán pro obrazovku přihlášení uživatele. Obsahuje textová pole pro zadání uživatelského jména a hesla, stejně jako tlačítka pro přihlášení a registraci nového uživatele.
   - **fragment_register.xml:** Tento layout je používán pro obrazovku registrace nového uživatele. Obsahuje textová pole pro zadání uživatelského jména, hesla a opakování hesla, stejně jako tlačítka pro odeslání žádosti o registraci.
2. **LocTracker:**
   - **button_background.xml:** Tento soubor obsahuje popis selektoru, který určuje vzhled pozadí pro tlačítka, stanovující hlavní barvu pro tlačítka v aplikaci.
   - **gradient_background.xml:** Tento soubor obsahuje popis vrstvy, která vytváří gradientní pozadí. Gradientní pozadí má plynulý přechod mezi několika barvami.
   - **activity_main.xml:** Tento layout je používán pro hlavní obrazovku aplikace.
   - **fragment_map.xml:** Tento layout slouží pro zobrazení mapy v rámci fragmentu.
   - **fragment_start.xml:** Tento layout je používán pro zobrazení úvodního fragmentu. Obsahuje tlačítko pro spuštění aplikace.
   - **twotone_5g.xml:** Tento soubor obsahuje vektorovou grafiku symbolizující technologii 5G.
   - **incident_24.xml:** Tento soubor obsahuje vektorovou grafiku znázorňující symbolické zobrazení incidentu, které může být zobrazeno na mapě. Grafika představuje červené vozidlo.

Kapitola 4
Testování a dokumentace

4.1 Testování

V této sekci bude popsáno testování aplikací.

4.1.1 Testování autorem

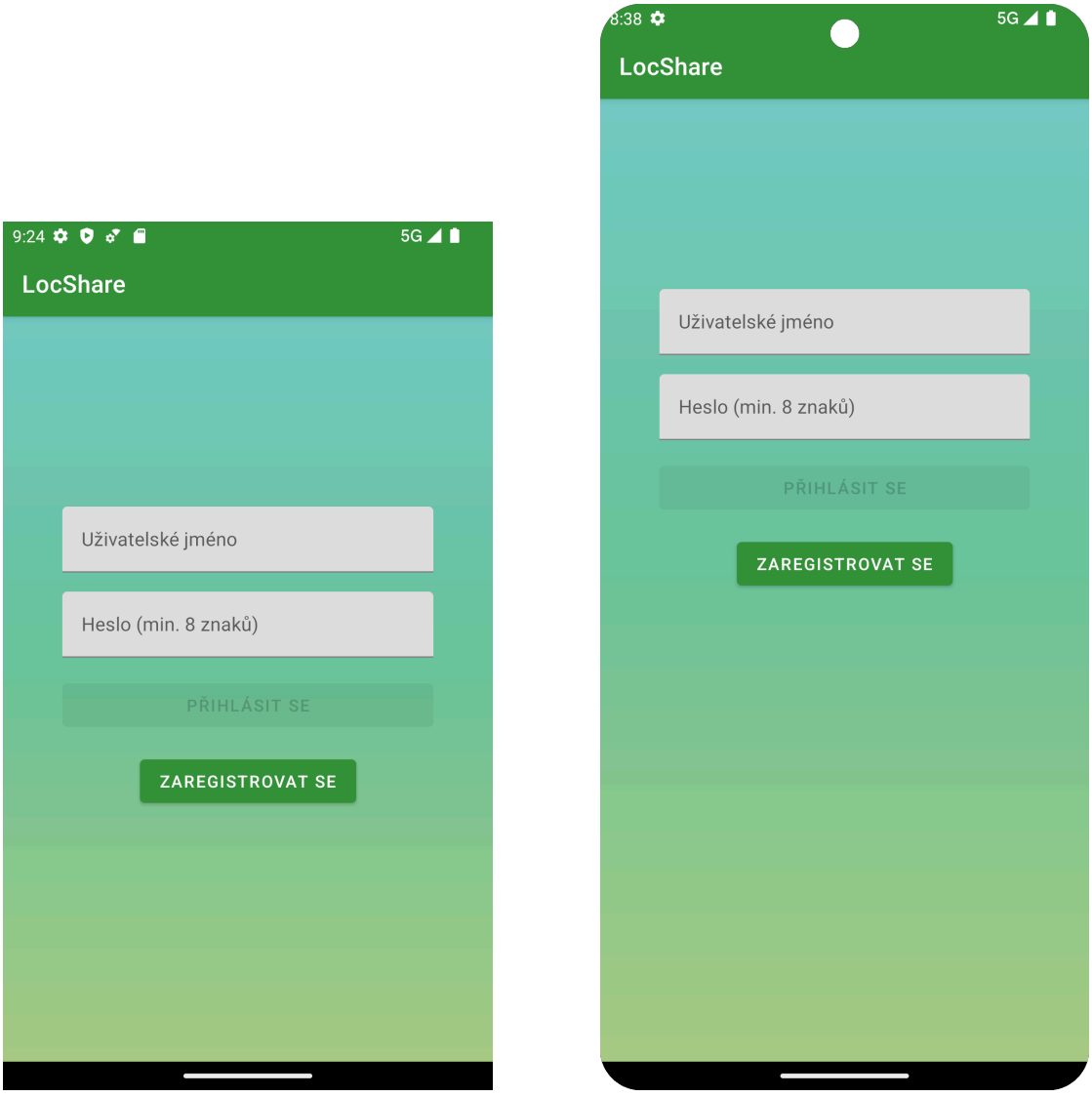
Pro testování, které by simulovalo reálné použití, byla spuštěna backendová část na virtuálním serveru a byla zajištěna databáze. V mobilních aplikacích LocShare a LocTracker byla v konfiguraci nastavena adresa URL na server, na kterou aplikace posílají své požadavky, tak, aby odpovídala adrese backendové části nasazené na serveru. Poté byly obě aplikace nainstalovány na fyzický mobilní telefon a také testovány na různých emulátorech dostupných v Android Studiu. Tento postup umožnil testování aplikací na zařízeních s různými verzemi operačního systému Android a různými rozměry displejů. Testování zahrnovalo mobilní telefony s úhlopříčkou obrazovky od 5 palců až po telefony s úhlopříčkou obrazovky přesahující 6 palců, stejně jako tablety s úhlopříčkou obrazovky přesahující 10 palců. Příklad dvou různých emulátorů telefonů na obrázku 4.1. Ve všech případech byla zjištěna stabilní funkčnost aplikací a všechny funkcionality byly dostupné.

Také během vývoje byl testován backend nezávisle na mobilních aplikacích. Byla vytvořena lokální databáze, ke které se připojoval backend, běžící rovněž lokálně na počítači. Po implementaci nových endpointů bylo možné je testovat pomocí požadavků v Postman. Pro efektivní správu vývoje byl využíván IntelliJ IDEA, který je pohodlným vývojovým prostředím pro práci s Kotlinem. Poskytoval pohodlný způsob sledování změn v databázi, prohlížení logu, sestavování a spouštění backend aplikace.

Během vývoje také probíhalo lokální testování mobilních aplikací LocTracker a LocShare. Tyto aplikace byly spuštěny na fyzickém mobilním telefonu a emulátorech pomocí Android Studia, které je známé jako velmi užitečné a efektivní prostředí pro vývoj Android aplikací. Bylo možné detailněji prozkoumat uživatelské rozhraní a zlepšit barevnou paletu a design, aby bylo uživatelsky přívětivější. Dále bylo možné pomocí logů sledovat správný tok dat v aplikaci, což napomohlo k identifikaci a opravě případných problémů.

Než byla backendová část nasazena na virtuální server pro účely testování, celý softwarový systém byl důkladně otestován jako celek na lokálním prostředí. Na lokálním

zařízení byla spuštěna backendová část aplikace a v její konfiguraci byla určena jako host IP adresa tohoto zařízení. V konfiguraci mobilních aplikací byla adresa URL pro požadavky nastavena na tuto IP adresu, což umožnilo testování v rámci jedné sítě, kde všechna zařízení byla připojena k témuž routeru. Tímto způsobem bylo možné simulovat běh aplikace na více zařízeních. Například na fyzickém mobilním telefonu bylo spuštěno aplikace LocTracker a na emulátoru běželo LocShare, a celkově vše fungovalo správně.



Obrázek 4.1 Porovnání úvodní obrazovky aplikace LocShare na telefonu s 5,0 a 6,17-palcovým displejem

4.1.2 Testy

4.1.2.1 Backend

Pro backend byly vytvořeny unit testy, které testují konkrétní služby. Hlavně byly testovány služby spojené s generováním JWT tokenů a hashováním hesel. Například, zda se správně vytvářejí access a refresh tokeny, správně funguje časový limit tokenu, a zda lze vytvořit tokeny s prázdným `secret`. Unit testy pro hashování zahrnují ověření správného generování soli a hashování hesla, kontrolu správné funkčnosti ověření hesla, zajištění unikátnosti soli a hashů pro různá hesla, identifikaci neplatných hesel a správné zpracování prázdného a dlouhého řetězce při hashování.

Pro spuštění těchto testů pomocí Gradle je možné použít příkaz:

```
./gradlew test
```

Tento příkaz spustí všechny unit testy v projektu a zobrazí výsledky. Příkaz je nutné spustit v kořenovém adresáři projektu.

4.1.2.2 Mobilní aplikace

Pro testování mobilních aplikací byly vytvořeny unit testy a instrumentační testy.

V aplikaci LocTracker jsou během provádění unit testů testovány spolehlivost různých komponent, jako je zpracování úspěšných volání metod, zpracování různých typů chyb při komunikaci s API a správnost vytváření prostředků vrácených z vzdáleného datového zdroje. Testy jsou zaměřeny na zajištění správného chování metod a ujištění se o korektním fungování aplikace v různých scénářích použití. V instrumentačních testech aplikace se ověřuje správnost zobrazení ikony 5G po aktualizaci sítě na 5G a změna viditelnosti této ikony při přepínání z 2G na 5G. Dále se testuje správnost navigace mezi fragmenty. Testuje se také, zda při spuštění navigace na mapě dochází k očekávanému přechodu na obrazovku s mapou. Tyto testy zajišťují, že uživatelské rozhraní aplikace je korektně vykresleno a že interakce s ním probíhají tak, jak je očekáváno.

V unit testech v LocShare se ověřuje správná funkčnost různých částí aplikace. Testy zahrnují autentizaci aplikace, komunikaci s API a správnost funkcí úložiště. Každý test se zaměřuje na konkrétní funkcionalitu a zajišťuje, že aplikace pracuje spolehlivě a korektně reaguje na různé situace a vstupy. Instrumentační testy prověřují funkčnost procesu registrace a přihlašení. Každý test zahrnuje ověření zobrazení všech potřebných prvků uživatelského rozhraní, včetně tlačítek a textových polí. Testuje se správnost zobrazení fragmentů pro přihlašování a registraci, stejně jako správnost jejich layoutu. Dále jsou kontrolovány vstupní pole pro zadání uživatelského jména a hesla, a to jak z hlediska zobrazení, tak i funkčnosti. Kromě toho jsou testovány i interakce s tlačítky pro přechod mezi fragmenty.

4.2 Dokumentace

V této sekci bude popsáno dokumentování aplikací. Pro generování dokumentace přímo z komentářů v kódu byl vybrán nástroj Dokka. Dokka je vhodný pro projekty psané v jazyce Kotlin a poskytuje široké možnosti konfigurace a nastavení [33].

4.2.1 Komentáře

```
/**
 * Toto je ukázkový komentář pro Dokku.
 * Můžeme použít různé značení, například:
 * - @param pro parametry funkce
 * - @return pro návratovou hodnotu
 * - @throws pro výjimky
 */
```

Dokka je schopná interpretovat tyto komentáře a generovat z nich srozumitelnou dokumentaci. Takové komentáře pokryjí všechny nejdůležitější části kódu, které by měly být v dokumentaci.

4.2.2 Generování

Pro generování dokumentace je nutné spustit následující příkaz v terminálu ve složce s kořenovým adresářem projektu:

1. Generuje dokumentaci ve formátu HTML:

   ```
   ./gradlew dokkaHtml
   ```

2. Experimentální formáty:

   - Generuje dokumentaci ve formátu GitHub Flavored Markdown:

     ```
     ./gradlew dokkaGfm
     ```

   - Generuje dokumentaci ve formátu Javadoc:

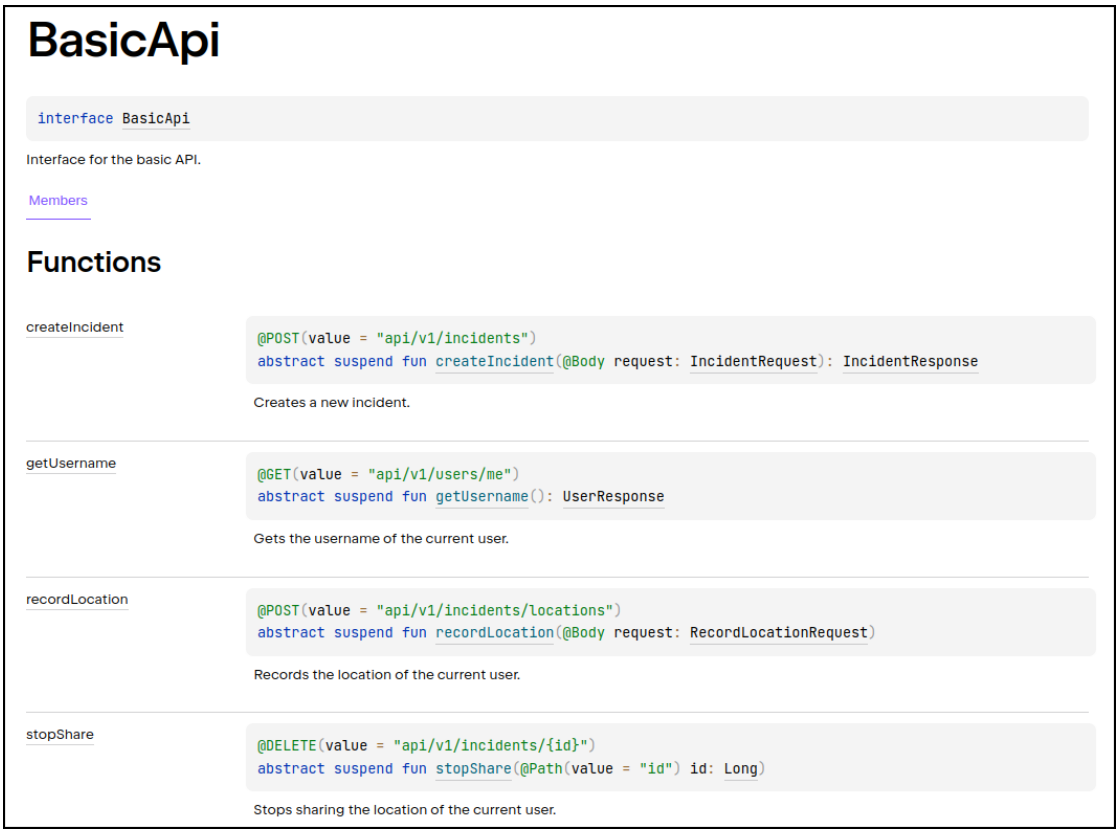
     ```
     ./gradlew dokkaJavadoc
     ```

   - Generuje dokumentaci ve formátu Jekyll Markdown:

     ```
     ./gradlew dokkaJekyll
     ```

Po spuštění tohoto příkazu je možné otevřít soubor `index.html`, který se nachází ve složce `build/dokka/{html/javadoc}`, ve webovém prohlížeči k prohlédnutí vygenerované dokumentace, nebo soubor `index.md` v adresáři `build/dokka/{gfm/jekyll}`, který slouží k prohlížení vygenerované dokumentace v Markdown formátu. Příklad z vygenerované dokumentace je vidět na obrázku 4.2.

Tento postup se týká jak backendové části, tak aplikací LocShare a LocTracker, protože každá z těchto částí má svou vlastní konfiguraci Dokka. Také každá část projektu má svůj vlastní soubor `README.md`, který obsahuje základní informace nutné k nastavení a spuštění.



BasicApi

```
interface BasicApi
```

Interface for the basic API.

Members

Functions

createIncident

```
@POST(value = "api/v1/incidents")
abstract suspend fun createIncident(@Body request: IncidentRequest): IncidentResponse
```

Creates a new incident.

getUsername

```
@GET(value = "api/v1/users/me")
abstract suspend fun getUsername(): UserResponse
```

Gets the username of the current user.

recordLocation

```
@POST(value = "api/v1/incidents/locations")
abstract suspend fun recordLocation(@Body request: RecordLocationRequest)
```

Records the location of the current user.

stopShare

```
@DELETE(value = "api/v1/incidents/{id}")
abstract suspend fun stopShare(@Path(value = "id") id: Long)
```

Stops sharing the location of the current user.

Obrázek 4.2 HTML dokumentace k rozhraní BasicApi v aplikaci LocShare

Kapitola 5
Závěr

V důsledku této práce byl úspěšně splněn hlavní cíl, kterým bylo vytvoření nástroje pro zlepšení bezpečnosti na silnicích prostřednictvím upozornění na blížící se dopravní prostředky, zejména vozidla IZS. Pro dosažení tohoto cíle byly vytvořeny dva prototypy mobilních aplikací pro operační systém Android. První aplikace, LocShare, umožňuje signalizovat přítomnost vozidla a informovat o jeho poloze, zatímco druhá aplikace, LocTracker, zobrazuje blížící se vozidla přímo na mapě v aplikaci, což umožňuje řidičům lépe reagovat na příchozí dopravní situace.

Během práce byla provedena analýza existujících řešení, na jejímž základě byly stanoveny funkční a nefunkční požadavky. Následně byl proveden návrh a implementace aplikací a backendu, který zprostředkovává komunikaci mezi aplikacemi. Tyto aplikace byly otestovány a zdokumentovány autorem práce a prokázaly svou funkčnost a shodu se specifikacemi.

Nicméně, existují určité oblasti, které lze v budoucím vývoji projektu vylepšit. Prvním z nich je zajištění spolehlivějšího registračního a přihlašovacího procesu, například prostřednictvím biometrického ověřování pro uživatele aplikace LocShare, aby bylo zabráněno neoprávněnému přístupu. Dále vzhledem k tomu, že během vývoje byly vytvářeny pouze prototypy aplikací, komunikace mezi nimi a backendem probíhala pomocí nešifrovaného protokolu HTTP. Nicméně při použití v praxi je nutné zabezpečit tuto komunikaci pomocí HTTPS, což zajišťuje šifrování dat během přenosu. Pro přechod z HTTP na HTTPS je nutné nakonfigurovat SSL/TLS certifikáty na serveru, což zabezpečí šifrovanou komunikaci. Také je nutné provést uživatelské testování s velkým počtem účastníků, aby byla ověřena stabilita systému a rychlost odezvy. S ohledem na komunikaci pomocí sítě 5G a asynchronní zpracování požadavků na backendu by však nemělo dojít k žádným problémům v této oblasti.

Celkově lze říci, že tato bakalářská práce může představovat přínos v oblasti bezpečnosti silničního provozu pomocí mobilních technologií a může sloužit jako základ pro další výzkumy a vývoj v této oblasti.

Bibliografie

1. WAZE MOBILE LTD. *Trasy, dopravní zpravodajství a upozornění na dopravní situace s Waze* [online]. 2024. [cit. 2024-04-15]. Dostupné z: `https://www.waze.com/cs/company`.
2. GOOGLE PLAY. *Waze - Waze Navigation & Live Traffic.* [Online]. 2024. [cit. 2024-04-17]. Dostupné z: `https://play.google.com/store/search?q=waze&c=apps`.
3. HAAS ALERT. *HAAS Alert Safety Cloud® | Digital Alerting for Connected Roads* [online]. 2024. [cit. 2024-04-18]. Dostupné z: `https://www.haasalert.com/`.
4. HAAS ALERT. *Safety Cloud® for Fire & EMS | HAAS Alert* [online]. 2024. [cit. 2024-04-21]. Dostupné z: `https://www.haasalert.com/safety-cloud-for-fire-ems`.
5. APPLIED INFORMATION. *Applied Information | Smart City and Connected Infrastructure Expertise* [online]. 2024. [cit. 2024-04-21]. Dostupné z: `https://appinfoinc.com/`.
6. GLANCE TRAVELSAFELY. *Glance TravelSafely - Android Apps on Google Play* [online]. 2024. [cit. 2024-04-22]. Dostupné z: `https://play.google.com/store/search?q=Glance%20TravelSafely&c=apps&hl=en&gl=US`.
7. APPLIED INFORMATION. *Connected Vehicle Solutions - Applied Information, Inc.* [Online]. 2024. [cit. 2024-04-21]. Dostupné z: `https://appinfoinc.com/solutions/connected-vehicle-solutions/`.
8. APPLIED INFORMATION. *Emergency Vehicle Preemption and Priority System - Applied Information, Inc.* [Online]. 2024. [cit. 2024-04-24]. Dostupné z: `https://appinfoinc.com/solutions/preemption-priority/`.
9. GUPTA, A.; JHA, R. K. A Survey of 5G Network: Architecture and Emerging Technologies. *IEEE Access*. 2015, roč. 3, s. 1206–1232. Dostupné z DOI: `10.1109/ACCESS.2015.2461602`.
10. EUROPEAN COMMISSION, DIRECTORATE-GENERAL FOR COMMUNICATIONS NETWORKS, CONTENT AND TECHNOLOGY. *Broadband coverage in Europe 2022 – Mapping progress towards the coverage objectives of the Digital Decade – Final report*. Publications Office of the European Union, 2023. Dostupné z DOI: `10.2759/909629`. Překlad vlastní.

11. JETBRAINS ACADEMY. *Three-tier architecture | Software architecture | Software construction | Essentials | Fundamentals | Computer science* [online]. 2024. [cit. 2024-04-24]. Dostupné z: https://hyperskill.org/learn/step/25083.
12. MANAGEMENTMANIA. *Třívrstvá architektura (Three-tier architecture) — managementmania.com* [online]. 2024. [cit. 2024-05-13]. Dostupné z: https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture.
13. JETBRAINS. *Kotlin Programming Language — kotlinlang.org* [online]. 2024. [cit. 2024-04-25]. Dostupné z: https://kotlinlang.org/.
14. SPRING BOOT. *Spring Boot — spring.io* [online]. 2024. [cit. 2024-04-25]. Dostupné z: https://spring.io/projects/spring-boot.
15. ECLIPSE FOUNDATION. *Vert.x for Kotlin | Eclipse Vert.x — vertx.io* [online]. 2024. [cit. 2024-05-08]. Dostupné z: https://vertx.io/docs/vertx-core/kotlin/.
16. ESPINA, Edgar. *jooby — jooby.io* [online]. 2024. [cit. 2024-05-08]. Dostupné z: https://jooby.io/.
17. DAVID DENTON, Ivan Sanchez. *http4k - a functional Kotlin HTTP toolkit — http4k.org* [online]. 2024. [cit. 2024-05-08]. Dostupné z: https://www.http4k.org/.
18. JETBRAINS. *Ktor: Build Asynchronous Servers and Clients in Kotlin — ktor.io* [online]. 2024. [cit. 2024-04-25]. Dostupné z: https://ktor.io/.
19. JETBRAINS. *GitHub — JetBrains/Exposed: Kotlin SQL Framework — github.com* [online]. 2024. [cit. 2024-04-25]. Dostupné z: https://github.com/JetBrains/Exposed.
20. POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL — postgresql.org* [online]. 2024. [cit. 2024-04-25]. Dostupné z: https://www.postgresql.org/.
21. KUMAR, Priyank. *Understanding MVVM Architecture in Android — medium.com* [online]. 2024. [cit. 2024-05-16]. Dostupné z: https://medium.com/swlh/understanding-mvvm-architecture-in-android-%20aa66f7e1a70b.
22. GOOGLE LLC. *Google Maps Platform Documentation | Maps SDK for Android | Google for Developers — developers.google.com* [online]. 2024. [cit. 2024-05-13]. Dostupné z: https://developers.google.com/maps/documentation/android-sdk?hl=en.
23. MAPBOX. *Maps SDK | Android Docs — docs.mapbox.com* [online]. 2024. [cit. 2024-05-13]. Dostupné z: https://docs.mapbox.com/android/maps/guides/.
24. HERE. *HERE SDK | Build Location-Based Mobile Apps | HERE — here.com* [online]. 2024. [cit. 2024-05-13]. Dostupné z: https://www.here.com/platform/here-sdk.
25. OSMAND BV. *Docs | OsmAnd — osmand.net* [online]. 2024. [cit. 2024-05-13]. Dostupné z: https://osmand.net/docs/intro.
26. OSMAND BV. *OsmAnd — github.com* [online]. 2024. [cit. 2024-05-13]. Dostupné z: https://github.com/osmandapp/.

27. LAMY, Olivier. *Maven; Welcome to Apache Maven — maven.apache.org* [online]. 2024. [cit. 2024-05-09]. Dostupné z: https://maven.apache.org/.

28. JETBRAINS. *Gradle | Gradle + Kotlin — gradle.org* [online]. 2024. [cit. 2024-05-09]. Dostupné z: https://gradle.org/kotlin/.

29. GRADLE INC. *The Groovy Plugin — docs.gradle.org* [online]. 2024. [cit. 2024-05-09]. Dostupné z: https://docs.gradle.org/current/userguide/groovy_plugin.html.

30. OKTA AUTH0.COM. *JWT.IO — jwt.io* [online]. 2024. [cit. 2024-04-25]. Dostupné z: https://jwt.io/.

31. OKTA. *Refresh access tokens | Okta Developer — developer.okta.com* [online]. 2024. [cit. 2024-05-13]. Dostupné z: https://developer.okta.com/docs/guides/refresh-tokens/main/.

32. BLOCK, INC. *Retrofit — square.github.io* [online]. 2024. [cit. 2024-05-11]. Dostupné z: https://square.github.io/retrofit/.

33. JETBRAINS. *Get started with Dokka | Kotlin — kotlinlang.org* [online]. 2024. [cit. 2024-05-04]. Dostupné z: https://kotlinlang.org/docs/dokka-get-started.html.

Obsah příloh

Zdrojové kódy implementace jsou k dispozici v následujících GitHub repozitářích:

- `Backend` – `https://github.com/bohpob/loc_backend`
- `Mobilní aplikace LocShare` – `https://github.com/bohpob/loc_share`
- `Mobilní aplikace LocTracker` – `https://github.com/bohpob/loc_tracker`

Struktura souboru attachment.zip:

- `readme.txt` ..... stručný popis obsahu média
- `doc` ..... dokumentace
  - `loc_backend/dokka`
  - `loc_share/dokka`
  - `loc_tracker/dokka`
- `exe` ..... adresář se spustitelnou formou implementace
  - `loc_share/app.apk`
  - `loc_tracker/app.apk`
- `src`
  - `implementace` ..... zdrojové kódy implementace jsou k dispozici na GitHub 5
  - `thesis.zip` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF