Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Zabezpečení a monitorování provozu automobilu s využitím Android zařízení

Plzeň 2019 Tomáš Šimandl

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 5. května 2019

Tomáš Šimandl

Poděkování

Rád bych poděkoval panu Ing. Ladislavu Pešičkovi za ochotu, pomoc a cenné rady při vedení této práce.

Abstract

Car security and log book are no longer only for companies, but also for common drivers. Most of existing systems can secure a car or create a car log book, but only a few systems can do both. These systems are based on specialized devices which connect to the car. This thesis focuses on the design and implementation of a system that is based on an Android device which will replace the specialized device. The system consists of an Android application for recording data in the car and a server which is used for managing records of the car. Security is realized by an alarm which detects intruder thanks to the device sensors and warns a user by SMS, call or e-mail. Car log book stores positions of the car in time and can depict a route of the car on the map, display a length of the route or show graphs of speed and altitude.

Abstrakt

Zabezpečení a vytváření knihy jízd automobilu není již jen určeno pro firemní účely, ale i pro běžné řidiče. Většina existujících systémů podporuje buď vytváření knihy jízd, nebo zabezpečení, ale jen pár systémů podporují obě funkcionality zároveň. Systémy jsou především založeny na specializovaném zařízení, které je trvale připojeno v automobilu. Tato práce se však zabývá vytvořením systému, jehož základem je Android aplikace, která nahradí specializované zařízení. V rámci systému je navržena a implementována serverová část pro přístup a správu naměřených dat a Android aplikace pro sběr dat. Zabezpečovací část systému je tvořena alarmem, který detekuje narušitele pomocí senzorů v mobilním telefonu a informuje uživatele pomocí SMS zprávy, hovorem nebo e-mailem. Kniha jízd ukládá polohu automobilu v čase a dokáže zobrazit jednotlivé trasy automobilu na mapě společně s údaji, jako je rychlost, nadmořská výška nebo například délka trasy.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Chytrá mobilní zařízení jsou stále populárnější a dají se využít takřka v každém aspektu našich životů. S přibývajícím zájmem a nároky uživatelů začala zařízení obsahovat stále více senzorů, díky nimž je možné detekovat stimuly z vnějšího prostředí. Úlohou každého zabezpečovacího systému je takovéto stimuly detekovat a určit, zda jsou vyvolány narušitelem či nikoliv. Téměř každý mobilní telefon s operačním systémem Android obsahuje pohybový senzor a mikrofon, díky kterým dokáže plnohodnotně nahradit běžný automobilový alarm.

Současně se zabezpečením automobilu je také zajímavé sledovat jeho provoz vytvářením knihy jízd. Mobilní telefony často disponují zabudovaným polohovým senzorem, který se dokonale hodí pro vytváření podrobné knihy jízd automobilu.

Tato práce se zabývá systémem pro zabezpečení a vytváření knihy jízd vozidla s využitím mobilního zařízení s operačním systémem Android. První část práce obsahuje seznámení s platformou Android a s přístupem k senzorům. Následuje popis vybraných existujících Android aplikací zabývajících se obdobným problémem a popis výběru funkcionalit, které budou v rámci práce implementovány. Návrh celého systému, který se skládá z Android aplikace a serverové části, předchází popisu jednotlivých částí implementace. V závěru práce se nachází popis způsobu testování aplikace.

2 Platforma Android

Android je multiplatformní open-source operační systém, který je primárně určen pro mobilní zařízení. Původně byl systém vyvíjen pouze pro chytré telefony. Časem se však rozšířil i pro tablety, chytré hodinky, televize a palubní systémy automobilů.

Android je, podle společnosti StatCounter, která se zabývá kromě jiného i analýzou využití mobilních zařízení, kterými se uživatelé připojují k internetu, aktuálně nejpopulárnější operační systém pro mobilní zařízení. Dle jejich celosvětových statistik z května roku 2017 využívá 76,5 procenta uživatelů zařízení se systémem Android. Mobilní zařízení s operačním systémem iOS od společnosti Apple využívá pouze 19 procent uživatelů. Zajímavostí je, že v Severní Americe a v Oceánii je podíl mezi mobilními zařízeními s operačním systémem Android a s operačním systémem iOS vyrovnaný. [6]

2.1 Historie

Operační systém Android nebyl vyvinut společností Google, ale byl jí odkoupen v roce 2005, dva roky po vzniku společnosti Android Inc. První chytrý telefon s operačním systémem Android na trhu byl až v roce 2008. Jednalo se o HTC Dream (G1). Větší úspěch však přišel až v roce 2009 s verzí Android 1.5 Cupcake. Od té doby vyšlo dalších 13 verzí operačního systému Android, které jsou rozepsané v tabulce 2.1.

2.2 Architektura

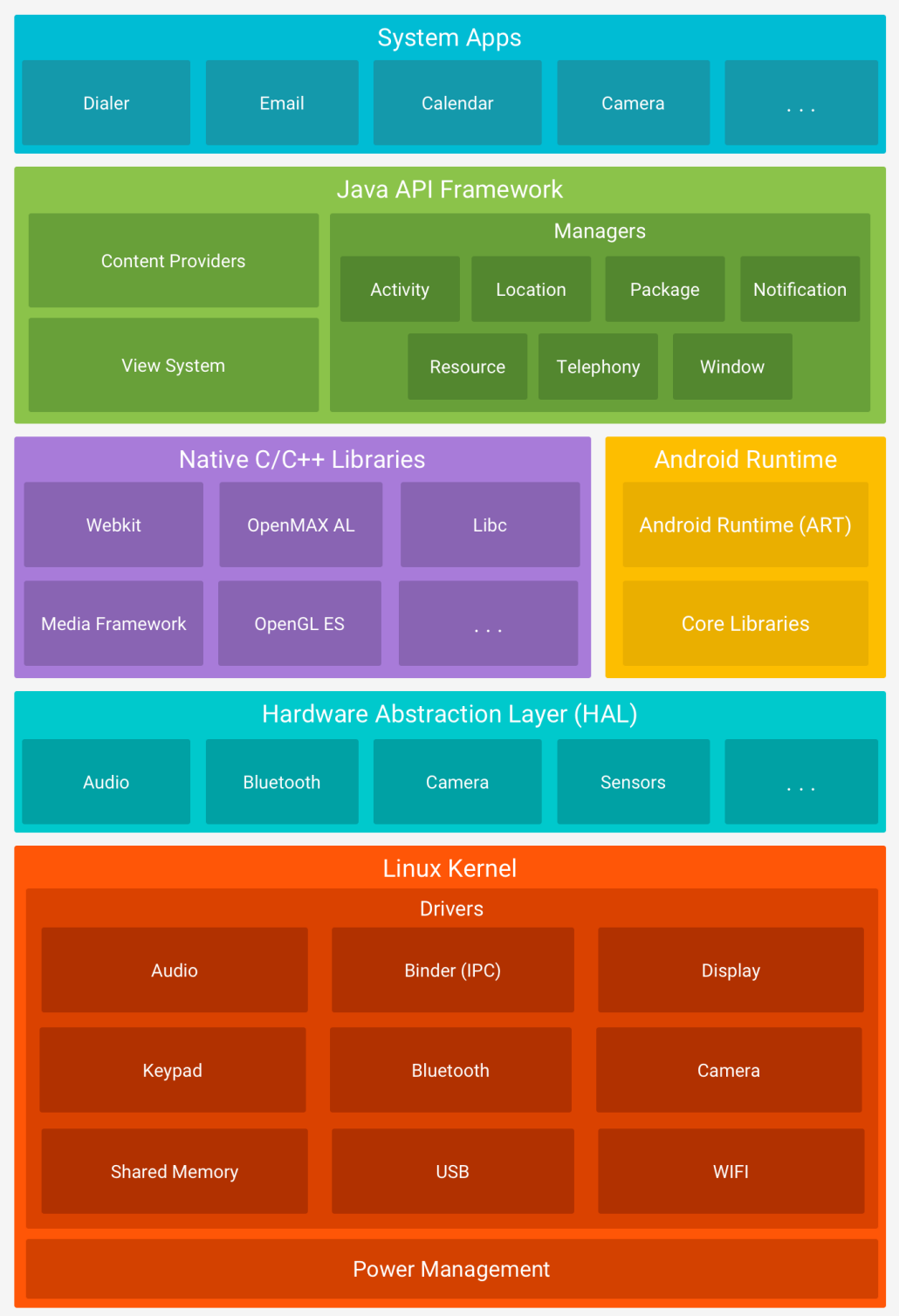
V této kapitole bude podrobně popsána architektura operačního systému Android, která je graficky znázorněna na obrázku 2.1. Popis jednotlivých komponent operačního systému Android jsou popsány dále v této kapitole.

2.2.1 Linux Kernel (Linuxové jádro)

Android je postaven na jádře operačního systému Linux, který je upraven pro potřeby mobilního zařízení. Jeho hlavní funkcí je zprostředkování komunikace mezi hardwarem a vyššími vrstvami operačního systému. Hlavní součástí jádra jsou tedy ovladače hardwaru. Kromě ovladačů obsahuje jádro i správu paměti, správu procesů či správu napájení.

Tabulka 2.1: Přehled jednotlivých verzí operačního systému Android, distribuce aktivních zařízení s danou verzí k 26.10.2018 a data vydání jednotlivých verzí [38],[9]

Verze OS	Distribuce	Datum vydání
Android 1.0 Alpha		říjen 2008
Android 1.1 Beta		únor 2009
Android 1.5 Cupcake		květen 2009
Android 1.6 Donut		září 2009
Android 2.0/2.1 Eclair		listopad 2009
Android 2.2 Froyo		červen 2010
Android 2.3 Gingerbread	0,2 %	listopad 2010
Android 3.0/3.2 Honeycomb		únor 2011
Android 4.0 Ice Cream Sandwich	0,3 %	říjen 2011
Android 4.1/4.2/4.3 Jelly Bean	3,0 %	červen 2012
Android 4.4 KitKat	7,6 %	říjen 2013
Android 5.0/5.1 Lollipop	17,9 %	listopad 2014
Android 6.0 Marshmallow	21,3 %	květen 2015
Android 7.0 Nougat	28,2 %	srpen 2016
Android 8.0 Oreo	21,5 %	srpen 2017
Android 9.0 Pie		březen 2018
Android 10.0 Q		neznámé



Obrázek 2.1: Schéma architektury operačního systému Android [9]

Správa napájení systematicky vypíná určité moduly při delší nečinnosti uživatele, jako je například obrazovka. Jelikož je někdy potřeba ponechat modul zapnutý i při nečinnosti uživatele, například při sledování videa či navigace, obsahuje Android zámky zvané `Wakelocks`. Naopak `Alarm Timer` slouží pro probuzení zařízení z režimu spánku.

Další součástí jádra je takzvaný `Binder`, který se stará o meziprocesovou komunikaci a volání metod. Aplikace se při spuštění registruje u služby `Service Manager`. Proces, který chce komunikovat s jiným procesem, musí nejprve získat potřebné informace od `Service Managera`. Poté již může proces komunikovat s jiným procesem, kde je použit `Binder` jako zprostředkovatel. [38, str.75]

2.2.2 Hardware Abstraction Layer (HAL)

HAL poskytuje rozhraní umožňující komunikaci mezi vyšší systémovou vrstvou **Java API Framework** a hardwarem. Skládá se z knihovních modulů, které implementují rozhraní pro specifický hardware. Při přístupu vyšších vrstev k hardwaru, systém Android načte knihovní modul pro daný hardware. Díky tomu vývojář aplikace nemusí řešit hardwarové specifikace různých zařízení. [9],[38, str.75]

2.2.3 Android Runtime

Aplikace, které jsou vytvořené pro platformu Android, jsou nejprve přeloženy do Java Byte kódu a až poté do takzvaného DEX byte kódu. DEX soubory, které jsou kompaktnější než CLASS soubory obsahující Java Byte kód, jsou složené právě z CLASS a JAR souborů. To, jak je poté DEX soubor v samotném zařízení překládán do strojového kódu, záleží na použité verzi systému Android.

Původně byl používán takzvaný Dalvik runtime, který je Just-In-Time (JIT). To znamená, že DEX byte kód je překládán až při běhu aplikace. Jelikož je byte kód překládán pouze po částech, není JIT překladač tak náročný na paměť.

S verzí Androidu 5.0 přišel Android Runtime (ART). Rozdíl oproti Dalvik je použití Ahead-Of-Time (AOT) překladače. Celý DEX byte kód je přeložen už při instalaci aplikace. Přeložený kód je uložen v paměti zařízení a při spuštění aplikace je pouze spouštěn a není již více překládán. Výhodou je menší zatížení procesoru a tedy i rychlejší chod aplikace a nižší zátěž baterie. Nevýhodou jsou však větší paměťové nároky a delší doba instalace.

Pokud aplikace běží při použití ART, měla by fungovat i při použití

Dalvik. Ovšem aplikace vytvořená pro Dalvik nemusí fungovat na ART. [37],[38, str.76]

2.2.4 Nativní C/C++ knihovny

Komponenty systémového jádra, jako je například ART nebo HAL, jsou převážně psané v nativním kódu a vyžadují tak nativní knihovny psané v C nebo C++. Jelikož jsou aplikace pro platformu Android převážně psané v Javě, jsou některé tyto knihovny přístupné přes Java API Framework (viz kapitola 2.2.5). Například knihovna pro 3D grafiku využívající OpenGL ES (OpenGL for Embedded Systems), která je přístupná přes Java OpenGL API.

Pro vytvoření aplikace, která používá vlastní C nebo C++ kód, je k dispozici Android NDK, který umožňuje přímý přístup k již zmíněným knihovnám. [9]

2.2.5 Java API Framework

Jedná se API napsané v jazyce Java, které zprostředkovává základní sadu služeb operačního systému Android. API se skládá ze znovu použitelných bloků, čímž zjednodušuje použití funkcí jádra. [9],[38, str.76-77]

View system - Obsahuje množství prvků grafického uživatelského rozhraní jako jsou například seznamy, tlačítka, textová pole či vestavěný webový prohlížeč.

Resource Manager - Spravuje přístup k nekódovým zdrojům jako jsou například různé texty podle lokalizace či soubory grafického uspořádání (layouts).

Notification Manager - Umožňuje aplikacím zobrazovat vlastní notifikace v notifikační liště systému.

Activity Manager - Spravuje životní cyklus aplikací.

Content Providers - Umožňuje přístup k datům jiných aplikací nebo sdílení vlastních dat aplikace.

2.2.6 Systémové aplikace

Nejvyšší vrstvou jsou aplikace. Android systém v základu obsahuje aplikace pro správu e-mailů a SMS zpráv, kalendář a další. Tyto aplikace nemají

žádná vyšší oprávnění než aplikace třetích stran. Je tedy možné systémové aplikace nahradit vlastní aplikací. Výjimkou jsou aplikace jako například nastavení. [9]

2.3 Základní komponenty Android aplikace

Každá aplikace pro platformu Android se skládá ze čtyř základních komponent, které jsou popsány v této kapitole. Důležité je, že každá aplikace nemusí obsahovat všechny čtyři komponenty, a může obsahovat více komponent stejného typu. Jedná se o komponenty:

- Aktivity (Activities),
- Služby (Services),
- Broadcast přijímače (Broadcast receivers),
- Poskytovatelé obsahu (Content providers).

Každá komponenta je vstupním bodem, přes který může systém nebo uživatel vstoupit do aplikace. Chování a účel užití je pro každou komponentu odlišný.

2.3.1 Aktivity (Activities)

Aby aplikace mohla graficky zobrazovat data na obrazovce a zachytávat interakce od uživatele, potřebuje aktivitu. Aktivita je tvořena jednou obrazovkou a uživatelským rozhraním. Aktivita je tedy první třída, která se zobrazí uživateli při spuštění aplikace.

Každá aktivita může obsahovat Fragmenty. Fragment představuje vzhled a chování v aktivitě, má vlastní životní cyklus a události nemůže ovšem existovat bez aktivity. Zjednodušeně by se dalo říct, že fragment je podaktivita. Jsou využívány zejména v aplikacích, kde se často přechází mezi obrazovkami, jelikož přepínání mezi fragmenty je efektivnější než přepínání mezi aktivitami.

Většina aplikací se skládá z více než jedné aktivity, mezi kterými je možné procházet. Příkladem může být aplikace kontaktů. Jedna aktivita obsahuje seznam všech kontaktů a druhá aktivita zobrazuje detail kontaktu. Pravidlem je, že v jednu chvíli je na obrazovce zobrazena pouze jedna aktivita. Výjimku tvoří tablety, kde je možné již zmíněný příklad s kontakty zobrazit na jedné obrazovce. Místo aktivit se použijí fragmenty. V levé části obrazovky by se nacházel fragment se seznamem a ve zbytku obrazovky by

byl fragment s detailem aktuálně otevřeného kontaktu. Při použití na malé obrazovce by se v jednu chvíli zobrazil pouze jeden fragment stejně jako v příkladu s aktivitami.

Další důležitou vlastností je spouštění aktivit z jiných aplikací. Například při použití aplikace pro zprávy můžeme chtít zobrazit detail kontaktu, se kterým si právě dopisujeme, přímo z aplikace zprávy. Pokud vývojář aplikace kontakty tuto možnost povolil, je možné při otevření aplikace kontakty zobrazit přímo aktivitu s detailem kontaktu.

2.3.2 Služby (Services)

Služby představují operace, které běží na pozadí, zatímco uživatel dále pracuje se zařízením. Jelikož služby běží na pozadí, neobsahují žádné uživatelské rozhraní. Typickým příkladem použití služeb je synchronizace dat s cloudovým úložištěm nebo přehrávání hudby při práci s jinou aplikací.

Služby jsou spouštěné a ukončované z jiných komponent. Jejich výhodou je možnost běhu i ve chvíli, kdy aplikace, ke které služba patří, není spuštěna. Při nedostatku operační paměti zařízení je možné, že služba bude ukončena a automaticky obnovena až při dostatku paměti. Služby však nejsou násilně ukončovány tak často jako aktivity, které nejsou aktuálně vykreslovány na obrazovce. Je tedy v zájmu programátora vytvořit obsluhy těchto událostí, aby služba pokračovala v požadované činnosti. Celkem existují tři různé typy služeb:

Na popředí (Foreground) - Tyto služby jsou pro uživatele důležité a zůstávají běžet i při ukončení aplikace. Aby uživatel byl informován o běhu služby, je nutné, aby byla zobrazena notifikace. Tuto notifikaci nelze odstranit a automaticky zmizí až po ukončení služby. Typickým příkladem takovéto služby je hudební přehrávač. Foreground služby jsou méně náchylné na násilné ukončení při nedostatku operační paměti než ostatní typy služeb.

Na pozadí (Backgroudn) - Background služby provádí operace na pozadí, které nejsou uživateli nijak zobrazovány. Příkladem může být již zmíněná synchronizace s cloudovým úložištěm. Od Android verze 8.0 jsou zavedeny omezení pro Background služby, není-li jejich aplikace na popředí. Pro takové úkoly by se měly používat plánované úkoly (Scheduled job)

Vázané (Bound) - Tyto služby mohou být vázané na jiné komponenty. Umožňují interakci mezi vázanou službou a komponentou v podobě

odesílání požadavků a přijímání výsledků. K jedné službě může být vázáno více komponent. Služba je automaticky ukončena ve chvíli, kdy jsou ukončené všechny komponenty, které se ke službě vážou. Je-li komponenta, ke které se služba váže, na popředí, snižuje se pravděpodobnost násilného ukončení komponenty při nedostatku operační paměti.

2.3.3 Broadcast přijímače (Broadcast receivers)

Komponenta umožňuje systému předat událost do aplikace a umožňují tak aplikaci reagovat na události, které se odehrávají v zařízení. Jelikož každá komponenta může být vstupním bodem do aplikace, může být událost doručena i do aplikace, která není aktuálně spuštěna. Aplikace samotná také může vytvořit broadcast pro ostatní aplikace. Například upozornění pro ostatní aplikace, že data byla stažena.

Broadcast receiver může zobrazit notifikaci pro upozornění uživatele, ale jinak neobsahuje žádné grafické uživatelské rozhraní. Byl navržen pro malou a rychlou obsluhu, která případně může vytvořit jinou komponentu, kde již může probíhat větší zpracování události či interakce s uživatelem.

2.3.4 Poskytovatelé obsahu (Content providers)

Komponenta umožňuje sdílení dat mezi aplikacemi a procesy. Aplikace, která má svá data uložená v paměti zařízení, v SQLite databázi, na webu nebo v jiném úložišti, ke kterému má přístup, může svá data sdílet s ostatními aplikacemi prostřednictvím poskytovatelů obsahu.

Poskytovatelé obsahu se mohou jevit jako rozhraní databáze, nicméně ze systémové hlediska jsou to vstupní body do aplikace pro publikování obsahu. [11],[13],[38, str.83-84, 543-550]

2.4 Senzory

Většina Android zařízení disponuje zabudovanými hardwarovými senzory, které dokáží měřit pohyb, orientaci a další fyzikální veličiny. Android API poskytuje kromě přímého přístupu k datům z těchto senzorů i takzvané virtuální senzory, které získávají data z více senzorů, a poskytují údaje vypočtené z těchto dat. Příkladem může být gravitační senzor nebo lineární akcelerační senzor. [38, str.431] Senzory se dělí do tří kategorií:

pohybové (motion) - Měří akcelerační síly a rotační síly ve třech osách.

polohové (position) - Měří fyzickou pozici zařízení pomocí geomagnetických údajů a údajů z družic GPS nebo triangulací GSM signálu.

environmentální (environmental) - Měří fyzikální parametry okolního prostředí jako je například tlak, osvětlení či teplota.

Senzory jsou přístupné přes Android sensor framework, který umožňuje přístup jak k senzorům hardwarovým tak i k virtuálním. Framework poskytuje následující funkce:

- Získání dostupných senzorů v zařízení.
- Získání vlastností jednotlivých senzorů jako je například maximální rozsah či nároky na baterii.
- Získání naměřených dat ze senzorů.
- Registrovat a zrušit posluchače na změnu stavu senzoru.

Každé zařízení neobsahuje veškeré podporované senzory a zároveň každá verze operačního systému Android nepodporuje veškeré senzory. V tabulce 2.2 jsou uvedené senzory, které jsou podporované od verze 4.0 Ice Cream Sandwich. Zařízení může mít i více než jeden senzor jednoho typu. Příkladem mohou být dva gravitační senzory, které mají různý rozsah. [8]

2.4.1 Práce se senzory

Jelikož zařízení nemusí být potřebným senzorem vybaveno, je vhodné před použitím otestovat přítomnost senzoru. K tomu existují dvě možnosti. První možností je zapsat požadovaný soubor do Manifestu aplikace. Aplikace poté nebude na Google Play dostupná na zařízeních, která senzorem nejsou vybaveny. Druhé řešení je vhodné pro aplikace, které mohou bez senzoru omezeným způsobem fungovat. Přítomnost senzoru je testována až při běhu aplikace před registrací posluchače. Není-li senzor nalezen je vhodné zobrazit uživateli hlášku, že funkcionalita nebude dostupná kvůli nepřítomnosti požadovaného senzoru.

Pro získání dat ze senzorů je potřeba vytvořit posluchač implementující rozhraní `SensorEventListener` a implementovat jeho dvě metody `onAccuracyChanged()` a `onSensorChanged()`. První z metod je volána po změně přesnosti měření senzorem. Přesnost je určena jednou ze čtyř konstant:

- `SENSOR_STATUS_ACCURACY_LOW`,

- `SENSOR_STATUS_ACCURACY_MEDIUM`,
- `SENSOR_STATUS_ACCURACY_HIGH`,
- `SENSOR_STATUS_ACCURACY_UNRELIABLE`.

Druhá jmenovaná metoda je volána při změně hodnoty senzoru. Vstupem do metody je `SensorEvent` objekt obsahující přesnost měření, informace o senzoru, časovou značku měření a naměřená data. Vytvořený posluchač je poté registrován do třídy `SensorManager`. Programátor by měl vypnout senzor odregistrováním posluchače, je-li aktivita zrušena, jelikož některé senzory mají velkou spotřebu energie. Systém automaticky nevypíná senzory při vypnutí obrazovky, což je vhodná vlastnost pro bezpečnostní systém, u kterého bude obrazovka většinu času vypnuta. [8]

Tabulka 2.2: Výpis senzorů podporujících platformou Android od verze 4.0 Ice Cream Snadwich a novější. [8]

Senzor	Typ	použití
Akcelerometr	Hardwarový	Detekce pohybu
Okolní teploty	Hardwarový	Sledování teploty vzduchu
Gravitační	Softwarový / Hardwarový	Detekce pohybu
Gyroskop	Hardwarový	Detekce rotace
Světelný	Hardwarový	Ovládání podsvícení obrazovky
Lineární akcelerometr	Softwarový / Hardwarový	Detekce akcelerace podél jedné osy
Magnetického pole	Hardwarový	Kompas
Orientace	Softwarový	Detekce pozice zařízení
Tlakový	Hardwarový	Sledování okolního tlaku vzduchu
Přiblížení (proximity)	Hardwarový	Pozice telefonu během hovoru
Okolní vlhkosti	Hardwarový	Monitorování rosného bodu a okolní vlhkosti
Rotační	Softwarový / Hardwarový	Detekce pohybu a rotace
Teplotní	Hardwarový	Sledování teploty

2.4.2 Změna stavu baterie

Aby aplikace mohla reagovat na změnu stavu baterie a případně omezit funkcionalitu při kritickém stavu baterie, je nutné nějakým způsobem získat nebo dostat upozornění na tyto změny. Systém Android umožňuje registrovat `Broadcast receiver`, který bude reagovat právě na tyto změny baterie. Při registraci `Broadcast receiveru` se zvolí hodnoty parametru `intent filter` na:

ACTION_BATTERY_CHANGED - pro monitorování změny kapacity baterie.

ACTION_POWER_CONNECTED - pro upozornění na připojení k externímu zdroji energie.

ACTION_POWER_DISCONNECTED - pro upozornění na odpojení od externího zdroje energie.

Systém poté automaticky spustí registrovaný `Broadcast receiver`, v jehož těle lze získat procentuální stav baterie a indikaci stavu napájení. Dále lze reagovat na změnu stavu baterie dle vlastního uvážení. [10]

2.5 Programovací jazyky pro Android

Hlavním a oficiálním programovacím jazykem pro vývoj aplikací pro platformu Android byla do roku 2017 pouze Java. Sice zde existovala a stále existuje podpora C/C++ jazyka prostřednictvím Native Development Kit (NDK), ale základ aplikace musí být stále napsaný v Javě. Dalšími možnostmi zde jsou například programovací jazyky Python, HTML5 + CSS nebo C# prostřednictvím Xamarin. Ani jeden však není oficiálně podporován, a tak většina aplikací byla vyvíjena v Javě.

Na konferenci Google I/O keynote bylo 17. května roku 2017 oznámeno, že programovací jazyk Kotlin se stává novým oficiálním programovacím jazykem pro platformu Android [36]. Java však zůstává i nadále oficiálním programovacím jazykem, a tak si vývojáři mohou vybírat, který jazyk použijí.

Jelikož bude aplikace, která je součástí této práce, vyvíjena v Kotlinu, bude se v této kapitole dále probírat převážně Kotlin a jeho srovnání s Javou.

2.5.1 Kotlin

Open source programovací jazyk Kotlin je vyvíjen firmou JetBrains, která je známa díky velké řadě svých vývojových prostředí. Vývoj začal již v roce

2010 a první oficiální verze se dočkal v únoru 2016. Kotlin je objektově orientovaným jazykem, stejně jako Java, ale zároveň je možné jej použít jako funkcionální programovací jazyk. Dále se jedná o staticky typovaný programovací jazyk pro Java Development Kit (JDK). To znamená, že program vytvořený v Kotlinu lze spustit na zařízeních, kde je nainstalovaná Java. Kromě toho je Kotlin s Javou plně kompatibilní, a tak je možné volat Java kód přímo z Kotlinu, a zároveň i kód napsaný v Kotlinu volat z Javy. Zdrojový kód Kotlinu lze rovněž převést do jazyka JavaScript nebo do nativního binárního kódu, který je určen pro zařízení, kde není možné použít virtuální stroj. Poslední zmíněný překlad je však zatím stále ve vývoji. [23],[34]

2.5.2 Srovnání Kotlinu a Javy

Z pohledu Android vývojáře se Kotlin jeví jako alternativa k Javě, a proto se dále v této kapitole budeme zabývat porovnáním těchto dvou programovacích jazyků. Zde budou uvedeny pouze základní rozdíly. Kompletní srovnání těchto dvou jazyků je uvedeno na oficiálních stránkách Kotlinu [24].

Výkon - Byte kód Kotlinu je velice podobný byte kódu Javy, proto je jeho rychlost srovnatelná se stejným programem napsaným v Javě.

Čas překladu - Kotlin podporuje inkrementální překlady, díky nímž je překlad stejně rychlý nebo rychlejší než v Javě.

Redundance kódu - O Javě je známo, že co se týče řádek kódu není zrovna úsporná. Kotlin tedy v porovnání s Javou ušetří přibližně čtyřicet procent řádek kódu. Tím přispívá k lepší přehlednosti a rychlosti vývoje (programátor nemusí psát tolik programového kódu).

Nenulové datové typy - Nástrahou v mnoha programovacích jazycích je přístup k proměnné, která nabývá hodnoty `null`, což může vést k výjimce. Kotlin se snaží tento problém vyřešit rozdělením datových typů na ty, které mohou obsahovat `null` a na ty které nemohou. Například datový typ `String` nemůže obsahovat `null`, ale datový typ `String?` již může.

Žádné primitivní datové typy - Kotlin na rozdíl od Javy neposkytuje primitivní datové typy. To znamená, že veškeré datové typy se chovají jako objekt. Některé datové typy, jako je například `number, character` nebo `boolean`, mohou být interně primitivní, ale pro programátora je toto skryté a může s nimi pracovat jako s třídami.

First-class funkce - Kotlin obsahuje first-class funkce. To znamená, že mohou být uloženy do proměnné, předány do jiné funkce vstupní proměnou nebo vráceny z funkce. Tyto funkce mají být náhradou Single Abstract Method (SAM) typů, které jsou v Javě od verze 8.

Hlídané výjimky (Checked exceptions) - Pro Javu jsou velice typické takzvané hlídané výjimky. Příkladem může být metoda `append` ze třídy `Appendable`, která je deklarována: `Appendable append(CharSequence csq) throws IOException`. Při každém použití této metody musíme v kódu odchytávat výjimku. V Kotlinu se hlídané výjimky nenacházejí. Zdůvodnění vývojářů pro nepoužití hlídaných výjimek je, že ve velké míře znepřehledňují kód, ale pouze minimálně přispívají k zlepšení jeho kvality.

Statické metody - Na rozdíl od většiny programovacích jazyků, Kotlin neobsahuje statické metody. Ve většině případů mohou být nahrazeny funkcemi na úrovni balíčků.

Ternární operátor - V Kotlinu ternární operátor není potřebný, jelikož jeho funkci nahrazuje rozšíření podmínky `if`. Získání nejvyšší hodnoty z číselných proměnných `a` a `b` by v Javě při použití ternárního operátoru vypadalo následovně:

```
int max = a > b ? a : b;
```

Stejný příklad se dá zapsat v Kotlinu použitím podmínky `if`:

```
val max = if (a > b) a else b
```

[23],[24]

3 Průzkum existujících systémů

V této kapitole bude popsáno pět vybraných existujících systémů zabývajících se analogickým problémem jako tato práce a které využívají mobilní telefony s platformou Android. Dvě aplikace slouží pouze pro zabezpečení vozidla, dvě pouze pro automatické vedení knihy jízd a poslední aplikace slouží jak pro zabezpečení vozidla, tak pro vedení knihy jízd, ale za použití externího zařízení.

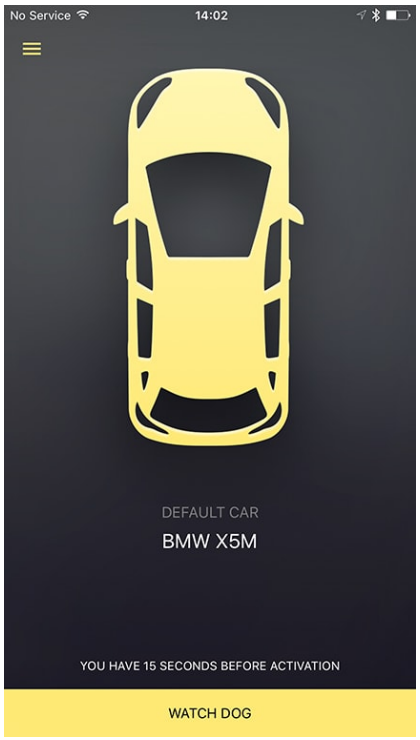
3.1 Car alarm

Car alarm je systém, jak už název napovídá, k zabezpečení automobilu. Nejedná se však o plnohodnotný bezpečnostní systém, ale je určen pouze jako doplněk k zabudovanému autoalarmu. Celý systém se skládá ze serveru a jedné aplikace (Obrázek 3.1), která musí být nainstalována minimálně na dvou mobilních zařízeních. Jeden mobilní telefon je trvale umístěn v automobilu a druhý jen zobrazuje notifikace o stavu prvního zařízení respektive automobilu. Je možno umístit mobilní zařízení do více aut a spravovat je přes jeden telefon. Při prvním spuštěním je nutná registrace a následné přihlášení na všech mobilních zařízeních.

Aktivace alarmu je možná přímo na telefonu v automobilu. Spuštění alarmu má patnácti sekundovou latenci, aby řidič stihl opustit vozidlo. Druhou možností, jak aktivovat alarm, je vzdáleně přes druhý telefon. Deaktivace alarmu lze buď přímo na sledovacím zařízení nebo opět vzdáleně přes druhý telefon.

Při spuštění poplachu je poslána push-notifikace do ovládacího zařízení. Má-li telefon v automobilu validní SIM kartu, je možné při spuštění poplachu vytočit dané telefonní číslo. Telefonní číslo je vytočeno i v případě, že datová síť není k dispozici. Poplach je spuštěn pohybem vozidla, což znamená, že je používán pohybový senzor mobilního zařízení.

V aplikaci je možné zobrazit historii změn alarmu napříč všemi automobily. Například automobil byl zamčen ve 12:30 a v 15:01 byl spuštěn poplach. V aplikaci je dále možné zobrazit mapu, kde je zvýrazněna pozice automobilu. Zajímavou funkcí systému je možnost ovládání přes AppleWatch nebo Android Wear zařízení.

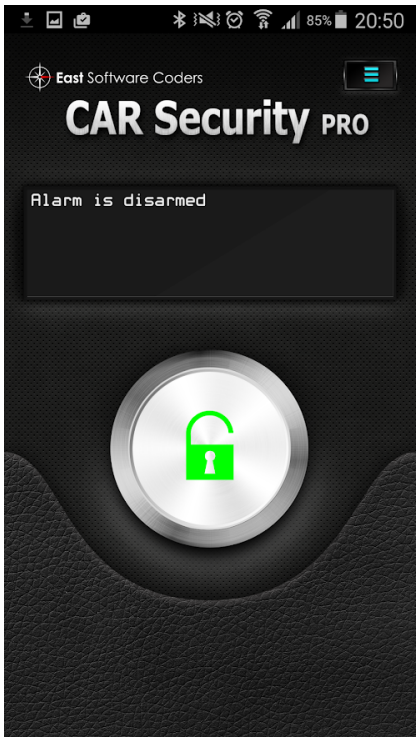


Obrázek 3.1: Úvodní obrazovka aplikace pro zabezpečení automobilu Car alarm.

Aplikace je dostupná, jak pro mobilní telefony s operačním systémem Android, tak i pro mobilní telefony s operačním systémem iOS. Aplikace je velice jednoduchá se zajímavým grafickým prostředím, které kazí zobrazované reklamy. [16]

3.2 Car Security Alarm Pro

Aplikace je určena pro zabezpečení vozidla pomocí mobilního telefonu s operačním systémem Android. Operační systém iOS není podporován. Základní aplikace (viz Obrázek 3.2), která je umístěna v automobilu, je zcela zdarma, ale obsahuje poměrně velké množství reklam, které lze za malý poplatek odstranit. Aplikace do druhého zařízení, kterým lze první aplikaci ovládat, je již placená a její cena se pohybuje v řádu desítek korun. Komunikace mezi telefony probíhá pouze pomocí SMS zpráv, takže komplexnější správa přes webové rozhraní není možná. Jelikož komunikace probíhá pouze pomocí SMS zpráv není aplikace pro ovládání zcela nutná a je možné SMS povely psát přímo ve výchozí aplikaci pro SMS zprávy v telefonu. Jedinou výhodou placené aplikace je propojení aplikací pomocí Bluetooth technologie a



Obrázek 3.2: Hlavní obrazovka aplikace pro zabezpečení vozidla Car Security Alarm Pro.

následné automatické spuštění alarmu při vzdálení se od automobilu. Další možnosti jak spustit alarm je pomocí SMS příkazu nebo přímým spuštěním na aplikaci v automobilu.

Při aktivaci alarmu jsou v pravidelných intervalech zasílány SMS zprávy s polohou zařízení na předem uvedené telefonní číslo. Četnost zpráv lze nastavit, ale pouze na mobilním telefonu, který se nachází ve vozidle. Defaultně je zasílána jedna zpráva každou minutu. Zpráva s polohou obsahuje URL adresu na Google Maps, kde po otevření je vyznačen bod na mapě ukazující aktuální polohu vozidla. V aplikaci nechybí ani notifikace o stavu baterie mobilního telefonu.

Zajímavou funkcí tohoto alarmu je funkce odposlechu, kdy po zaslání SMS příkazu aplikace zavolá na číslo uvedené jako kontaktní.

Systém je velice jednoduchý, ale k zabezpečení vlastního vozidla zcela postačující. Nevýhodou systému je kromě ovládání pouze přes SMS zprávy i nekompatibilita se zařízeními, které mohou mít vložené dvě SIM karty zároveň, takzvané dualSIM telefony. [15]

3.3 Triplog

Triplog je velice komplexní systém pro sledování vozidel. Je určen jak pro jednotlivce pro sledování vlastních jízd, tak pro společnosti, které chtějí sledovat celé flotily vlastních vozidel. Nejedná se tedy o bezpečnostní systém, ale o automatické vytváření knihy jízd. Základní verze systému je zdarma, má však velice omezenou funkčnost. Například správa systému přes webové prostředí je až ve verzi za 40 dolarů ročně za uživatele. Existuje i levnější verze za 20 dolarů a nejvyšší verze Enterprise, u které však není uvedena cena.

Spuštění sledování jízdy je možné hned několika způsoby. Kromě ručního spuštění přímo v aplikaci, je možné spustit sledování po připojení zařízení k napájení nebo při připojení k určitému Bluetooth zařízení. Nahrávání se automaticky spustí při překročení rychlosti pěti mil za hodinu (8 kilometrů za hodinu). Další možností je nastavení časového intervalu během dne, kdy má aplikace nahrávat pohyb vozidla. Poslední možností spuštění sledování jízdy je za pomoci zařízení zvané iBeacon, které se připojí do USB portu v automobilu. Zařízení je bezdrátově spojeno s mobilním telefonem. Při nastartování automobilu je do USB portu zaveden elektrický proud, který aktivuje zařízení, a spustí se sledování aktivity v mobilním telefonu.

Kniha jízd vytvořena systémem může být, dle výrobců, použita při podání daňového přiznání. Tomu napomáhá i rozdělení jednotlivých jízd do několika kategorií, jako například jízda pro pracovní účely či osobní užití vozidla. Kromě vytváření knihy jízd podporuje systém i vkládání účtů jako například za parkování, benzín, opravy a pojištění. K jednotlivým záznamům je možné nahrát i fotografie účtenek.

Kromě prohlížení jednotlivých jízd poskytuje systém i souhrnné statistiky z celého dne. S tím souvisí i možnost zobrazit veškeré jízdy z celého dne na jedné mapě. Jelikož je systém určen převážně pro firemní účely, obsahuje i funkce jako například živé sledování veškerých automobilů v systému či shlukování uživatelů podle jejich pracovního zařazení.

Ačkoliv to výrobce nikde neuvádí, je zřejmé z vyobrazení trasy na mapě, že pozice automobilu je ukládána pouze v daných intervalech, čímž je ušetřeno paměťové místo serveru a využití mobilních dat, ale ztrácí se tím přesnost měření.

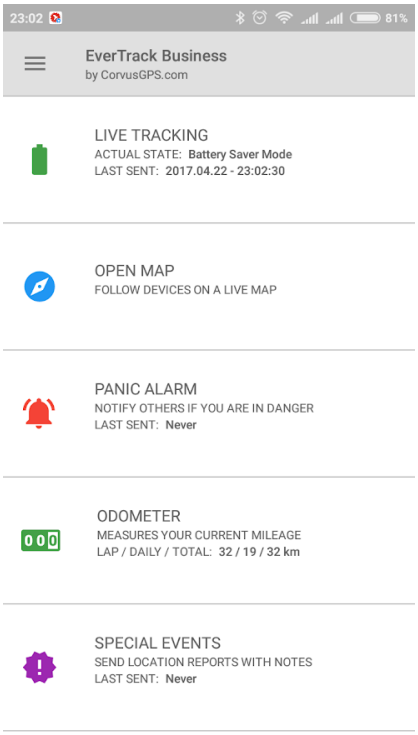
Zajímavou funkcí systému je možnost získávání počtu ujetých kilometrů pomocí externího zařízení, které se připojí do On-Board Diagnostics 2 (OBD2) zásuvky v automobilu. Na základě informací získaných z tohoto zařízení je možné i automaticky spouštět sledování cesty. Další výhodou v porovnání s ostatními systémy je možnost šetření mobilních dat a synchro-

nizovat knihu jízd pouze při připojení k Wi–Fi síti.

Aplikace je dostupná jak pro mobilní telefony s operačním systémem Android, tak i pro mobilní telefony s operačním systémem iOS. Aplikace vypadá velice profesionálně a podporuje velké množství nastavení. Je určena převážně pro firemní účely. Při osobním užití nebude zdaleka využit potenciál aplikace. [7]

3.4 EverTrack

Systém EverTrack slouží pro sledování provozu vozidel pomocí GPS ve firmě, ale je možné jej použít i pro osobní či rodinné účely. Skládá se z aplikace v mobilním telefonu řidiče (Obrázek 3.3), serveru a webového rozhraní. Výhodou je možnost nainstalovat sledovací aplikaci přímo do mobilního zařízení řidiče. Není tedy potřeba instalovat do vozidla přídavné zařízení nebo trvale umístit v automobilu další mobilní telefon. Do systému je však možné připojit různé GPS sledovací zařízení třetích stran, které je zabudované v automobilu, a nahrazuje funkci mobilní aplikace. Navíc systém podporuje i zařízení, které se připojí do OBD2 konektoru v automobilu, díky kterému je možné získat detailnější informace o využití automobilu jako například



Obrázek 3.3: Obrazovka aplikace EverTrack pro vytváření knihy jízd.

otáčky motoru, spotřebu paliva, teplotu motoru či chybové kódy motoru. Dalším externím zařízením, které je možné připojit, je Bluetooth teploměr od firmy Ruuvi[1]. S GPS polohou je poté sdílená i aktuální teplota. Pro použití teploměru však musí být použita již zmíněná mobilní aplikace, jelikož zařízení komunikuje pouze s mobilním telefonem.

Systém je placený a je nabízen ve dvou variantách. První je pro osobní užití a stojí 0.1 dolaru na den za každé zařízení. Business řešení poskytuje více funkcionality a stojí 0.3 dolaru na den za každé zařízení. Velkým omezením aplikace je uchování historie pouze 100 dní i při nejdražším business plánu. Pro osobní užití je historie ukládána pouze 5 dní.

Jelikož je aplikace určena pro firemní užití, mohou řidiči přidávat do systému události jako například zásilka doručena nebo mohou přidat textovou poznámku. Dispečer poté má větší přehled o aktuálním stavu. Dále může řidič zaslat nouzový signál, který se automaticky zobrazí manažerovi či dispečerovi. Dispečer také může odeslat odkaz pro sdílení pozice automobilu přes e-mail zákazníkovi. Tato funkce je vhodná například pro firmy, které se zabývají rozvozem, a chtějí sdílet pozici doručovatele se zákazníkem.

Spuštění nahrávání jízdy je automatické, kdy by aplikace měla sama detekovat jízdu v automobilu. Uživatel aplikace tedy nemůže zapomenout spustit sledování. Výrobce systému uvádí, že rozdíl mezi kilometry naměřenými aplikací a skutečnými kilometry naměřenými tachometrem v autě je od 1 do 5%. Pro šetření baterie je možné změnit interval zjišťování pozice automobilu během jízdy pomocí GPS. Samozřejmě s vyšší úsporou klesá i přesnost měření.

Aplikace je dostupná pouze pro operační systém Android. Z grafického hlediska je zajímavá možnost zobrazit v menu pouze funkce, které chce uživatel používat. Dalším grafickým prvkem je změna barvy cesty automobilu na mapě v závislosti na rychlosti automobilu. [3]

3.5 Carlock

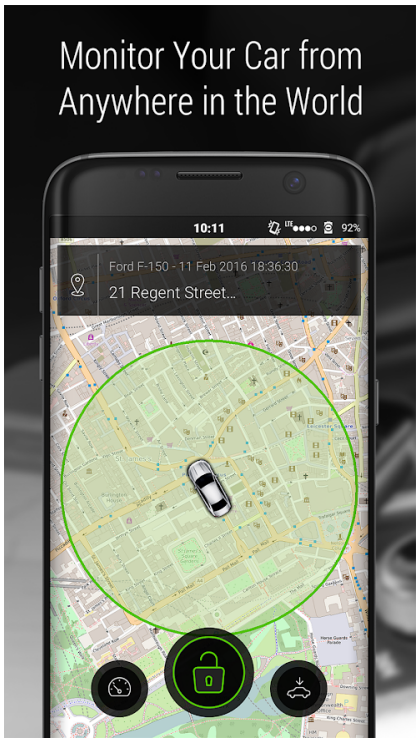
Systém Carlock, jako jediný ze zkoumaných, podporuje jak funkci alarmu, tak knihy jízd. Systém však není založen na mobilním telefonu umístěném v automobilu. Pro chod systému je potřeba zakoupit zařízení, které se připojí do OBD2 konektoru v automobilu. Zařízení poté nahrazuje, a v něčem i přesahuje, funkce mobilního telefonu umístěného v automobilu jako v ostatních případech. Pořizovací cena zařízení je zhruba 100 euro a poté se platí měsíční poplatky ve výši zhruba 8 eur. Součástí celého systému je již zmiňované

[1] https://ruuvi.com/ruuvitag-specs/

zařízení, mobilní telefon s nainstalovanou aplikací (obrázek 3.4), webové rozhraní a server, na který se ukládají data. Data ze zařízení, které se nachází v automobilu, jsou odesílána na server, se kterým následně komunikuje mobilní aplikace a webové prostředí.

Spuštění a vypnutí alarmu lze buď přes ovládací aplikaci v mobilním telefonu nebo použitím speciální zařízení označovaného jako tag. Ten je také umístěn v automobilu. Při přiblížení ovládacího telefonu se zapnutým Bluetooth k automobilu se alarm deaktivuje. Další možností je naplánovat, kdy má být alarm aktivován. Například od 8 večer do 6 ráno bude alarm aktivní.

Systém obsahuje velké množství notifikací, které jsou zobrazovány prostřednictvím aplikace v mobilním telefonu, která pracuje pouze přes internet. Je však možné dostávat notifikace i přes SMS zprávy či hovorem. Kromě klasických notifikací, jako například detekce pohybu, obsahuje systém i notifikace na opuštění vymezeného prostoru, například města, notifikace prudkého brzdění, akcelerace a zatáčení, notifikace překročení nastavené rychlosti či ztrátu GPS signálu. Díky připojení přes OBD2 zásuvku disponuje systém dalšími notifikacemi jako například upozornění na start motoru či odpojení od baterie automobilu. Detekuje-li systém nehodu, informuje pomocí SMS zprávy osobu pro případ nouze.



Obrázek 3.4: Obrazovka aplikace pro ovládání systému Carlock

Zajímavostí je zobrazení skóre řidiče za poslední měsíc. Skóre je vypočteno podle akcelerace, brzdění a prudkého zatáčení automobilu.

Zobrazení knihy jízd obsahuje pouze základní údaje jako počet kilometrů, čas cesty, počáteční a koncový bod cesty. Na mapě, která je hlavním prvkem obrazovky, je současně zobrazena trasa cesty. Nejsou však k dispozici žádné statistiky. Knihu jízd je možné exportovat do csv nebo xls souboru.

Aplikace je dostupná jak pro operační systém Android, tak i pro iOS. Celý systém je graficky přívětivý, a pro osobní sledování vlastních aut je zcela vhodný. Jeho hlavní nevýhodou jsou pořizovací náklady a následné měsíční poplatky. [17]

4 Návrh systému

4.1 Hlavní funkcionalita

V této kapitole bude vybrána a popsána funkcionalita, kterou by systém tvořený v rámci této práce měl podporovat. Celý systém se bude skládat ze serverové části, a alespoň jednoho mobilního telefonu s nainstalovanou aplikací. Server bude obsahovat databázi pro uložení zaznamenaných dat, přístup přes webovou aplikaci a přístup přes API. Dvě hlavní funkcionality systému budou zabezpečení vozidla a vytváření knihy jízd. Funkce bude možné použít nezávislé na sobě. Nebude tedy nutné využít zároveň obě hlavní funkce systému a bude možné použít jen zabezpečovací systém nebo jenom vytváření knihy jízd. Jednotlivé funkcionality jsou dále podrobněji rozepsány v dalších podkapitolách. V každé podkapitole se nachází podrobný popis funkcionalit následovaný jejich výčtem. Kromě dvou hlavních funkcí bude systém doplněn o další funkce, které jsou také dále popsány v této kapitole.

4.1.1 Zabezpečení automobilu

Pro zabezpečení automobilu budou využívány vnitřní senzory mobilního telefonu s operačním systémem Android, který bude trvale umístěn v automobilu. Konkrétně budou používány senzory pohybu, polohy a zvukový senzor.

Zabezpečovací systém (dále jen alarm) bude možné aktivovat několika způsoby. Prvním možností bude odeslat SMS příkaz do mobilního telefonu se spuštěnou aplikací. Druhou možností bude aktivace přes webové prostředí systému. Podmínkou však je stálé připojení mobilního telefonu k internetu. Další způsob jak aktivovat alarm bude přímo přes aplikaci v mobilním telefonu umístěném v automobilu. Po spuštění tímto způsobem bude nastaven časový interval na opuštění vozidla před samotným spuštěním alarmu. Deaktivace alarmu bude možná stejnými způsoby jako aktivace.

Poplach bude aktivován při detekci pohybu pohybovým senzorem nebo při překročení hladiny zvuku. Výběr a citlivost senzorů si uživatel bude moci upravit dle potřeby přímo v aplikaci.

Poplach vyvolá sekvenci akcí, které si uživatel systému zvolí. Hlavní akcí bude odeslání SMS zprávy s informací o poplachu na předem uložené telefonní číslo. Současně bude možné aktivovat funkci volání, která při aktivaci poplachu vytočí jedno dané telefonní číslo. Při zvednutí hovoru bude uživatel moci poslouchat co se děje uvnitř vozidla. Výhodou je, že hovor je těžší pře-

hlédnout než SMS zprávu. Dále, bude-li telefon připojen k internetu, odešle se upozornění o poplachu na server, který zobrazí poplach ve webovém prostředí a odešle e-mail s upozorněním na danou elektronickou schránku. Dále bude možné při poplachu odesílat průběžně zprávy s aktuální lokací zařízení. Další nedílnou součástí bezpečnostního systému je siréna. Ve výchozím nastavení systému bude siréna deaktivována kvůli bezpečnosti, jelikož reproduktory telefonu nejsou dostatečně silné, aby byla siréna slyšet vně auta, a akorát by prozradila pozici skrytého mobilního telefonu. Funkci bude vhodné použít až po připojení externí sirény v pohodě externího reproduktoru.

Výčet funkcionalit zabezpečovacího systému

- Spuštění a vypnutí alarmu:
  - V mobilní aplikaci v automobilu.
  - SMS zprávou.
  - Přes webové prostředí.
- Aktivace poplachu:
  - Detekováním pohybu automobilu pohybovým senzorem mobilního telefonu.
  - Překročením hladiny zvuku.
- Akce vyvolené poplachem:
  - Odeslání SMS upozornění na dané telefonní číslo.
  - Odeslání upozornění na sever.
  - Odeslání e-mailového upozornění.
  - Spuštění zvukového poplachu (siréna).
  - Zavolání na dané telefonní číslo s možností odposlechu.
  - Průběžné odesílání GPS pozice automobilu v daných intervalech.

4.1.2 Vytváření knihy jízd

Automatické vytváření knihy jízd bude využívat zejména polohový senzor. Nahraná trasa bude odeslána na server k dalšímu zpracování a zobrazení přes webové prostředí. Odeslání proběhne ihned po ukončení nahrávání. Nebude-li však mobilní telefon připojen k internetu, proběhne synchronizace při nejbližší příležitosti (například při připojení k domácí Wi–Fi po příjezdu domů).

Ve webovém prostředí budou k zobrazení veškerá data, která byla úspěšně nahrána na sever. Každá jízda bude obsahovat různé statistiky, viz kapitola 4.1.3. Trasa bude viditelná na mapě. Záznam z cesty bude možné exportovat do souboru ve formátu GPX[1].

Výčet funkcionalit automatického vytváření knihy jízd

- Synchronizace nahraných dat se serverem.
- Synchronizace nahraných dat pouze přes Wi–Fi.
- Zobrazení mapy s trasou ve webovém prostředí.
- Zobrazení statistických údajů ve webovém prostředí.
- Export knihy jízd ve formáty GPX.

4.1.3 Zobrazení knihy jízd

Kniha jízd bude zobrazena ve webovém prostředí, kde budou k dispozici i různé statistiky nad naměřenými údaji. Kromě základních údajů jakou jsou průměrná rychlost, ujetá vzdálenost a doba jízdy, bude aplikace umožňovat i zobrazení různých grafů. Grafy bude možné zobrazit nad jednotlivou jízdou. Grafy budou zobrazovat rychlost v průběhu trasy a nadmořskou výšku vozidla.

Výčet zobrazených údajů a grafů

- Počáteční poloha automobilu.
- Konečná poloha automobilu.
- Počet ujetých kilometrů.
- Doba jízdy.
- Čas začátku jízdy.
- Průměrná rychlost.
- Graf rychlosti.
- Graf nadmořské výšky.

[1] XML formát pro ukládání GPS souřadnic a cest.

4.1.4 Správa baterie

Mobilní zařízení bude automaticky sledovat stav a kapacitu baterie. Uživatel bude moci dostávat upozornění na připojení a odpojení zařízení od zdroje elektrické energie. Součástí správy baterie bude i možnost přepnutí aplikace do takzvaného `power save` módu. Zapnutí a vypnutí módu bude probíhat automaticky při poklesu nebo navýšení kapacity baterie nad uživatelem zvolenou hladinu. Funkce, které se vypnou nebo omezí během `power save` módu, jsou vypsány v následujícím seznamu:

- Zvukový senzor se deaktivuje.
- Záznam knihy jízd se vypne.
- Siréna se deaktivuje.
- Je-li povoleno odesílání polohy při alarmu, interval mezi zprávami se prodlouží na 10 minut.
- Komunikace se serverem je omezena na minimum.

Automatická zapínání módu bude možné deaktivovat v nastavení aplikace.

Výčet funkcionalit správy baterie

- Upozornění na připojení a odpojení od napájení.
- Umožnění automatického přepnutí do `power save` módu.
- Nastavení kritické hladiny baterie během kterého se aplikace přepne do `power save` módu.

4.1.5 Další funkcionalita

Hlavní dodatečnou funkcionalitou bude vzdálené ovládání aplikace v mobilním telefonu umístěném v automobilu. Vzdálené ovládání bude prostřednictvím webového prostředí, kde bude možno aktivovat a deaktivovat jednotlivé nástroje. Druhou možností budou SMS zprávy.

Jelikož systém není určen pouze pro jednoho uživatele, bude na serveru vytvořena správa uživatelů. Každý uživatel bude mít vlastní účet, se kterým se přihlásí do webového prostředí a zároveň i do aplikace v telefonu. Do každého účtu poté bude možné přidat více automobilů.

Výčet dalších funkcionalit

- Vzdálené ovládání.
- Správa uživatelů na serveru.

4.2 Komunikace mezi serverem a klienty

Ve většině moderních systémů probíhá síťová komunikace, a ani v tomto systému tomu nebude jinak. Aplikace v mobilním telefonu, umístěném v automobilu, bude odesílat data na server a zároveň server bude moci odeslat například příkaz pro aktivaci zabezpečovacího systému do mobilního telefonu.

Server, se kterým chce klient komunikovat přes internet, musí mít veřejnou IP adresu. Tato adresa je unikátní na celém světě, a díky tomu můžeme na server jednoduše odeslat požadavky či data. Oproti tomu klienti, v našem případě mobilní zařízení, většinou nemají veřejnou IP adresu a odesílání požadavků přímo klientům, bez toho aby klient zprávu očekával, je nemožné. Z toho důvodu budou v této kapitole rozepsány různé metody pro komunikaci mezi serverem a klientem, zejména pak jak kontaktovat klienta, který nemá veřejnou IP adresu.

4.2.1 Polling

Nejjednodušší možností jak odesílat požadavky ze serveru klientovi je pomocí metody zvané polling. Nejedná se však přímo o odesílání požadavku ze serveru. Klient se opakovaně dotazuje serveru, zda neobsahuje nová data určená pro něj. Server odpoví pozitivně nebo negativně, ale vždy odpoví. Velkou nevýhodou této metody je časté dotazování serveru, a tedy velké zatížení sítě a převážně serveru. V případě mobilní aplikace to přináší i nevýhodu v podobě zvýšeného využití mobilních dat. Tato metoda může být použita v případech, kdy je očekávána častá komunikace mezi serverem a klientem. [30]

4.2.2 Long-polling

Nevýhodu pollingu řeší long-polling. Princip zůstává stejný. Klient se dotazuje serveru zda neobsahuje data. Rozdíl je ten, že server neodpoví na klientův dotaz ihned, ale až ve chvíli, kdy může odpovědět pozitivně, tedy má pro klienta nová data nebo vypršel definovaný čas spojení takzvaný timeout. Ve chvíli, kdy klient získá odpověď, ať již v podobě dat nebo vypršení

spojení, odešle na server další požadavek. Problém nastává při měnícím se připojení k internetu. Například přepnutí mezi Wi–Fi a mobilními daty. [14]

4.2.3 WebSockets

Další velice známou metodu pro komunikaci mezi serverem a klientem jsou WebSockety. WebSocket je dlouhotrvající obousměrné spojení. To znamená, že přes toto spojení může odesílat data jak server tak klient. Klient vytvoří WebSocket spojení na server. Data mohou být odesílána jak v binární tak textové podobě. Výhodou je jedno spojení, které je otevřené po celou dobu komunikace na rozdíl od pollingu, kdy je při každém požadavku vytvářené nové spojení. [14] WebSockety jsou vhodné pro real-time aplikace jako jsou například Instant Messaging (IM) aplikace nebo online hry. [30]

4.2.4 Server-Sent Events (SSE)

Již ne tak známou metodu je Server-Sent Events (SSE). Byl navržen přímo pro odesílání zpráv ze serveru na klienta. SSE vytvoří jednosměrné spojení mezi serverem a klientem, což je hlavní rozdíl mezi WebSockets, kde je vytvořené obousměrné spojení. Výhodou je, že může odeslat dotaz na klienta i bez předchozího klientského dotazu jako u pollingu. [30]

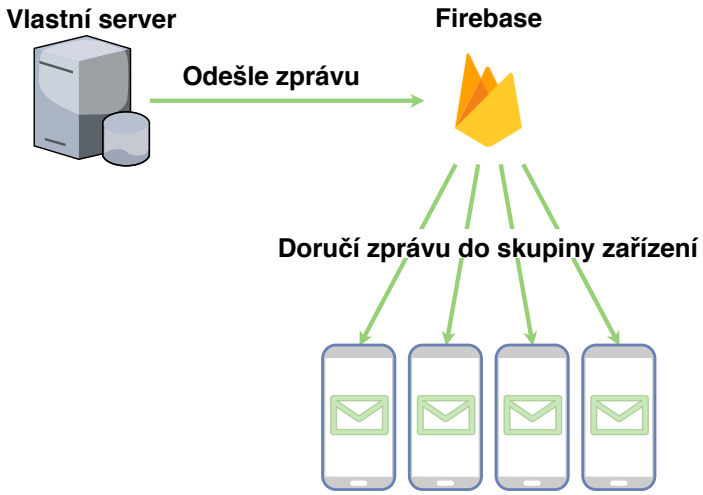
4.2.5 Firebase Cloud Messaging (FCM)

Cloud Messaging je pouze jedna z mnoha funkcí platformy Firebase. Platforma je určena nejen pro Android, ale i pro iOS a webové aplikace. Nabízí realtime databázi, hosting, autentifikaci a další funkce[2] [26]. Firebase je nabízen v několika placených verzích, které se liší množstvím funkcí a datovým limitem. Existuje však i neplacená verze, která obsahuje i FCM.

FCM poskytuje spojení mezi serverem a klientem s ohledem na využití baterie zařízení, které umožňuje odesílat a přijímat zprávy v iOS, Android a webových aplikacích. Zprávy mohou být odeslány, buď přímo danému zařízení, nebo skupině zařízení. Mohou být doručené okamžitě nebo v budoucnu například podle časové zóny uživatele. Zprávy se rozdělují do dvou skupin a to na notifikační zprávy a datové zprávy. Notifikační zprávy jsou zpracovávané pomocí FCM SDK a pouze zobrazí notifikaci uživateli. Datové zprávy jsou však předány do aplikace, která zprávu zpracuje a případně může také zobrazit notifikaci. Velikost zprávy je pro oba typy omezena na čtyři kilobyty. Komunikace s FCM je vyobrazena na obrázku 4.1. Velkou výhodou

[2]Veškeré funkce Firebase jsou na: `https://firebase.google.com/products/`

FCM je, že problémy, jako je například nedostupnost zařízení při odesílání zprávy, jsou již vyřešené. [19]



Obrázek 4.1: Zobrazení komunikace mezi aplikačním serverem, Firebase serverem a aplikací v mobilních zařízeních

4.2.6 Závěr

Metody zvané polling a long-polling nejsou vhodné pro navrhovaný systém, jelikož doba mezi jednotlivými zprávami může být i několik dní, a tyto metody by zatěžovaly server. Metody WebSocket a SSE by mohly být v systému použité, ovšem FCM je pro systém vhodnější, jelikož už řeší problémy nedostupnosti klientů. FCM je navíc pro vývojáře Android aplikací doporučenou možností [18]. V systému pro zabezpečení automobilu bude tedy použit FCM.

4.3 Komunikační rozhraní

Pro efektivní komunikaci mezi jednotlivými částmi systému je potřeba zvolit formát komunikace. Jelikož se bude jednat o microservice architekturu[3] je vhodné vybrat formát komunikace, který je nezávislý na platformě a použitém programovacím jazyce. Během let bylo vyvinuto mnoho formátů jako například RPC, SOAP, REST nebo GraphQL. V následující části budou některé tyto formáty probrány důkladněji a na konci bude vybrán vhodný formát pro navrhovaný systém.

[3]Architektura, ve které je systém rozdělen na nezávisle běžící komponenty.

4.3.1 SOAP

SOAP je nejstarší ze třech vybraných formátů. Snahou SOAP je být nezávislý na protokolu a na architektuře. Data jsou přenášena pomocí XML formátu s předem definovanou strukturou identifikují, že se jedná o SOAP požadavek. Jako přenosový protokol se běžně používá HTTP nebo SMTP. [25]

4.3.2 REST

Snahou RESTu je jednoduchá implementace a lidsky čitelné zprávy. Pro komunikaci se používá protokol HTTP a formát zpráv je typicky JSON, ale je možné použít i jiné formáty jako například HTML nebo XML. Charakteristické pro REST je, že každý typ dat má vlastní URL adresu definovanou programátorem systému. Například pro uživatele můžeme mít adresu `/user` a pro auta `/car`. Druh operace je následně definován HTTP metodou: `GET` pro čtení dat, `PUT` pro aktualizaci dat, `POST` pro vložení dat a `DELETE` pro odstranění. [31]

SOAP vs REST

Rozdíl mezi formáty je v tom, že REST se hodí spíše pro získávání dat přes vystavené API. Výhodou je také oddělený přístup k datům pomocí URL. Oproti tomu SOAP je spíše vhodný pro vzdálené spouštění operací.

Ve skutečnosti je možné pomocí obou systémů dosáhnout stejného výsledku.

4.3.3 GraphQL

Nejnovější ze zmíněných formátů je GraphQL. Byl vyvinut firmou Facebook pro jejich aplikace. V roce 2015 byl zveřejněn pro veřejné užití. Jelikož se jedná pouze o specifikaci není zde omezení protokolem, ale je samozřejmě možné jako v případě REST použít HTTP protokol.

Vzhledem k tomu, že je GraphQL novější ze zmíněných formátů, je vidět, že při návrhu se vývojáři snažili poučit z nedostatků RESTu a SOAPu. [22]

GraphQL vs REST

GraphQL nepoužívá rozdělení zdrojů podle URL, ale k datům se přistupuje pouze přes jednu URL adresu. Díky tomu je možné v jednom požadavku získat několik typů dat. Lze tedy jedním požadavkem získat jak například

uživatele, tak jeho automobil. V případě RESTU by bylo pro tento příklad potřeboval alespoň dvou požadavků.

Další výhodou GraphQL je získání pouze potřebných informací, které se ale musí specifikovat v požadavku. Příkladem může být dotaz na získání uživatelského jména. V případě RESTu je potřeba získat celou strukturu uživatele, ze které se následně získá uživatelské jméno. U GraphQL je možné zavolat požadavek pouze na získání uživatelského jména.

Nevýhodou GraphQL je vracení chyb. Na rozdíl od REST, kde jsou vraceny HTTP status kódy, vrací GraphQL vždy status kód 200 OK a chyba je vložena do těla odpovědi.

GraphQL vs SOAP

GraphQL i SOAP používají pouze jednu URL adresu pro přístup a požadavek je ukryt v těle dotazu. Výhodou GraphQL je možnost používat méně dotazů spojením několika požadavků v jeden. Další výhodou je, že požadavky neobsahují tolik textu a nezatěžují tedy tolik síť. Další společnou vlastností je nutnost deklarování datových typů posílaných dat.

4.3.4 Závěr

Pro potřeby systému je vhodnější použít formát REST nebo GraphQL. GraphQL je novější a odstraňuje nedostatky RESTu. V systémech jako je například Facebook má jeho použití velký význam, jelikož zde probíhá přenos velkého množství dat, což se odrazilo i na formátu GraphQL, kdy je snahou vytvářet co nejméně a co nejkratší požadavky. Pro potřeby navrhovaného systému však postačí komunikace založená na RESTu. Jako formát přenášených dat bude použit JSON. [35]

4.4 Interaktivní mapa

Interaktivní mapa bude tvořena v rámci oborového projektu KIV/OPSWI. Ve webové aplikaci bude možné zobrazit jednotlivé záznamy z knihy jízd. Součástí bude tedy i interaktivní mapa s vyznačenou trasou automobilu. V současné době již existuje mnoho nástrojů, které podporují vizualizaci interaktivní mapy ve webové aplikaci. Zobrazení vlastní trasy na mapě již však není tak běžné. Dále bude představeno několik vybraných nástrojů, které podporují vizualizaci vlastní trasy na mapě.

4.4.1 Google Maps Platform

Mezi nejznámější mapovou platformu patří Google Maps. Kromě přidání mapy do vlastní webové aplikace, podporují i integraci map do Android či iOS aplikací. Jejich mapy poskytují zobrazení trasy na interaktivní mapě i zobrazení rychlostních limitů a funkci Snap to Roads. Tato funkce z předaných GPS souřadnic nalezne nejbližší silnici, po které se pravděpodobně vozidlo pohybovalo. Snahou je eliminovat chybu vzniklou při měření. Další funkce, která by se dala v systému využít je získání adresy z GPS souřadnic. Google Maps poskytují velké množství funkcí, které však nejsou pro účely bezpečnostního systému důležité, a proto zde nejsou zmíněny. [20]

V roce 2018 přešly Google Maps na nový platební systém, kdy aplikace dostane každý měsíc kredit 200 dolarů. Každý požadavek o zobrazení mapy je účtován. Po vyčerpání 200 dolarového kreditu jsou poplatky automaticky strženy z registrované platební karty. Karta musí být uvedena při registraci, jinak nelze službu používat. Například do 200 dolarového limitu je možné 20 tisíckrát volat funkci Snap to Roads, ale v jednom požadavku může být pouze 100 bodů. Pokud by v aplikaci byla zobrazována pouze interaktivní mapa, mohla by být načtena celkem 28 tisíckrát. [21]

Kredit 200 dolarů by pro účely systému byl dostačující, ale nutnost zadání kreditní karty vyřadila Google Maps z nástrojů, které bych mohl pro práci použít.

4.4.2 Bing Maps

Bing Maps je další platformou pro zobrazení interaktivních map a práce s nimi. Stejně jako Google Maps podporuje Bing integraci map do vlastní webové aplikace. Jelikož Bing je součástí firmy Microsoft, nechybí ani podpora pro platformu Windows [1]. Na interaktivní mapě může být zobrazena požadovaná trasa a nechybí zde ani rychlostí limity, funkce Snap to Road a překlad GPS pozice na adresu.

Bing Maps poskytují své služby zdarma pro veřejné webové stránky, které mají ročně méně jak 125 tisíc zpoplatněných požadavků[4]. Pro studijní účely nebo pro neziskové organizace poskytuje Bing zdarma 50 tisíc požadavků na dvacet čtyři hodin. [2]

Bing Maps zcela postačují pro účely navrhovaného systému. Jejich nevýhodou však je nepřehledná dokumentace, která je ovšem velice rozsáhlá.

[4]Požadavky potřebné pro navrhovaný systém spadají do kategorie zpoplatněných požadavků.

4.4.3 Here

Společnost Here, která patří společnosti Nokia, existuje již přes třicet let a spolupracuje i s automobilkami na vývoji navigací v palubních systémech. Kromě toho poskytují i integraci map do webových, Android a iOS aplikací. Zobrazení vlastní trasy na mapě je plně podporováno. Funkce Snap to Road by měla být platformou podporována, kdy se odešle GPX[5] soubor a server vrátí JSON odpověď s vypočtenými souřadnicemi. Z dokumentace však nelze vyčíst, kolik bodů může GPX soubor obsahovat. V dokumentaci se nachází i zmínka o rychlostních limitech, ale jak limity získat, již popsáno není. [4]

Poplatky jsou u Here zcela jednoduché. Měsíčně je zdarma k dispozici 250 tisíc transakcí jak pro komerční tak pro nekomerční účely. Každých dalších tisíc transakcí je účtováno poplatkem jeden dolar [5].

Z finančního pohledu se jedná o nejlevnější řešení ze zde popsaných. Nicméně funkce Snap to Road je zde velice nejasná a rychlostní limity se mi nepodařilo získat.

4.4.4 Závěr

Z vybraných mapových platforem byl vybrán Bing Maps, jelikož je z dokumentace jasné, že požadované funkce jsou podporovány a limity pro nekomerční použití jsou také dostačující.

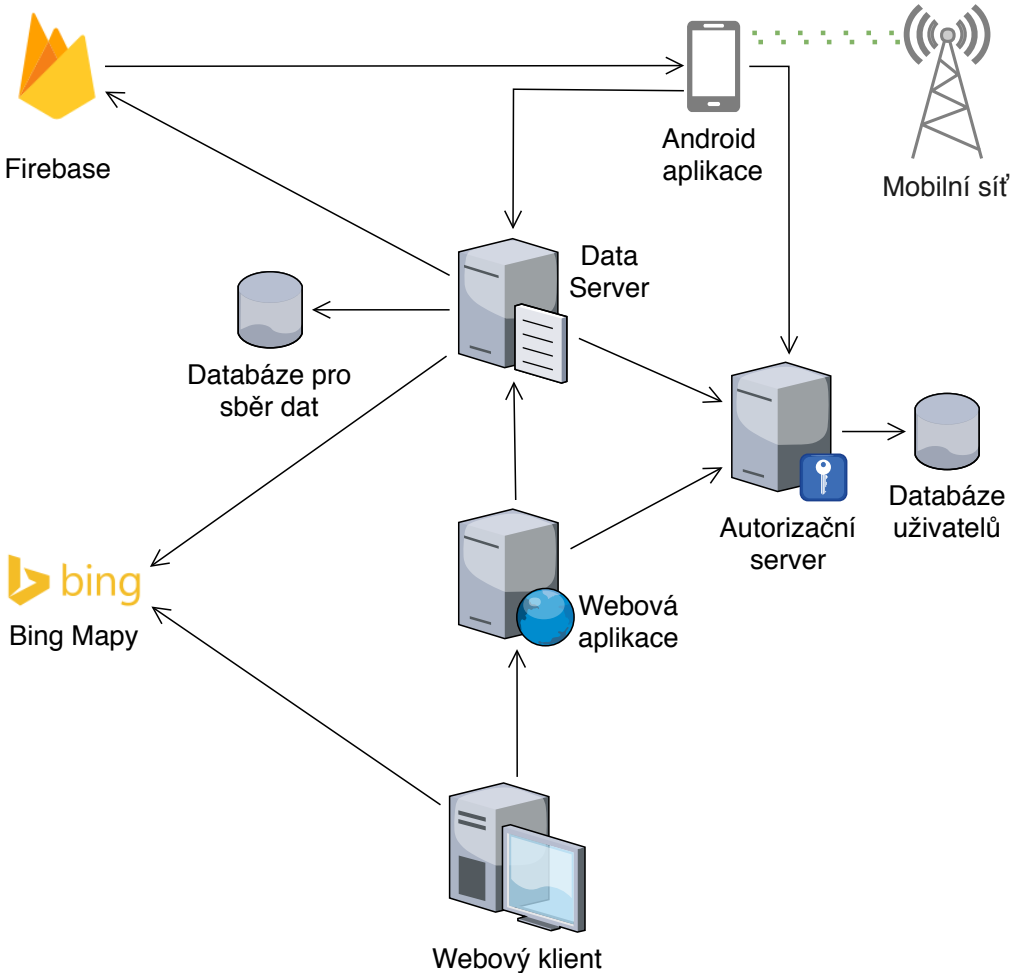
4.5 Struktura systému

Celý systém se bude skládat ze serverové části a mobilní aplikace. Snahou při návrhu serverové části bylo vytvoření architektonického stylu microservice. Z tohoto důvodu bude serverová část rozdělena na autorizační server, server pro přístup k datům, webový server a dvě databáze. Pro komunikaci mezi jednotlivými částmi systému bude použito rozhraní založené na REST (viz kapitola 4.3) Celá struktura včetně systémů třetích stran je na obrázku 4.2.

V následující části bude popsána interakce jednotlivých částí systému.

Autorizační server bude použit pro správu a autorizaci uživatelů napříč celým systémem. Pro autorizaci bude použit protokol OAuth 2.0, který se používá pro přístup přes API nebo pro dlouhodobé přihlášení (Android aplikace) bez nutnosti uložení uživatelských přihlašovacích údajů v zařízení. Autorizační server bude vytvořen za pomoci Spring Security. Pro ukládání uživatelů bude server využívat MySQL databázi.

[5]GPX soubor obsahuje GPS souřadnice uložené v XML formátu.



Obrázek 4.2: Struktura systému včetně interakce se systémy třetích stran.

Datový server bude sloužit jako zprostředkovatel dat a služeb. Každý požadavek bude vyžadovat ověření pomocí OAuth tokenu. Platnost tokenu a identita uživatele bude ověřena prostřednictvím Autorizačního serveru. Veškerá data budou uložena v MySQL databázi. Pro volání Android aplikace ze strany serveru bude využita služba Firebase (viz kapitola 4.2). Pro náhled jednotlivých tras bude Datový server využívat Bing Maps (viz kapitola 4.4).

Webová aplikace bude poskytovat webové rozhraní pro uživatele. Pro přihlášení bude využívat Autorizační server, od kterého získá unikátní token, se kterým se následně bude autorizovat u Datového serveru. Aplikace bude umožňovat zobrazení a správu aut, cest a událostí. Aplikace nebude uchovávat žádná data, ale bude sloužit pouze jako rozhraní pro Autorizační a Datový server. Na klientské části webové aplikace bude

pomocí Java Scriptu přistupováno k Bing Maps API (viz kapitola 4.4) pro zobrazení mapy s cestou.

Android aplikace bude sloužit pro monitoring a sběr dat. Pro komunikaci se systémem bude využívat internetové připojení. Pro upozornění a notifikace bude také využívat GSM mobilní síť pro odesílání SMS zpráv a volání na předem dané telefonní číslo. Pro možnost příjímání příkazů ze serveru, bude Android aplikace připojena ke službě FCM (viz kapitola 4.2.5). Dále bude aplikace komunikovat s Datovým serverem, do kterého bude převážně ukládat získaná data (události a trasy). Pro ověření identity na Datovém serveru bude nejprve nutné přihlášení uživatele přímo v aplikaci. Pro přihlášení bude aplikace komunikovat s Autorizačním serverem.

5 Implementace Android aplikace

Při návrhu architektury Android aplikace byl brán zřetel na možná další rozšíření aplikace o senzory, nástroje nebo komunikační kanály. Na obrázku 5.1 je zobrazen zjednodušený UML diagram architektury, ke kterému se bude stahovat následující text této kapitoly.

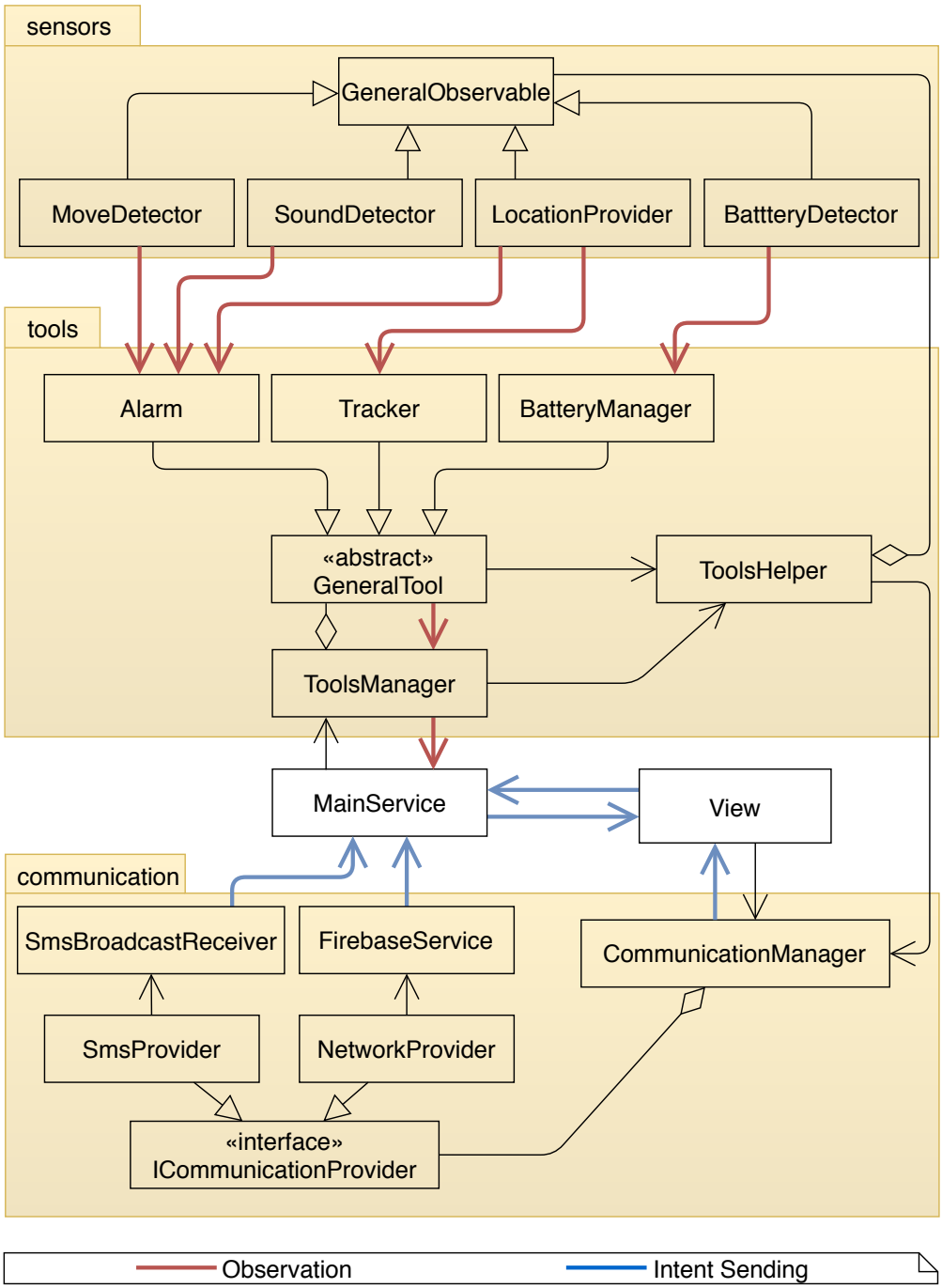
5.1 Senzory

Každý senzor je reprezentován jednou třídou, která komunikuje přes API operačního systému s fyzickým nebo virtuálním senzorem (viz kapitola 2.4). Třída se současně stará o správu daného senzoru a umožňuje registraci pozorovatelů, kteří jsou informováni o změně údajů na senzoru. Pro tuto realizaci je použit návrhový vzor pozorovatel (anglicky Observer). Pozorovateli nezáleží na tom, jakým způsobem jsou data získávána ze senzoru. Pozorovaný (senzor) pouze informuje pozorovatele o změně a předá mu naměřená data. Třída automaticky spustí sledování senzoru při registraci nového pozorovatele a zároveň ukončí sledování senzoru při odregistrování posledního pozorovatele, čímž šetří spotřebu elektrické energie. Výhodou je, že nad těmito třídami nemusí být žádná další třída, která by zapínala a vypínala sledování senzoru. Pozorovatelé jsou výhradně nástroje (viz kapitola 5.2), které se k senzorům registrují sami dle potřeby.

Každý senzor dědí od třídy `GeneralObservable`, která obsahuje logiku pro návrhový vzor pozorovatel, která je totožná pro každý senzor.

5.2 Nástroje

Nástroje jsou hlavním výkonným kódem aplikace. Ke každému nástroji je možné za běhu registrovat a odregistrovat libovolné množství senzorů (viz kapitola 5.1), které následně informují nástroje o změně v senzory měřených datech. Aby nebyla vytvářena nová instance třídy reprezentující senzor v každém nástroji, je mezi všemi nástroji sdílena třída `ToolsHelper`, která zprostředkovává přihlašování k senzorům a drží instance všech použitých senzorů.



Obrázek 5.1: Zjednodušený UML diagram architektury Android aplikace. Modré a červené šipky naznačují další komunikaci jednotlivých tříd, která neprobíhá prostým voláním metod. `View` v tomto případě obsahuje veškeré Aktivity a Fragmenty použité v aplikaci.

Každý nástroj musí dědit od abstraktní třídy `GeneralTool`. Díky tomu je možné jednodušeji vytvořit v systému nový nástroj.

Nad nástroji je vytvořena třída `ToolManager`, která se stará o vypínání a zapínání jednotlivých nástrojů. Mezi nástroji a touto třídou je taktéž použit návrhový vzor pozorovatel zmíněný v kapitole 5.1. Při zapnutí nebo vypnutí nástroje je o této změně informován `ToolManager` a ten ji propaguje dále až do GUI aplikace. Návrhový vzor je použit, aby se každý nástroj mohl libovolně deaktivovat, a aby byl uživatel v takovémto případě informován o změně. Dvěma hlavními implementovanými nástroji jsou Alarm a nástroj pro vytváření knihy jízd. Je-li alespoň jeden z těchto nástrojů spuštěn, je služba (`MainService`), která řídí celou logiku aplikace, přenesena do popředí (Foreground service viz kapitola 2.3.2). Dalším druhem nástrojů jsou takzvané výchozí nástroje, které jsou spuštěny automaticky při startu aplikace a není možné je za běhu ukončit. Příkladem takového nástroje, který je implementován, je správce baterie.

5.3 Komunikace

Při spuštění aplikace je inicializována třída `CommunicationManager`, která automaticky vytvoří uživatelem povolené komunikační kanály. V aplikaci jsou implementovány celkem dva komunikační kanály pro komunikaci prostřednictvím SMS zpráv a pro komunikaci přes internet. Některé nástroje mohou vyžadovat použití určitého komunikační kanálu a proto není možné nástroj spustit je-li daný komunikační kanál zakázán uživatelem nebo nemá-li aplikace dostatečná oprávnění pro použití komunikačního kanálu.

5.3.1 SMS komunikace

Komunikační kanál pro odesílání a příjem SMS zpráv je rozdělen do dvou tříd. První třídou je `SmsProvider`, který se stará o celkovou inicializaci komunikačního kanálu a odesílání SMS zpráv. Pro realizaci odesílání SMS zpráv je využita již existující třída `SmsManager` poskytovaná operačním systémem. Třída během své inicializace vytvoří a inicializuje druhou potřebnou třídu.

Druhou třídou je `SmsBroadcastReceiver`, která dokáže detekovat příchozí SMS zprávu a provést příslušnou akci, která je ve zprávě definována. Pro zabezpečení jsou přijímány příkazy pouze od telefonního čísla definovaného uživatelem v nastavení aplikace. Třída předává příkaz přímo do `MainService`. K tomu jsou využity vnitřní konstrukce operačního systému Android.

5.3.2 Síťová komunikace

Síťová komunikace je stejně jako SMS komunikace rozdělena do dvou částí. První část se stará o přímou komunikaci se servem a druhá část přijímá příkazy ze serveru. Odesílání zpráv je pomocí třídy `NetworkProvider`. Obdobně jako u SMS komunikace, tato třída slouží pro kompletní inicializaci a správu síťové komunikace.

Odesílání zpráv

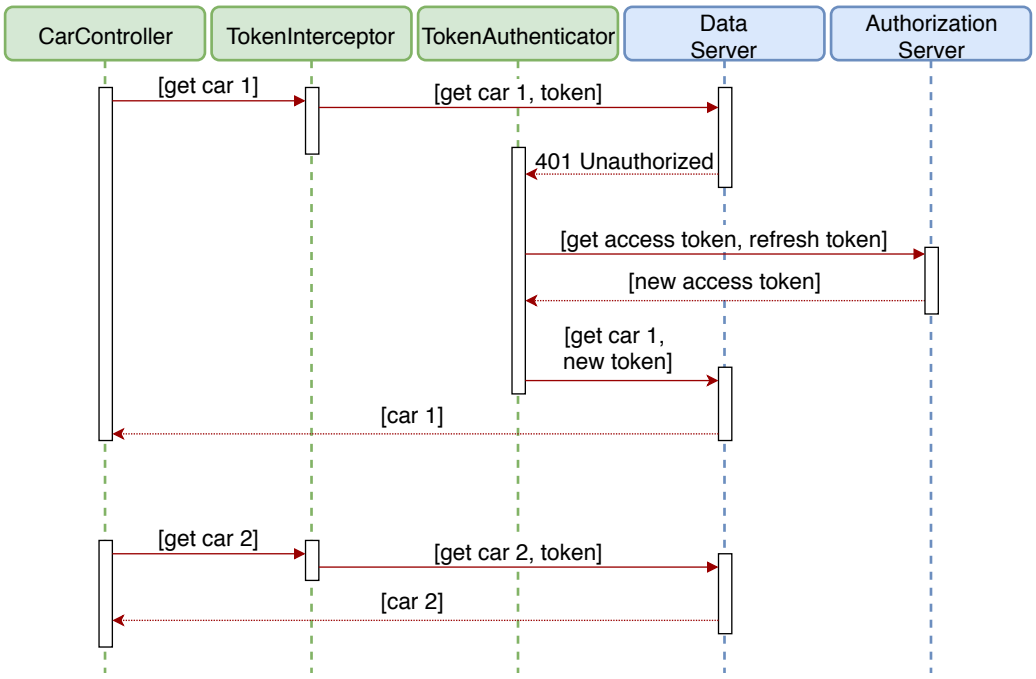
Přímá komunikaci se serverem je zprostředkována prostřednictvím externí knihovny `Retrofit`, která obaluje síťovou komunikaci a poskytuje odesílání HTTP/HTTPS požadavků. Knihovna dokáže automaticky převádět tělo zprávy z JSONu na objekty a zpět. Vzhledem k tomu, že server komunikuje právě pomocí JSON formátu, je příhodné použít knihovnu `Retrofit`.

Koncové body pro komunikaci se serverem jsou definovány rozhraními v balíčku `api`. Tato rozhraní jsou dále využívána v balíčku `controller`. Jednotlivé kontrolery jsou poté přímo volány ze třídy `NetworkProvider`. Datové modely využívané pro přenos dat přes síť jsou definovány v balíčku `dto`.

Zabezpečení

Datový server je zabezpečen pomocí autorizačního protokolu OAuth 2.0 (více viz kapitola 9). Aby nemusel být ke každému požadavku ručně přidáván autorizační token, je využita třída `TokenInterceptor`, která dědí od třídy `Interceptor`. Třída `Interceptor` poskytuje možnost automatického volání s každým požadavkem a umožňuje změnu parametrů požadavku. Je-li tedy uživatel přihlášen, automaticky se přiloží autorizační token ke každému požadavku.

Jelikož autorizační token má pevně danou dobu platnosti, je nutné po vypršení platnosti zažádat o nový autorizační token pomocí obnovovacího tokenu (refresh token). K tomu slouží třída `TokenAuthenticator`, která dědí od třídy `Authenticator`. Je-li odpověď serveru `401 Unauthorized`, třída použije obnovovací token a získá od serveru nový autorizační token, který uloží a opět vykoná požadavek. Výsledkem je, že není nutné dále v kódu řešit autorizaci přihlášeného uživatele. Celý proces je znázorněn sekvenčním diagramem na obrázku 5.2.



Obrázek 5.2: Sekvenční diagram automatické správy tokenů v požadavcích za použití tříd `TokenInterceptor` a `TokenAuthenticator`. V prvním požadavku vypršela autorizačnímu tokenu platnost a je zažádáno o nový. Ve druhém požadavku je již autorizační token platný a data jsou získána ihned.

Komunikace při nedostupnosti serveru

Jelikož knihu jízd je možné vytvářet i bez přístupu k internetu, je nutné data ukládat v zařízení, než bude zařízení opět připojeno k internetu. Vzhledem k tomu, že zařízení může být offline delší dobu, je potřeba držet v paměti až miliony záznamů o pozici. Pro ukládání je tedy využita knihovna `Room`, která tvoří přístupovou vrstvu k zabudované SQLite databázi (viz kapitola 5.8). Po připojení zařízení k internetu se spustí synchronizace uložených dat se serverem. Úspěšně nahraná data jsou ze zařízení smazána. Kromě knihy jízd jsou v databázi při nedostupnosti serveru ukládány i jednotlivé události jako je například odpojení od elektrické energie nebo aktivace alarmu.

Příjem neočekávaných zpráv

Pro komunikaci, která je inicializována serverem (první zprávu odešle server), je použito již zmíněné Firebase Cloud Messaging (FCM) (viz kapitola 4.2.5). Do aplikace je přidáno `Firebase SDK`, které umožňuje serveru komunikovat se zařízením prostřednictvím Firebase serverů. Po přihlášení uživatele do zařízení je odeslán Firebase token do datového serveru. Tímto

tokenem je následně možné jednoznačně identifikovat zařízení. Potřebuje-li server kontaktovat mobilní zařízení, odešle zprávu společně se získaným Firebase tokenem na Firebase server. Ten zprávu přepošle přímo do zařízení. Následující komunikace již probíhá přímou komunikací bez FCM.

Tento způsob komunikace je využíván pro vzdálené zapínání a vypínání nástrojů a pro získání aktuálního stavu zařízení přes webové prostředí systému.

Jelikož je Firebase token vytvářen unikátně pro zařízení, nikoliv pro uživatele, je potřeba ošetřit případy, kdy se na jednom zařízení odhlásí jeden uživatel a přihlásí druhý. V takovém případě mají oba uživatelé pro jedno ze svých zařízení stejný Firebase token. V takovémto případě by mohl odhlášený uživatel zasílat do zařízení příkazy. Pro ošetření je ve zprávě současně s příkazem odesláno i uživatelské jméno. V zařízení jsou následně přijímány příkazy pouze od aktuálně přihlášeného uživatele. Toto ošetření ale neřeší útok, kdy útočník zná jak Firebase token tak i uživatelské jméno. Aby ovšem mohl útočník využít tyto údaje, musel by buď kompromitovat uživatelskou databázi, ve které by pozměnil údaje, nebo by musel získat přihlašovací údaje ke službě Firebase, které jsou také pouze na serveru. V obou případech by tedy útočník musel nejprve kompromitovat server a v takovém případě by ani složitější zabezpečení nepomohlo.

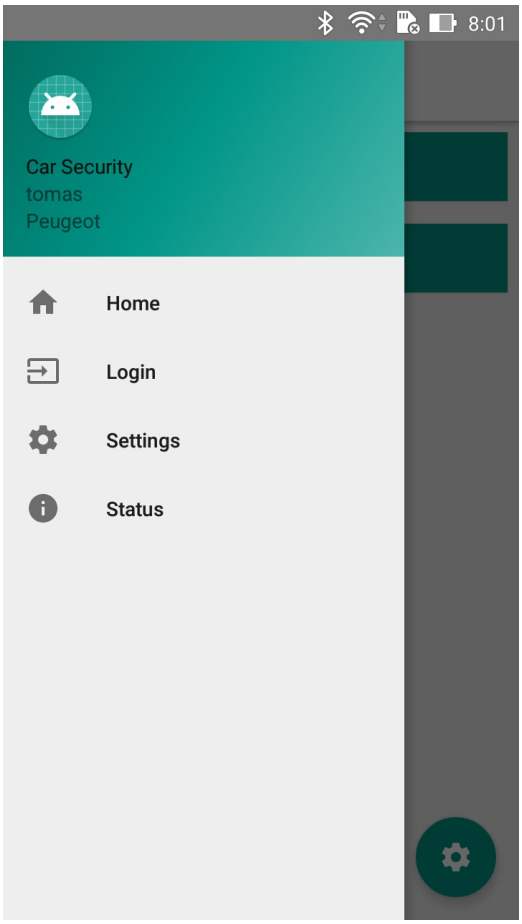
5.4 Aktivity, Fragmenty a služba

Pro navigaci v aplikaci je použit pro Android typický Drawer Layout (viz obrázek 5.3). UML diagram tříd zobrazující aktivity, fragmenty a služby v aplikaci je na obrázku 5.4.

Hlavní aktivita aplikace se skládá ze dvou fragmentů. První fragment obsahuje tlačítka pro zapínání a vypínání nástrojů, současně s indikací aktuálního stavu nástroje. Fragment při své inicializaci spustí službu `MainService`, se kterou komunikuje pouze prostřednictvím zasílání zpráv.

Druhý fragment představuje přihlašovací obrazovku. Součástí fragmentu je i přihlašovací logika. Tento fragment nekomunikuje se službou ale komunikuje přímo se třídou `CommunicationManager`. Jelikož je odesílání zpráv asynchronní, je zpráva s požadavkem pouze předána do `CommunicationManager`. V okamžiku, kdy server odpoví na požadavek, odešle se výsledek přímo do fragmentu, který čeká na odpověď pomocí `LocalBroadcastManagera`. Úspěšně přihlášený uživatel je uložen včetně přihlašovacích tokenů do lokální databáze.

Služba `MainService` v aplikaci propojuje veškeré komponenty a obsahuje



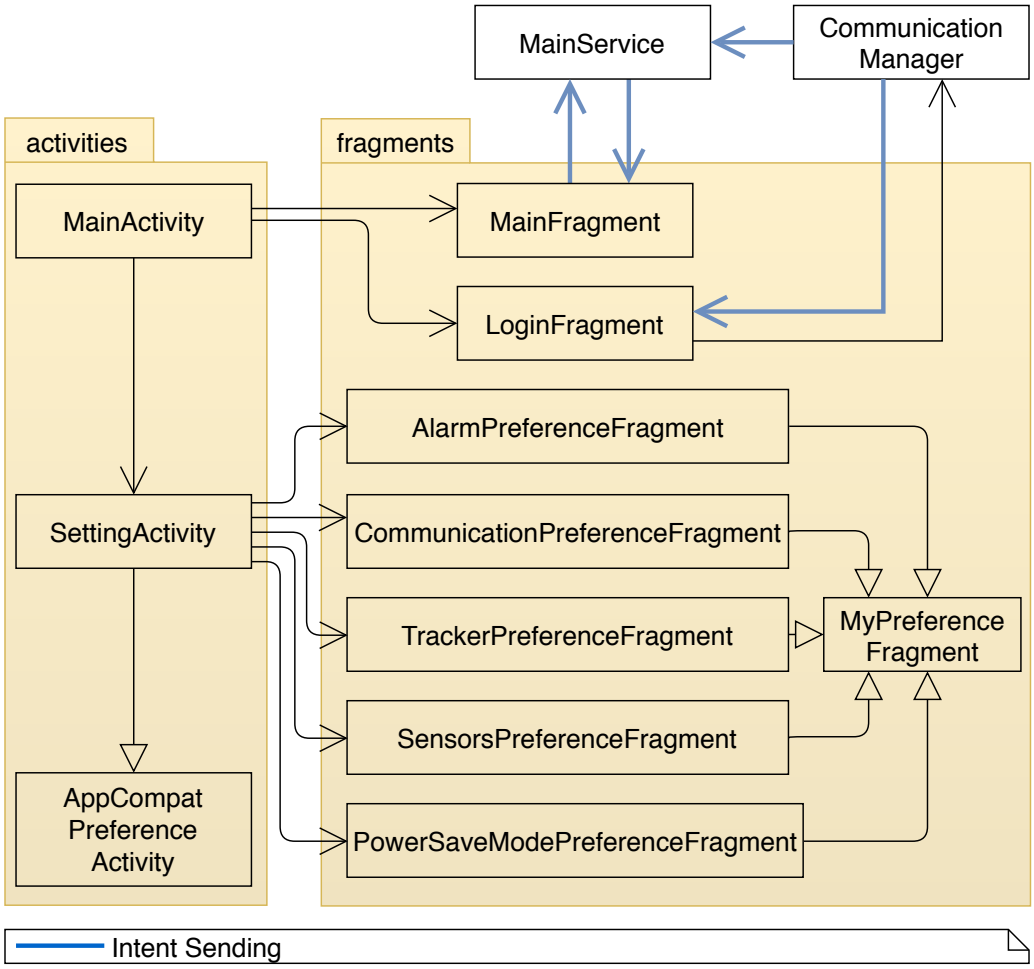
Obrázek 5.3: Ukázka Drawer Layoutu ve vytvořené Android aplikaci. Drawer je realizován postranním výsuvným menu.

tedy řídící logiku aplikace. Služba při svém spuštění inicializuje komunikační kanály a spustí výchozí nástroje (`BatteryManager`). Při spuštění nástroje, který není výchozí (`Alarm`, `Tracker`), je služba přenesena do popředí (Foreground service viz kapitola 2.3.2). Při vypnutí nástroje je služba opět přenesena do pozadí. Veškeré uživatelské příkazy z grafického rozhraní nebo z komunikačních kanálů musí přes tuto službu projít. Příkazy jsou do třídy zasílány zprávami pomocí systémové třídy `LocalBroadcatManager`. Opačná komunikace, do grafického rozhraní, musí také procházet přes `MainService`, která zprávu opět odešle přes `LocalBroadcastManager`.

Druhou aktivitou v aplikaci je `SettingActivity`. Tato aktivita se stará o veškeré obrazovky s nastavením v aplikaci. Nastavení je podrobněji popsáno v samostatné kapitole 5.7.

Veškeré texty a chybové hlášky v aplikaci jsou definované v lokalizovaném `xml` souboru. Pro rozšíření, například o češtinu, tedy pouze stačí vytvořit

překlad tohoto souboru a přidat jej do projektu. Android poté detekuje soubor, jenž je určen pro lokalizaci zařízení, a zobrazí veškeré texty v češtině. V aplikaci je však tento soubor pouze v anglické verzi.



Obrázek 5.4: UML diagram aktivit, fragmentů a služby využívaných v Android aplikaci. Modré šipky představují předávání zpráv pomocí vnitřních konstrukcí sytému Android určených pro komunikaci dvou komponent.

5.5 Vlákna

Dále zmíněná vlákna jsou reprezentovány třídou `WorkerThread`, která dědí od systémové třídy `HandlerThread`. Principem vláken třídy `WorkerThread` je, že vlákno bude existovat během celého života aplikace a ostatní vlákna mohou tomuto vláknu zadávat `Runnable` úlohy. Nové úlohy se ukládají do fronty úloh. Je-li fronta prázdná, vlákno se neukončuje, ale čeká na další úlohu. Výhodou je šetření výpočetního času, jelikož inicializace vlákna se

provádí pouze jednou. Zmíněný přístup je podobný modelu Farmer-Worker s tím rozdílem, že Farmer nemusí být přesně definované vlákno a úlohy mohou být zadávány jakýmkoliv vláknem.

V aplikaci se nachází celkem čtyři nejdůležitější vlákna: `MainThread`, `MainServiceThread`, `NetworkThread` a `ToolsThread`. `MainThread` je hlavním vláknem, ve kterém je spuštěna aplikace.

Příkazy z `MainFragment`, které jsou předávány do `MainService`, běží ve vlákně `MainThread`, ve kterém se zároveň provádí i inicializace `MainService`. Zpracování příkazů však již běží v `MainServiceThread`. V tomto vlákně běží taktéž logika tříd zpracovávajících data ze senzorů a notifikace z nástrojů implementované návrhovým vzorem pozorovatel. Ve vlákně `ToolsThread` běží veškerá logika všech nástrojů s výjimkou jejich inicializace a destrukce. Komunikační kanály si vytvářejí vlastní vlákna dle potřeby, neexistuje tedy žádné globální pro všechny komunikační kanály. Logika SMS komunikačního kanálu je tak malá, že běží ve vlákně volajícího. Síťový komunikační kanál již ale obsahuje vlastní vlákno `NetworkThread`, ve kterém jsou spouštěny veškeré odchozí požadavky.

5.6 Context

Pro jednotný přístup k `SharedPreferences` (viz kapitola 5.7) je napříč aplikací sdílen vlastní kontext. Hlavní třídou je `MyContext`, která obsahuje instance tříd `SensorContext`, `ToolContext` a `CommunicationContext`. Tyto třídy obsahují metody pro čtení a zápis jednotlivých hodnot nastavení aplikace, které jsou uloženy právě v `SharedPreferences`. Dále třída `MyContext` obsahuje proměnné objektů, které jsou potřebné napříč celou aplikací. Příkladem může být `ApplicationContext`, který je přístupný pouze z Android komponent. Díky třídě `MyContext` je `ApplicationContext` dostupný v celé aplikaci.

5.7 Nastavení

Uživatelské nastavení aplikace je uložené v Android úložišti, které se nazývá `SharedPreferences`. Do tohoto úložiště se data ukládají ve formátu klíč–hodnota. Úložiště je nastaveno tak, aby byla data přístupná v celé aplikaci, ale nepřístupná z vnějšku aplikace. Toto je možné zaručit pouze v telefonech, které nepodstoupily takzvaný root[1].

[1]Operační systém Android, na kterém byl proveden root, umožňuje přepnutí na super uživatele, který má práva prohlížet veškerá data a provádět veškeré operace. [33]

Pro úpravy uživatelského nastavení za běhu aplikace je použita třída `PreferenceFragment`, která zobrazí nastavení definované v `XML` dokumentu, jehož kořenovým elementem je `PreferenceScreen`. V `XML` dokumentu jsou dále definované jednotlivé položky nastavení a o jaký typ se jedná. Při změně hodnoty v nastavení se automaticky změní přidružená hodnota v `SharedPreferences`. Nastavení je v aplikaci rozděleno do pěti kategorií.

Alarm - nastavení týkající se nástroje alarm.

Kniha jízd - nastavení týkající se nástroje knihy jízd.

Senzory - nastavení jednotlivých senzorů.

Komunikace - nastavení komunikačních kanálů.

Úsporný režim - nastavení úsporného režimu.

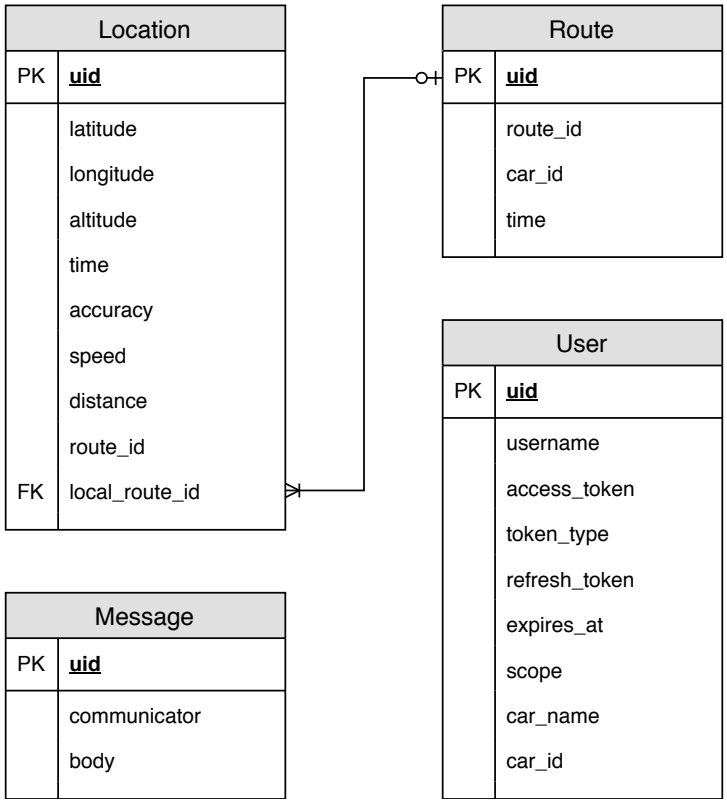
Další výhodou použití `SharedPreferences` je možnost zaregistrovat posluchač na změnu hodnoty v úložišti. Příkladem může být změna citlivosti senzoru, která se přiděluje senzoru pouze při inicializaci. Není dostačující změnit pouze hodnotu v úložišti. Třída se senzorem tedy zaregistruje posluchač a při změně příslušné hodnoty senzor opětovně inicializuje. Výhodou je, že hodnota může být měněna kdekoliv bez nutnosti přímého volání inicializace senzoru.

5.8 Databáze

V aplikaci je použita i lokální databáze, do které se ukládají záznamy z knihy jízd, před tím než jsou odeslány na server. Zároveň jsou v databázi ukládány neúspěšně odeslané požadavky na server a informace o aktuálně přihlášeném uživateli. Jako databáze je použita SQLite databáze. Přístup k databázi je řízen abstraktní vrstvou zvanou `Room`. Data v této databázi jsou přístupná pouze z aplikace, která databázi vytvořila [29, str. 150]. SQLite databáze společně s `Room` jsou doporučenou metodou pro ukládání velkého množství strukturovaných dat v Android aplikaci [12].

Pro databázi jsou vytvořeny entity (modely) a pro přístup jsou vytvořeny DAO objekty. ERA model popisující strukturu tabulek je na obrázku 5.5. Tabulka `Message` slouží pro ukládání událostí (`Event`). Položka `communicator` identifikuje o jaký komunikační kanál se jedná a `body` je text zprávy převedený na znaky. V případě síťového komunikátoru se jedná o již vytvořený JSON.

Tabulka `User` obsahuje vždy pouze jeden záznam aktuálně přihlášeného uživatele. Při odhlášení jsou automaticky všechny záznamy v databázi odstraněny.



Obrázek 5.5: ERA model lokální SQLite databáze v Android aplikaci.

6 Implementace datového serveru

Hlavní server pro správu dat má jako jediný přístup k uživatelským datům uloženým v databázi s výjimkou dat týkajících se uživatelského účtu, které jsou přístupné pouze přes autorizační server. Datový server je napsán v programovacím jazyce `Kotlin` za použití Frameworků `Spring, Spring Boot` a `Spring Security`. Pro nasazení aplikace je použit `Docker`.

Server samotný neumožňuje žádnou možnost přihlášení. Pro získání nebo uložení dat je ale nutné ověřit svoji identitu odesláním OAuth autorizačního tokenu v hlavičce požadavku. Odesílající musí token získat přímo od autorizačního serveru. Datový server tento token ověří vůči autorizačnímu serveru, a pokud je platný, a uživatel má dostatečná práva, je provedena požadovaná operace. Uživatel má práva na čtení pouze vlastních dat a vkládat záznamy může pouze k vlastním autům.

6.1 Databáze

Jako výchozí databáze je zvolena `MySQL`, ale díky použití frameworku `Spring` a standardu `Java Persistence API (JPA)` je možné databázi změnit pouhým přepsáním konfiguračního souboru. Databázové tabulky jsou mapovány na `modely`, se kterými se následně pracuje ve zbytku aplikace. ERA model databáze je na obrázku 6.1.

Pro přístup k datům je použit `Spring Data Repository`, díky kterému není nutné psát metody pro přístup k tabulkám, ale je pouze definováno rozhraní tříd, které k tabulkám zprostředkovávají přístup. `Spring` dokáže vytvořit správný dotaz do databáze podle anotací a podle názvu metody. Například název metody `findAll` vytvoří SQL dotaz `SELECT * FROM ...` Pro složitější konstrukce je možné zadefinovat i vlastní SQL dotaz. Výhodou je i automatické ošetření SQL Injection (více viz kapitola 9).

Sloupce `avg_speed, length` a `seconds_of_travel` v tabulce `route` jsou dopočítávány z hodnot v tabulce `position`. Při každém vložení nových záznamů do tabulky `position` jsou tyto statistiky v příslušných cestách vymazány. Smazané statistiky jsou opět dopočítány, ale až ve chvíli, kdy je volán požadavek na čtení cesty. Tento způsob je zvolen kvůli úspoře výpočetního času, jelikož pozice pro jednu cestu mohou přijít v několika různých zprávách

a nejsou-li statistiky potřeba bylo by zbytečné je neustále dopočítávat.

6.2 Kontrolery

Jak bylo popsáno v kapitole 4.3, s datovým serverem se komunikuje pomocí JSON zpráv, které jsou přenášené pomocí HTTPS protokolu. JSON zprávy jsou mapované na objekty, se kterými se dále v aplikaci pracuje. Pro úsporu procesorového času jsou tyto modely ve většině případů shodné s databázovými modely a není tedy nutné další mapování.

Nad všemi kontrolery je třída `GlobalControllerAdvice`, která zachytává výjimky vzniklé v aplikaci, aby nebyly zobrazeny přímo uživateli. Volající dostane v odpovědi JSON obsahující chybovou zprávu. Vzniklá výjimka je navíc zapsána do serverového logu.

6.3 Firebase

Jak bylo zmíněno v kapitole 4.2 je využita komunikace prostřednictvím FCM. V serverové aplikaci je přidána závislost do `Maven`[1] skriptu na Firebase SDK. Na serveru je dále uložen přístupový klíč v podobě JSON souboru. Inicializace FCM komunikace je spuštěna ihned po startu serveru. V případě příchozího požadavku z webové aplikace obsahující příkaz pro specifické zařízení, je odeslán příkaz do Firebase, který jej předá dále do zařízení. Je-li to žádoucí, zařízení kontaktuje datový server s odpovědí na příkaz.

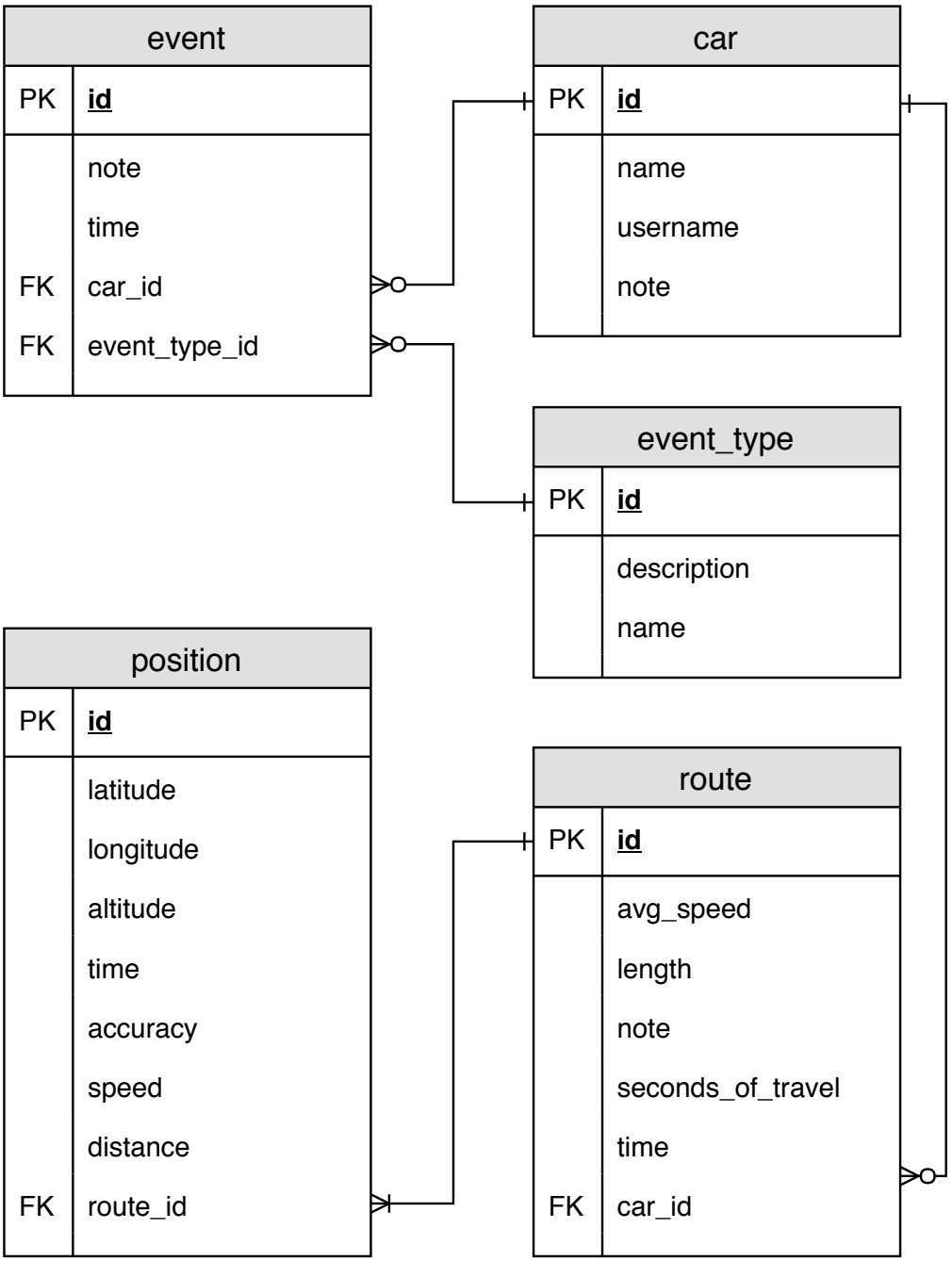
6.4 Bing Maps API

Datový server komunikuje s `Bing Maps API` pro získání náhledů jednotlivých cest. Jednotlivé náhledy jsou ukládány v souborovém systému datového serveru pro šetření limitů Bing Maps API. Pokud není náhled nalezen v datovém serveru, vygeneruje se požadavek, který se odešle do Bing Maps API. Získaná mapa se na datovém serveru uloží pro pozdější užití. Celý proces získání a uložení mapy je na obrázku 6.2.

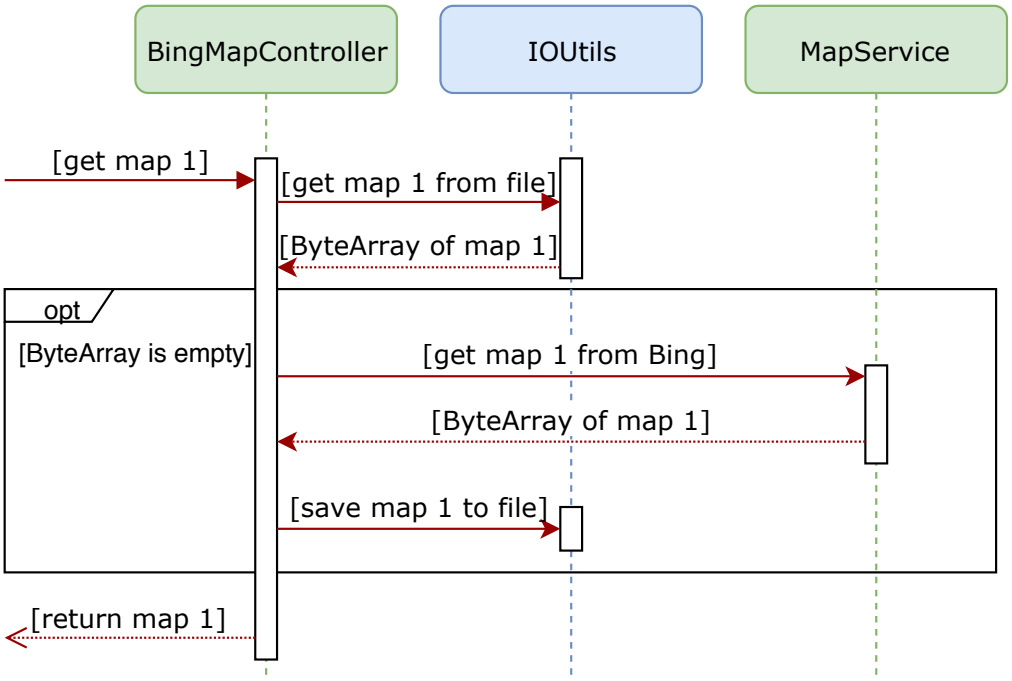
6.5 Export trasy

Jednotlivé trasy je možné exportovat do `GPX` formátu. Pro export je použita Java GPX knihovna (JPX), která je samostatnou součástí knihovny

[1] Maven je nástroj pro řízení překladu aplikací.



Obrázek 6.1: ERA model databáze používané v datovém serveru pro ukládání dat z mobilní Android aplikace.



Obrázek 6.2: Sekvenční diagram získávání a cachování náhledů cest.

`Jenetics`. Do knihovny jsou nahrány jednotlivé pozice, které jsou uloženy v databázi. Jelikož knihovna nepodporuje export do řetězce, je nutné převést data z knihovny do `ByteArrayOutputStream`, který již lze převést na řetězec. Vytvořený řetězec je vrácen uživateli při požadavku o export trasy do GPX.

6.6 Odesílání e-mailů

Datový server také odesílá e-mailová upozornění. Zprávy jsou generovány na základě přijatých událostí ze zařízení. Mezi událostí, které způsobí odeslání e-mailu patří:

- Spuštění poplachu.
- Vypnutí poplachu.
- Připojení zařízení k napájení.
- Odpojení zařízení od napájení.
- Spuštění módu pro šetření baterie.

Pro odesílání zpráv bylo vytvořeno rozhraní `MailService` a následně jeho implementace `MailServiceImpl`. E-maily jsou odesílány prostřednictvím třídy `JavaMailSender`, která je součásti Frameworku `Spring`. Veškerá konfigurace e-mailového klienta je v konfiguračním souboru aplikace.

Pro získání e-mailové adresy uživatele je nutné kontaktovat autorizační server. Při každém odeslání e-mailu je tedy odeslán požadavek do autorizačního serveru. Pro autorizaci je použit ten samý token, kterým se uživatel ověřil u datového serveru. E-mail je poté odeslán na získanou e-mailovou adresu.

7 Implementace autorizačního serveru

Autorizační server slouží pro přihlášení, registraci a správu uživatelů a klientů[1]. Stejně jako datový server je autorizační server napsán v programovacím jazyce `Kotlin` za použití Frameworků `Spring, Spring Boot` a `Spring Security`. Pro nasazení aplikace je taktéž použit `Docker`.

7.1 Zabezpečení

Aby klienti mohly se serverem komunikovat, musí být zaregistrovaní u autorizačního serveru svým `client id` a `client secret`. Každý klient má sadu přidělených rolí, které definují rozsah jeho pravomocí. Například registrovat nové uživatele může pouze klient s rolí `USER_REGISTRATION_CLIENT`. Ostatním klientům je při pokusu o registraci nového uživatele vrácena odpověď `401 Unauthorized`.

Uživatel po přihlášení obdrží autorizační token, se kterým může následně přistupovat k údajům jak na autorizačním serveru, tak na datovém serveru. Tento token je využíván v Android aplikaci, ale také i ve webové aplikaci.

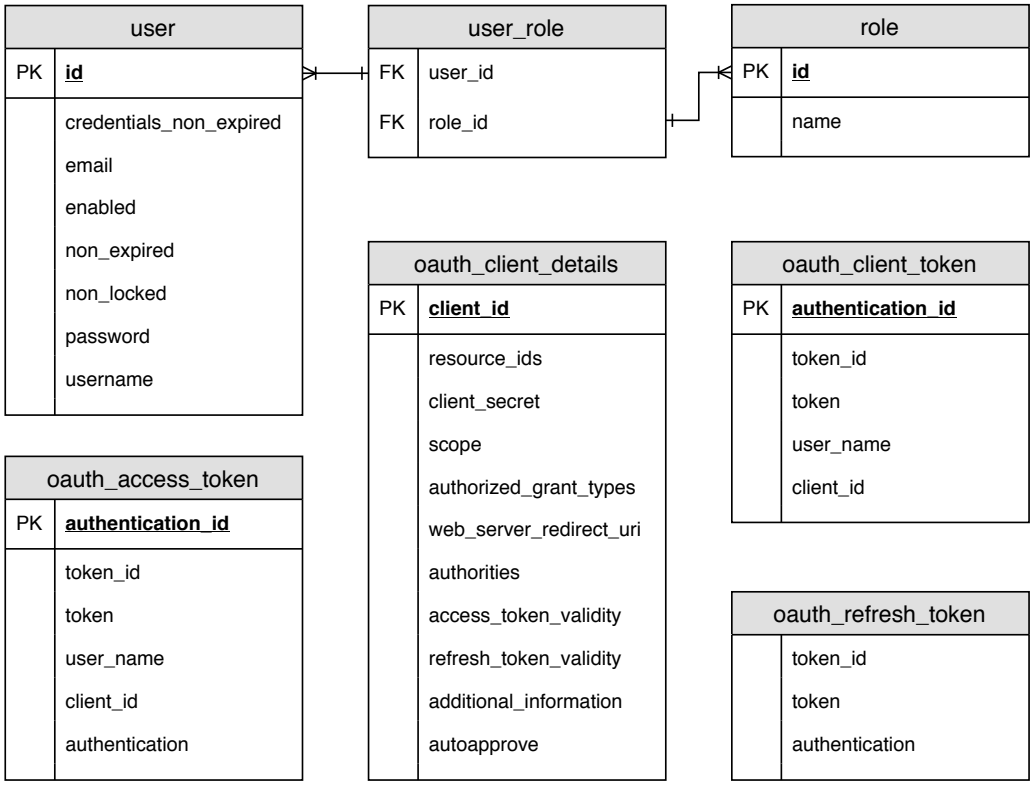
Každý požadavek s výjimkou přihlášení musí obsahovat autorizační hlavičku, která obsahuje autorizační token, čímž se ověří, že klient nebo uživatel mají dostatečná práva pro vykonání požadované operace. Další zabezpečení je individuálně v každém kontroleru pomocí anotací. Ve většině případů má uživatel přístup pouze ke svým uživatelským datům, pouze uživatel s rolí administrátora má přístup i k ostatním účtům. Veškerá hesla jsou v aplikaci kódována pomocí hašovacího algoritmu `BCrypt`.

7.2 Databáze

Úložiště všech tokenů, klientů a uživatelů je pomocí konfigurace nastaveno na `MySQL` databázi, ve které je vytvořeno databázové schéma vycházející z ERA modelu na obrázku 7.1. Toto schéma je vytvořeno částečně pomocí SQL skriptu `data.sql`, který se nachází ve složce `resources` a částečně pomocí modelů vytvořených v balíčku `domain`. Stejně jako v případě datového

[1]Klientem jsou myšleny aplikace přistupující k serveru. V případě navrženého systému jsou to Android aplikace, datový server a webová aplikace.

serveru je použit pro přístup k databázi `Spring Data Repository`, takže je definováno pouze rozhraní obsahující hlavičky metod pro přístup k datům.

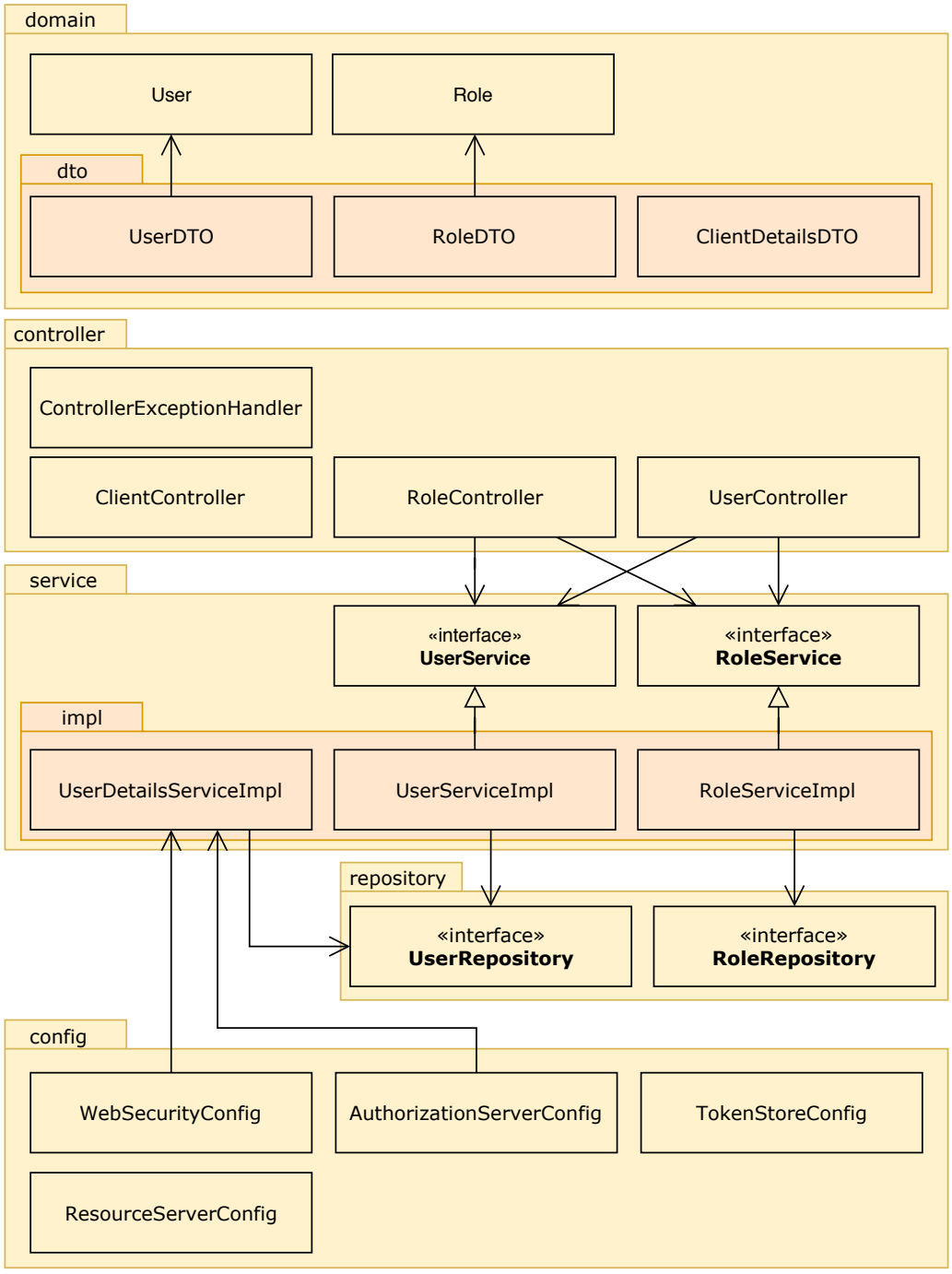


Obrázek 7.1: ERA model databáze používané v autorizačním serveru pro ukládání dat o uživatelích a klientech. Kromě tabulek `user` a `role` jsou použity výchozí tabulky získané ze `Spring Security`.

7.3 Architektura

Při implementaci architektury byla snaha o dodržení vícevrstvé architektury viz UML diagram na obrázky 7.2. UML diagram obsahuje pouze třídy vytvořené v rámci aplikace a nejsou zde zaneseny závislosti vytvořené použitím `Bean`. Konfigurace `Spring Security` se nachází v samostatném balíčku `config` a je tvořena pomocí čtyř tříd, které jsou automaticky načítány při startu aplikace díky anotaci `Configuration`. Pro přístup k databázovým tabulkám jsou vytvořena rozhraní v balíčku `repository`. Mezi balíčky `repository` a `controller` se nachází balíček `service` obsahující logiku pro přístup k datům přes `repository`. Poslední vrstvou jsou kontrolery. K těm již lze přistupovat pomocí HTTPS požadavků. Nad všemi kontrolery je postavena třída `ControllerExceptionHandler`. Tato metoda zachytí výjimky

z kontrolerů a zapíše je do logu serveru. Další přístupové body aplikace, které slouží například pro přihlášení jsou generované automaticky pomocí `Spring Security`.



Obrázek 7.2: Diagram tříd autorizačního serveru. V diagramu nejsou kvůli přehlednosti zakresleny vazby do balíčku `domain` jehož třídy jsou používány napříč aplikací.

8 Implementace webové aplikace

Webová aplikace slouží pro správu alarmu a knihy jízd přes webové prostředí, což obnáší jednotlivá auta, události a ujeté trasy. Stejně jako datový server a autorizační server je webová aplikace napsána v programovacím jazyce `Kotlin` za použití Frameworků `Spring, Spring Boot` a `Spring Security`. Pro nasazení aplikace je taktéž použit `Docker`. Pro vytvoření HTML stránek je použit framework `Thymeleaf` za použití knihovny `Bootstrap`. Aplikace komunikuje jak s autorizačním serverem pro přihlášení uživatele, tak i s datovým serverem pro zobrazení uživatelských dat. Veškerá data tedy získává přes servery a neobsahuje žádnou lokální databázi.

8.1 Přihlášení a registrace

`Spring` pro přihlášení využívá třídu `AuthenticationProvider`, která je v aplikaci překryta třídou `CustomAuthenticationProvider` pro zajištění vlastní přihlašovací logiky. V metodě `authenticate` je odeslán HTTPS dotaz pomocí `RestTemplate` do autorizačního serveru. Aby byl požadavek autorizačním serverem přijat, musí být přiložena hlavička `Authorization` s aplikačním `client id` a `client secret`. Pokud je vše v pořádku je od uživatele získán autorizační token, který je uložen do session. V opačném případě je uživateli zobrazena chybová hláška.

Pro registraci uživatele je nejprve nutné ověřit identitu aplikace přihlášením se pomocí OAuth client credentials flow do autorizačního serveru. To znamená přihlášení se pomocí `client id` a `client secret`. V následujícím kroku je odeslán registrační formulář do autorizačního serveru. Výsledek operace je opět zobrazen uživateli. Aby mohla aplikace vytvářet nové uživatele v autorizačním serveru, potřebuje mít roli `USER_REGISTRATION_CLIENT`.

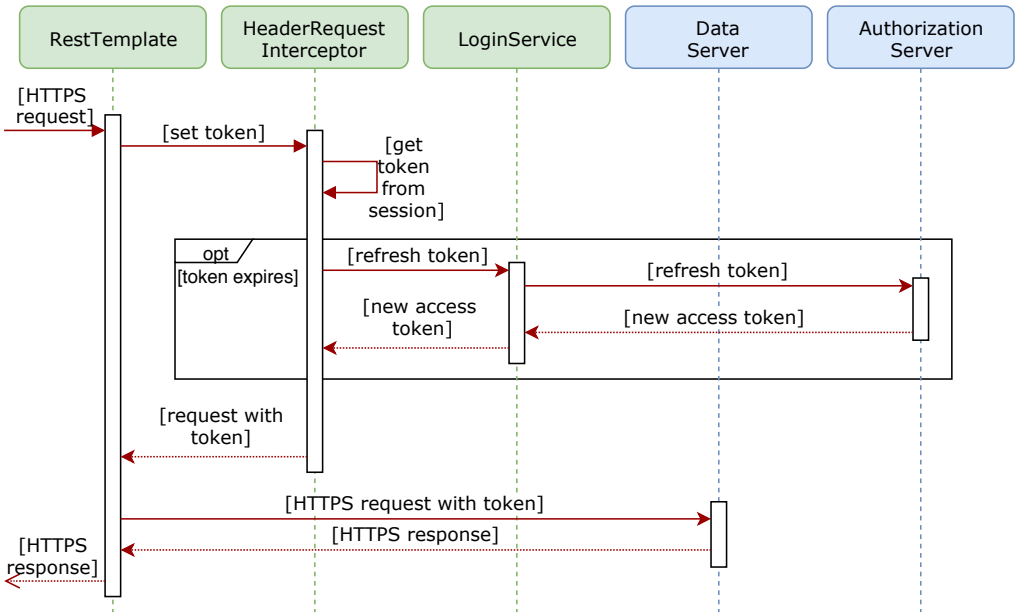
Přihlašovací a registrační obrazovka jsou jako jediné přístupné bez přihlášení uživatele.

8.2 Komunikace se servery

Do této kapitoly spadá veškerá komunikace s datovým serverem a zobrazení, úprava a smazání uživatelského účtu prostřednictvím autorizačního serveru.

Se servery aplikace komunikuje pomocí třídy `RestTemplate`. Zprávy jsou odesílány prostřednictvím HTTPS protokolu a data jsou přenášena ve formátu JSON.

Jelikož datový server vyžaduje ověření pomocí OAuth tokenu, je v aplikaci využívána vlastní třída `HeaderRequestInterceptor` pro automatické doplňovaní hlavičky `Authorization`. Sekvenční diagram odeslání požadavku včetně přidání hlavičky je na obrázku 8.1. Autorizační token je získán ze session, do které byl uložen během přihlašování. Pokud vypršela platnost tokenu, je odeslán požadavek do autorizačního serveru na obnovu tokenu. Získaný token je opět uložen do session. Aktuální token je přidán do požadavku, který je odeslán na server.



Obrázek 8.1: Sekvenční diagram automatického správy tokenu v požadavcích za použití třídy `HeaderRequestInterceptor`.

Stejně jako u ostatních serverů i zde je použito globální zachycení výjimek ze všech kontrolerů pomocí třídy `CustomErrorAttributes`. Zachycená výjimka je vypsána do logu a uživateli je zobrazena zpráva s chybovou hláškou.

8.3 Thymeleaf

Pro vytváření webových stránek je použit šablonovací framework `Thymeleaf`. Díky tomu je možné dynamicky vytvářet HTML stránky na straně serveru, které jsou následně zobrazeny uživateli. Značnou výhodou je automatická

validace vkládaných dat z databáze. Jakýkoliv editovatelný text na stránce, jako je například název auta, nemůže rozbít vzhled stránky vložením HTML tagu. V aplikaci se nachází celkem devět nezávislých stránek:

car - zobrazení seznamu aut,

event - zobrazení seznamu událostí,

routes - zobrazení seznamu cest,

route - zobrazení podrobností o cestě,

status - stránka se aktuální statusem všech připojených zařízeních,

login - přihlašovací obrazovka,

register - obrazovka pro registraci uživatelů,

setting - stránka s uživatelským nastavením,

error - zobrazení chybového stavu.

Dále jsou využity takzvané fragmenty. Jedná se o část stránky, která se vkládá do většího celku. Příkladem je navigační panel, který je stejný na všech stránkách. Místo toho aby byl definován v každé šabloně, je vytvořen pouze jednou jako fragment, který je následně v každé šabloně vložen. Kromě prostého vložení elementů je možné fragment vložit a zároveň jej rozšířit o další elementy. Tento postup je využíván pro vložení HTML hlavičky do všech stránek. Ve fragmentu jsou definované základní parametry a odkazy na skripty a kaskádové styly. V každé stránce je poté možno doplnit hlavičku o další prvky dle potřeby. V aplikaci je používáno celkem sedm fragmentů:

event - představuje jednu událost na stránce,

status - představuje jeden status na stránce,

filter - hlavička seznamů, které se dají filtrovat podle aut,

head - HTML hlavička všech stránek,

navigation - navigační panel všech stránek,

error toast - lišta zobrazující chybovou hlášku,

modal delete - modální okno pro odstranění.

8.4 JavaScript

V klientské části aplikace je použit JavaScript převážně pro komunikaci se serverem a pro vizualizaci cesty viz kapitola 8.5. Pro odesílání požadavků je použita knihovna `JQuery`. Kromě toho je `JavaScript` využíván pro vytvoření modálních oken pomocí knihovny `Bootstrap`.

Jednotlivé funkce jsou rozděleny do dvou souborů, kdy soubor `route.js` obsahuje veškeré funkce pro vytvoření mapy a grafů na stránce s cestou, a soubor `app.js` obsahuje veškeré ostatní funkce pro odesílání `ajax` dotazů.

8.5 Zobrazení cesty

Webová aplikace také komunikuje s `Bing Maps API` pro zobrazení interaktivní mapy s ujetou trasou. Komunikace probíhá na klientské straně a tím pádem není zatěžován server. Integrace interaktivní mapy byla vytvořena v rámci oborového projektu KIV/OPSWI.

Stránka s cestou také obsahuje grafy s rychlostí vozidla a nadmořskou výškou. Pro jejich realizaci je použita knihovna `Chart.js`. `Chart.js` je jednoduchá Open Source knihovna pro vytváření grafů pomocí JavaScriptu. Funkcionalita knihovny zcela vyhovuje potřebám aplikace.

Během vytváření vzhledu jsou pomocí `Thymeleaf` vloženy do stránky veškeré pozice z cesty. Ty jsou při načtení zpracovány pomocí JavaScriptu a uloženy pro pozdější užití. Následně je spuštěna inicializace grafů, která použije předzpracované trasy. Při načtení stránky je spuštěn skript pro získání potřebných dat z `Bing Maps API`. Po načtení dat je automaticky zavolána JavaScript funkce, která inicializuje mapu a vloží do ní předzpracované pozice.

9 Zabezpečení systému

Při vývoji aplikací, obzvláště těch, které komunikují přes internet, by měl být kladen důraz na zabezpečení. V praxi to však vždy neplatí a nejednou se stalo, že uživatelská data byla zneužita. Příkladem může být společnost Uber, kdy vývojáři omylem nahráli přihlašovací údaje k databázi do veřejně přístupného účtu na GitHubu[1] [27]. Téměř s jistotou můžeme říci, že žádná aplikace není stoprocentně bezpečná. Existuje však několik běžných bezpečnostních chyb, jejichž prevencí snížíme riziko na minimum. V této kapitole budou dále popsána tato základní bezpečnostní rizika a jak jsou tyto chyby v navrhovaném systému eliminovány. Nakonec budou zmíněna další použitá bezpečnostní opatření. [28]

9.1 Uživatelské vstupy

Jakýkoliv vstup do aplikace, který není validován představuje bezpečnostní riziko. Samozřejmě ovšem záleží jak bude s daty dále nakládáno. Častou chybou bývá validace vstupů formuláře pouze v klientské části aplikace. Útočník však může formulář obejít a odeslat data přímo na server. Proto je nutné na serveru validovat veškerá data od uživatele. Typickými příklady jsou SQL Injection a vložení HTML kódu. SQL injection je vložení SQL příkazu, například místo uživatelského jména ve formuláři. SQL příkaz je následně v databázi spuštěn, což představuje velké bezpečnostní riziko. Vložení HTML kódu může naopak poškodit vzhled zobrazované webové stránky, či přesměrovat uživatele na jiné webové stránky. Validace vstupů by měla být co nejpřísnější. Například věk uživatele nebude čtyřciferné číslo proto by vstup měl být shora i zdola omezen.

V systému je pro zabránění SQL injection použit `Spring Repository`, který dále využívá JPA. Veškeré uživatelské vstupy jsou do SQL dotazů vkládány pomocí parametrizovaných dotazů, díky kterým je eliminováno riziko SQL injection. Pro zabránění úpravě HTML kódu jsou použity vnitřní konstrukce frameworku `Thymeleaf` (viz kapitola 8.3).

[1] Webová služba sloužící převážně pro verzování zdrojového kódu softwaru

9.2 Přenos dat

Během komunikace po síti mohou být data zachytávána a čtena třetí stranou. Tento útok se nazývá Man-In-The-Middle. Obranou proti tomuto druhu útoků je použití šifrovaného spojení. Příkladem může být použití protokolu HTTPS [2]. Zabezpečený protokol HTTPS je použit pro veškerou komunikaci v systému:

- Komunikace uživatele s webovou aplikací.
- Komunikace webové aplikace s datovým serverem.
- Komunikace mezi datovým serverem a Android aplikací.
- Komunikace z webové a Android aplikace s autorizačním serverem.
- Ověřování identity uživatele v datovém serveru vůči autorizačnímu serveru.
- Odesílání zpráv z datového serveru do Android aplikace přes Firebase.
- Interakce s `Bing Maps API`.

9.3 Autentizace uživatele

Přihlašovací údaje uživatele by nikdy neměly být uložené v databázi v čitelné podobě. Při prolomení systému dostane útočník přihlašovací údaje všech uživatelů v systému. Pokud je nutné uložit heslo, mělo by být v šifrované podobě. Přihlášení poté probíhá tak, že zadané heslo se také zašifruje a porovná se s uloženým zašifrovaným heslem.

Pro autentizaci je vytvořen vlastní autentizační server. Server používá OAuth 2.0 protokol, který je vhodný pro přístup přes API, které je implementováno mezi Android aplikací a aplikačním server. Autorizační server využívá Spring a Spring Security. Uživatelská hesla jsou v databázi šifrována pomocí hašovací funkce Bcrypt. Ostatní serverové aplikace také využívají Spring a Spring Security pro zabezpečení.

[2] Protokol HTTPS je zabezpečenou verzí protokolu HTTP, který se široce používá pro komunikace přes internet.

9.4 OAuth 2.0

V jednoduché klient server aplikaci přistupuje klient k zabezpečeným datům pomocí uživatelského jména a hesla. Potřebuje-li tedy aplikace komunikovat se serverem dlouhodobě, musí mít uložené uživatelské přihlašovací údaje uložené v prostém textu. Tento způsob tedy přináší značné riziko, zejména pokud se jedná o aplikaci třetí strany.

V následující části bude popsán OAuth 2.0 ve velice zjednodušené podobě jen pro pochopení základních informací. OAuth poskytuje možnost přístupu k zabezpečeným datům bez nutnosti zadávat heslo do aplikace třetí strany a bez nutnosti ukládání uživatelského hesla v aplikaci. Systém je založený na existenci třetího hráče, kterým je autorizační server. Přes autorizační server se uživatel přihlásí a obdrží přístupový token. Tento token je předán klientské aplikaci, která pomocí něj přistupuje k zabezpečeným údajům. Server, který spravuje zabezpečené údaje, ověří uživatele taktéž prostřednictvím autorizačního serveru. Po vypršení platnosti tokenu ztrácí klientská aplikace přístup k zabezpečeným datům.

OAuth 2.0 poskytuje celkem čtyři různé možnosti přihlášení, které jsou vhodné pro různé případy. V následující části budou jednotlivé části stručně popsány.

Authorization Code - Tento přístup je nejtěžší a nejbezpečnější z níže uvedených. Klientská aplikace přesměruje uživatele na autorizační server, kde se uživatel přihlásí. Po přihlášení je uživatel přesměrován zpět na klientskou aplikaci. Pro získání tokenu musí klientská aplikace opět kontaktovat autorizační server.

Implicit - Podobný přístup jako Authorization Code s tím rozdílem, že při přesměrování z autorizačního serveru na klientskou aplikaci je rovnou předán přístupový token.

Password - V tomto přístupu uživatel zadá své přihlašovací údaje přímo do klientské aplikace, která se pomocí nich přihlásí k autorizačnímu serveru. Tento způsob je vhodný pouze pro důvěryhodné aplikace, tedy aplikace, které nebyly vytvořeny třetí stranou.

Client Credentials - Klientská aplikace vlastní své přihlašovací údaje, se kterými se přihlásí k autorizačnímu serveru. Tento způsob se nevyužívá pro přístup k uživatelským datům, ale k datům týkajících se pouze klientů.

Kromě přístupového tokenu exituje také refresh token, který má delší platnost než přístupový token. Po vypršení platnosti přístupového tokenu se použije jenom refresh token, kterým lze získat nový přístupový token. [32]

V systému je využíváno dvou metod pro přihlášení. Metoda `Client Credentials` je využívána webovou aplikací pro registraci nových uživatelů, protože v danou chvíli uživatel nemá přihlašovací údaje. Pro všechny další případy je využívána metoda `Password`. Jelikož jak webová, tak Android aplikace jsou důvěryhodné aplikace, a nebyly vytvořeny třetí stranou, není zde riziko záměrného ukládání přihlašovacích údajů, je možné tuto metodu použít.

10 Ověření funkcionality

Pro nalezení chyb v implementaci systému byly použity programátorem psané testy a byly vytvořeny testovací scénáře pro ověření funkcionality. Dále byl systém testován v reálném provozu. Testování probíhalo na třech zařízeních s verzí Androidu 4.1, 5 a 9.

10.1 Unit testy

Datový server, autorizační server a webová aplikace byly otestovány pomocí jednotkových testů. Pro vytvoření jednotkových testů byly použity frameworky `JUnit 5` a `Mockito`. První zmíněný framework slouží pro vytvoření a spouštění jednotkových testů. Druhý framework umožňuje podvržení instance v testované třídě. Například během testování `kontroleru` jsou podvrženy třídy `repozitářů`, jelikož nechceme volat skutečnou databázi. Díky tomu dokážeme otestovat pouze `kontroler` nezávisle na chybách v `repozitářích`.

Grafické rozhraní webové aplikace nebylo testováno automatizovaně, ale uživatelskými testy.

Android aplikace je silně provázána s hardwarem mobilního telefonu. Z toho důvodu je vytvoření automatizovaných testů nad rámec této práce. Android aplikace byla otestována pomocí uživatelských testů.

10.2 Uživatelské testování

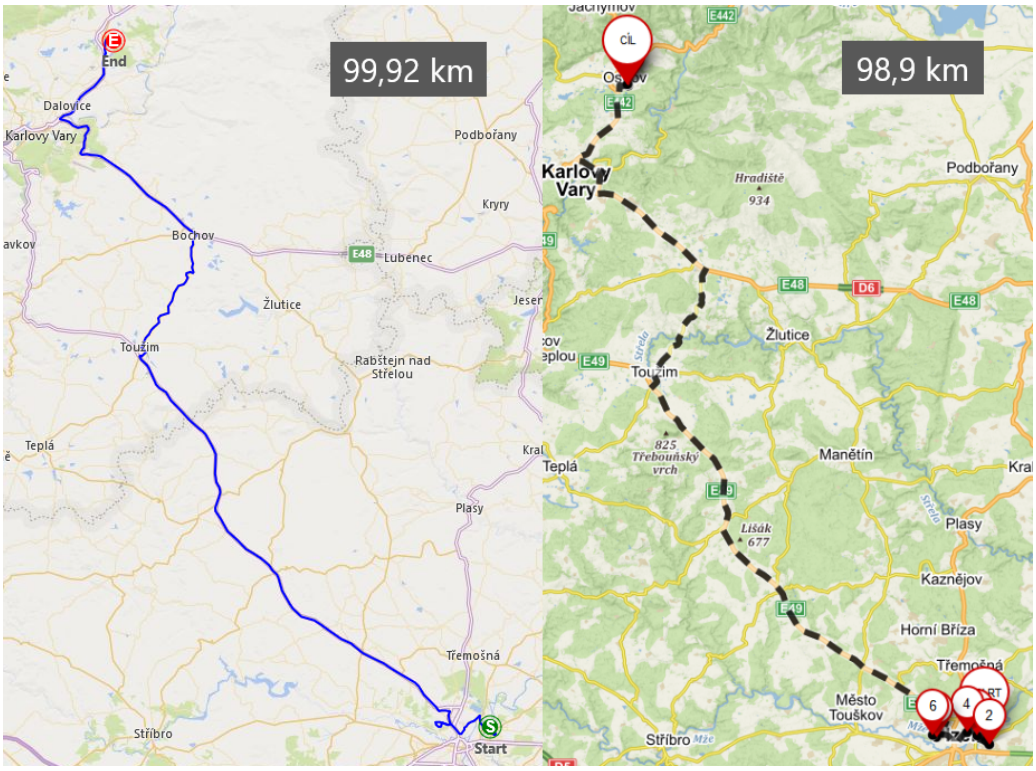
Kromě testů zmíněných v předchozí kapitole, byl systém testován i v reálném provozu. Systém byl využíván dvěma uživateli od poloviny března roku 2019 po dobu dvou měsíců, kdy bylo vytvořeno zhruba sto záznamů v knize jízd.

Pro rozsáhlejší testování byly také vytvořeny testovací scénáře, které se nachází v příloze této práce. V následující části budou popsány výsledky některých vybraných testovacích scénářů.

10.2.1 Správná délka naměřené trasy

Pro porovnání délky naměřené trasy a té skutečné byly využity mapy z portálu `mapy.cz`, které umožňují naplánovat trasu z libovolného množství bodů. Na obrázku 10.1 je vidět porovnání při cestě z Plzně do Ostrova.

Plánovaná trasa byla odhadnuta na 98,9 kilometru. Aplikace však reálně naměřila 99,92 kilometru. Rozdíl jednoho kilometru považuji za přijatelný s ohledem na nepřesnost GPS senzoru v mobilním zařízení, které je navíc z roku 2012.



Obrázek 10.1: Trasa z Plzně do Ostrova. Vlevo trasa získaná pomocí vytvořeného systému s naměřenou délkou trasy 99,92 km. Vpravo trasa plánovaná pomocí `mapy.cz` s odhadovanou vzdáleností 98,9 km.

Druhou testovanou trasou je cesta z Plzně do Pelhřimova. Dle portálu `mapy.cz` by trasa měla být dlouhá 209 kilometrů. Vytvořená mobilní aplikace naměřila trasu o délce 209,38 kilometrů. V tomto případě se jednalo o nový telefon z roku 2018.

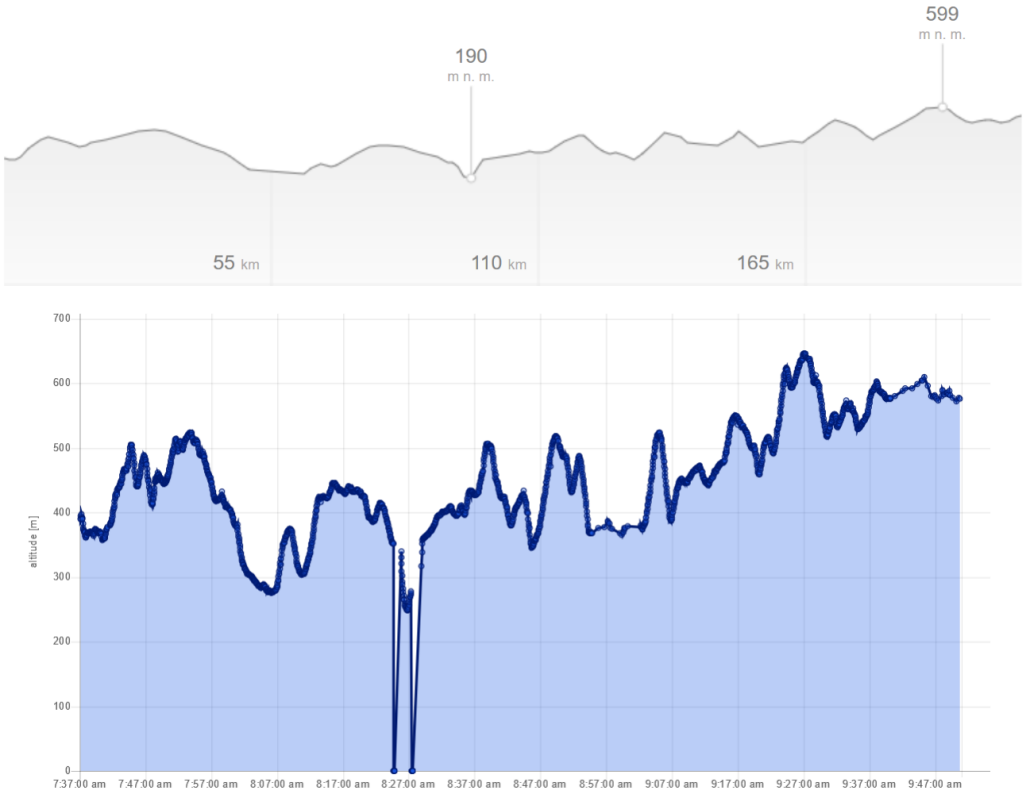
Další trasy již nebyly podrobně porovnávány, ale výsledná vzdálenost vždy odpovídala předpokládané vzdálenosti, a nikdy nebyla vypočtena vzdálenost, která by se značně lišila od té předpokládané.

10.2.2 Správná naměřená nadmořská výška

Výškový profil cesty byl taktéž porovnáván s portálem `mapy.cz`. Zde však nastává menší rozdíl, kdy `mapy.cz` poskytují výškový profil podle kilometrů,

ale webová aplikace zobrazuje výškový profil vozidla v čase. Proto je nutné k tomuto přihlédnout.

Na obrázku 10.2 je porovnání výškového profilu při cestě z Plzně do Pelhřimova. Na první pohled je patrné, že hodnoty získané z Android aplikace jsou zhruba o 50 metrů vyšší, ovšem výškový profil je více méně shodný. Jelikož je údaj o nadmořské výšce získáván z mobilního telefonu, je výsledný graf závislý na správné kalibraci senzoru v mobilním telefonu. Zajímavé je, že stejné nepřesnosti dosahovalo i zařízení z roku 2012 při cestě z Plzně do Ostrova.

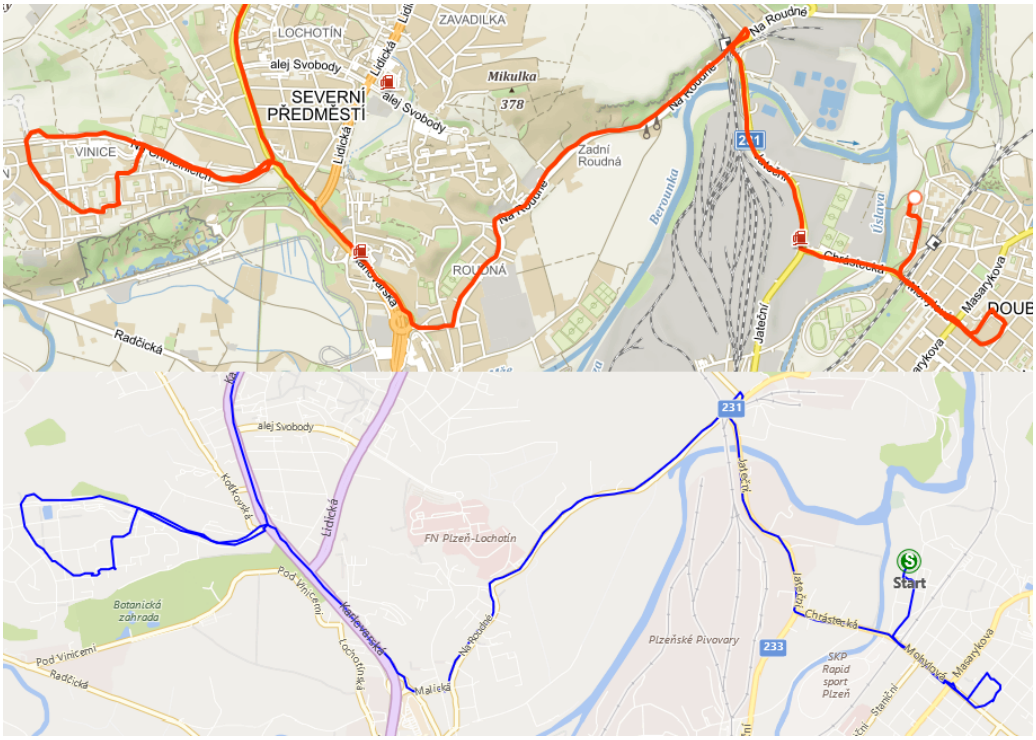


Obrázek 10.2: Výškový profil při cestě z Plzně do Pelhřimova. Horní graf zobrazuje výškový profil získaný z portálu `mapy.cz`. Spodní graf zobrazuje nadmořskou výšku získanou z vytvořeného systému. Chybějící hodnoty jsou způsobeny jízdou v tunelu.

10.2.3 Odpovídající exportovaná trasa

Pro ověření exportované trasy byl opět využit portál `mapy.cz`, který umožňuje nahrání GPX trasy. Na obrázku 10.3 je porovnání náhodně vybrané trasy. Trasa zobrazená ve webové aplikaci zcela odpovídá trase nahrané do

portálu `mapy.cz`, včetně vzdálenosti.



Obrázek 10.3: Porovnání trasy ve webové aplikaci (dolní mapa) s exportovanou trasou nahranou do portálu `mapy.cz` (horní mapa)

11 Návrh rozšíření

Systém tohoto rozsahu je možné neustále rozšiřovat a vytvářet nové funkcionality. V následující části si však popíšeme pouze pár funkcionalit, o které by systém mohl být obohacen, čímž by stoupla jeho užitná hodnota.

Automatický režim - Jedná se o automatické spouštění a vypínání alarmu a vytváření knihy jízd. Pro toto rozšíření je nutné vytvořit detekci majitele automobilu nejlépe pomocí bezdrátové technologie. Nejpřijatelnější by byla identifikace majitele podle mobilního telefonu s aktivovaným Bluetooth nebo použitím Bluetooth klíčenky. Díky rozšíření by nebylo potřeba k mobilnímu zařízení přistupovat a mohl by být umístěn trvale v automobilu. Odpadá také riziko zapomenutí zapnutí alarmu nebo záznamu knihy jízd.

Nastavení Android aplikace z webové aplikace - V případě, že by byl implementován automatický režim, bylo by vhodné vytvořit i kompletní vzdálené nastavení přes webovou aplikaci. Díky tomu by se k zařízení v automobilu nemuselo již vůbec přistupovat a mohlo by být daleko důmyslněji ukryto. Druhou výhodou tohoto rozšíření by bylo možné zálohování nastavení a přenesení do jiného zařízení.

Export pro daňové přiznání - Zajímavým rozšířením by byla možnost exportu knihy jízd ve formátu vhodném pro daňové přiznání. Export by mohl být proveden pro vybrané období. Využití by bylo zejména pro firemní účely.

Zobrazení tras v Android aplikaci - Momentálně slouží Android aplikace pouze jako zařízení pro sběr dat. Do budoucna by stálo za zvážení, vytvořit rozšíření umožňující prohlížení tras v mobilní aplikaci, kde by bylo vidět, zda je trasa uložena pouze lokálně nebo zda je již synchronizovaná se serverem.

12 Závěr

Cílem práce bylo navrhnout a vytvořit systém pro zabezpečení a správu provozu vozidla pomocí Android aplikace. Nejprve proběhlo podrobnější seznámení s platformou Android a s obdobnými existujícími systémy. Na základě získaných informací byl navržen a implementován systém skládající se z Android aplikace pro sběr dat v automobilu, webové aplikace pro správu systému, datového serveru pro ukládání naměřených dat a autorizačního serveru pro zabezpečení přístupu k datům.

Systém obsahuje funkcionalitu alarmu a vytváření knihy jízd. V případě detekce narušitele je uživatel upozorněn pomocí SMS zprávy, hovorem nebo e-mailem. Veškeré události jsou viditelné a zpětně vyhledatelné ve webové aplikaci. Kniha jízd je po připojení zařízení k internetu automaticky synchronizována s datovým serverem. Webová aplikace následně dokáže zobrazit knihu jízd pro jednotlivé automobily a zobrazit detaily každé jízdy.

Systém je napojen na `Firebase`, konkrétně na službu FCM, díky které je možné odesílat příkazy z webové aplikace přímo do Android aplikace v mobilním telefonu uživatele. Webová aplikace komunikuje s `Bing Maps` pro zobrazení interaktivní mapy se zvýrazněnou trasou automobilu.`Bing Maps` byly využity i pro zobrazení náhledu trasy v přehledu všech ujetých tras.

Systém byl otestován pomocí jednotkových testů, uživatelsky otestován pomocí testovacích scénářů a byl reálně využíván po dobu dvou měsíců dvěma uživateli.

V budoucnu by mohl být systém rozšířen o další funkcionality, jako je například zcela automatický režim zapínání a vypínání alarmu a knihy jízd nebo exportu knihy jízd pro daňové přiznání.

Přehled zkratek

AOT Ahead-Of-Time

ART Android Runtime

FCM Firebase Cloud Messaging

HAL Hardware Abstraction Layer

IM Instant Messaging

JDK Java Development Kit

JIT Just-In-Time

JPA Java Persistence API

NDK Native Development Kit

OBD2 On-Board Diagnostics 2

REST Representational State Transfer

SAM Single Abstract Method

SOAP Simple Object Access Protocol

SSE Server-Sent Events

Literatura

[1] Choose your Bing Maps API, [online], . Dostupné z: https://www.microsoft.com/en-us/maps/choose-your-bing-maps-api. (Navštíveno 4.8.2018).

[2] Bing Maps - Help Choosing the Right License, [online], . Dostupné z: https://www.microsoft.com/en-us/maps/licensing/options. (Navštíveno 4.8.2018).

[3] GPS Fleet Tracking System for Phones, [online]. Dostupné z: https://corvusgps.com/. (Navštíveno 30.7.2018).

[4] Here - Documentation, [online], . Dostupné z: https://developer.here.com/documentation. (Navštíveno 4.8.2018).

[5] Here - Plans and Pricing, [online], . Dostupné z: https://developer.here.com/plans. (Navštíveno 4.8.2018).

[6] StatCounter Mobile Operating System Market Share Worldwide, [online]. Dostupné z: http://gs.statcounter.com/os-market-share/mobile/worldwide/. (Navštíveno 29.6.2018).

[7] Milage Tracker App and Expense Reimbursement Cloud - TripLog, [online]. Dostupné z: https://triplogmileage.com/. (Navštíveno 30.7.2018).

[8] Hibernate ORM, [online], . Dostupné z: http://hibernate.org/orm/. (Navštíveno 18.10.2018).

[9] Android developers, [online], . Dostupné z: https://developer.android.com/. (Navštíveno 30.6.2018).

[10] Android developers - Monitor the Battery Level and Charging State, [online], . Dostupné z: https://developer.android.com/training/monitoring-device-state/battery-monitoring. (Navštíveno 27.9.2018).

[11] Android developers - Application Fundamentals, [online], . Dostupné z: https://developer.android.com/guide/components/fundamentals. (Navštíveno 3.7.2018).

[12] Android developers - Save data in a local database using Room, [online], . Dostupné z: https://developer.android.com/training/data-storage/room/. (Navštíveno 17.10.2018).

[13] Android developers - Services overview, [online], . Dostupné z: https://developer.android.com/guide/components/services. (Navštíveno 4.7.2018).

[14] API Design Guide, [online], 2017. Dostupné z: https://realtimeapi.io/hub-category/api-design-guide/. (Navštíveno 2.8.2018).

[15] Google play - Car Security Alarm Pro, [online], . Dostupné z: https://play.google.com/store/apps/details?id=com.eastcoders.caralarm&hl=cs/. (Navštíveno 29.7.2018).

[16] The car security app - Car alarm, [online], . Dostupné z: https://car-alarm.link/. (Navštíveno 29.7.2018).

[17] Advanced Real-Time Car Tracking & Alert System - CarLock, [online], . Dostupné z: https://www.carlock.co/. (Navštíveno 30.7.2018).

[18] Google Cloud Messaging, [online]. Dostupné z: https://developers.google.com/cloud-messaging/. (Navštíveno 3.8.2018).

[19] Firebase Cloud Messaging, [online]. Dostupné z: https://firebase.google.com/products/cloud-messaging/. (Navštíveno 3.8.2018).

[20] Google Maps Platform Documentation, [online], . Dostupné z: https://developers.google.com/maps/documentation/. (Navštíveno 4.8.2018).

[21] PRICING FOR MAPS, ROUTES, AND PLACES, [online], . Dostupné z: https://cloud.google.com/maps-platform/pricing/sheet/. (Navštíveno 4.8.2018).

[22] Introduction to GraphQL, 2019. Dostupné z: https://graphql.org/learn/. (Navštíveno 17.3.2019).

[23] Kotlin Frequently asked questions , [online], . Dostupné z: https://kotlinlang.org/docs/reference/faq.html/. (Navštíveno 2.7.2018).

[24] Comparison to Java Programming Language , [online], . Dostupné z: https://kotlinlang.org/docs/reference/comparison-to-java.html/. (Navštíveno 3.7.2018).

[25] RFC - SOAP Version 1.2, 2007. Dostupné z: https://www.w3.org/TR/soap12/. (Navštíveno 17.3.2019).

[26] Firebase Tutorial, [online]. Dostupné z: https://www.tutorialspoint.com/firebase/index.htm. (Navštíveno 3.8.2018).

[27] Uber's massive hack: What we know, [online]. Dostupné z: https://money.cnn.com/2017/11/22/technology/uber-hack-consequences-cover-up/index.html. (Navštíveno 4.8.2018).

[28] The Basics of Web Application Security, [online], 2017. Dostupné z: https://martinfowler.com/articles/web-security-basics.html. (Navštíveno 4.8.2018).

[29] Ableson, W. F. et al. *Android in Action, Third Edition.* Manning, 2011. ISBN 978-1-61729-050-3.

[30] Bidelman, E. Stream Updates with Server-Sent Events, [online], 2010. Dostupné z: https://www.html5rocks.com/en/tutorials/eventsource/basics/. (Navštíveno 2.8.2018).

[31] Fielding, R. T. Representational State Transfer (REST), 2000. Dostupné z: http://roy.gbiv.com/pubs/dissertation/rest_arch_style.htm. (Navštíveno 17.3.2019).

[32] Hardt, D. RFC - The OAuth 2.0 Authorization Framework, 2012. Dostupné z: https://tools.ietf.org/html/rfc6749#section-1.3.1. (Navštíveno 3.4.2019).

[33] Hildenbrand, J. Everything you need to know about rooting your Android, [online], 2016. Dostupné z: https://m.androidcentral.com/root/. (Navštíveno 2.10.2018).

[34] Kodytek, S. Úvod do jazyka Kotlin, platformy a IntelliJ, [online]. Dostupné z: https://www.itnetwork.cz/kotlin/zaklady/uvod-do-jazyka-kotlin-platformy-a-intellij/. (Navštíveno 2.7.2018).

[35] Logan, P. REST, SOAP, GraphQL — Gesundheit!, [online], 2018. Dostupné z: https://medium.com/postman-engineering/rest-soap-graphql-gesundheit-6544053f65cf. (Navštíveno 17.3.2019).

[36] Shafirov, M. Kotlin on Android. Now official, [online], 2017. Dostupné z: https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/. (Navštíveno 2.7.2018).

[37] Sinhal, A. Closer Look At Android Runtime: DVM vs ART, [online], 2017. Dostupné z: `https://android.jlelse.eu/closer-look-at-android-runtime-dvm-vs-art-1dc5240c3924/`. (Navštíveno 1.7.2018).

[38] Lacko. *Mistrovství - Android*. Computer Press, 2017. ISBN 978-80-251-4875-4.

A Uživatelská příručka Android aplikace

Mobilní aplikace `Car security` slouží pro zabezpečení automobilu a pro vytváření knihy jízd. Aplikace může komunikovat prostřednictvím SMS zpráv, voláním a přes internetové připojení. Není však nutné využívat veškeré možnosti připojení.

A.1 Ovládání aplikace

Po spuštění aplikace se otevře výchozí obrazovka s ovládacími prvky (viz obrázek A.1). Tlačítka uprostřed obrazovky slouží pro aktivaci a deaktivaci nástrojů. Více viz kapitoly A.2 a A.3. Kulaté tlačítko v pravé spodní části obrazovky slouží pro vstup do nastavení aplikace.

Menu aplikace je schováno pod tlačítkem v levém horním rohu, které má podobu třech vodorovných čar. Menu je na obrázku A.2. Je-li uživatel přihlášen, je zobrazeno jeho jméno a vybrané auto v horní části menu. Dále menu obsahuje tlačítka:

Home - Úvodní obrazovka aplikace.

Login - Obrazovka pro přihlášení a odhlášení uživatele. Vyžaduje síťovou komunikaci.

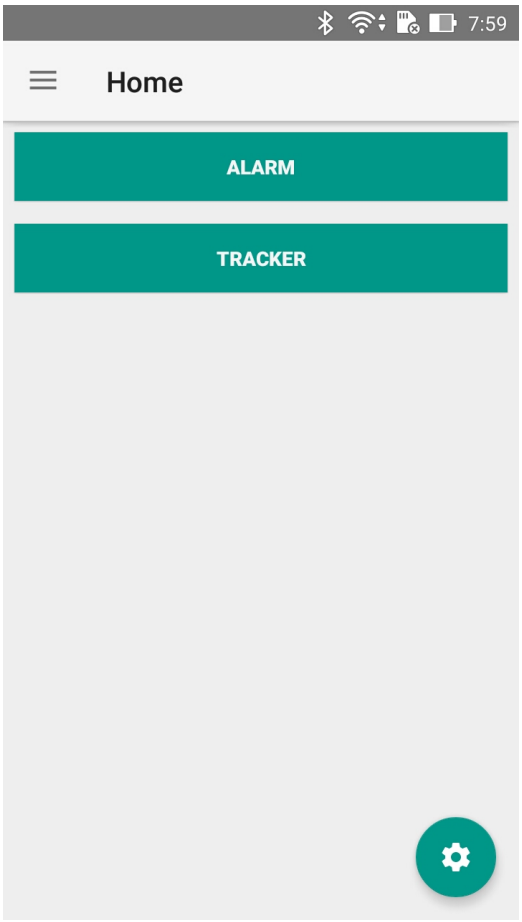
Settings - Kompletní nastavení aplikace.

Status - Zobrazení aktuálního statusu síťové komunikace a seznam nesynchronizovaných záznamů z knihy jízd.

A.2 Alarm

Alarm slouží k zabezpečení vozidla. Pro detekci narušitele může alarm využívat pohybový a zvukový senzor. Aby mohl být alarm spuštěn, je potřeba mít aktivovaný alespoň jeden ze senzorů. Alarm lze spustit či vypnout:

- Z hlavní obrazovky tlačítkem `Alarm`.



Obrázek A.1: Úvodní obrazovka Android aplikace při spuštění aplikace.

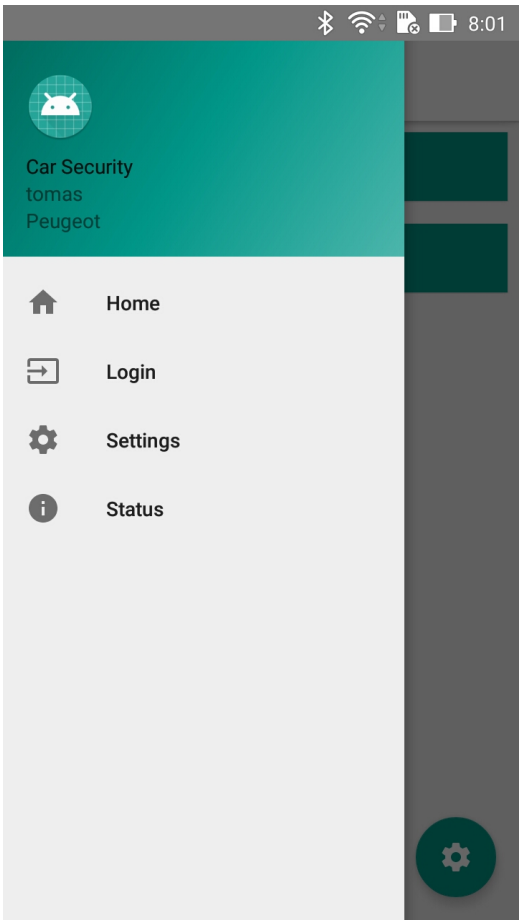
- Pomocí SMS zprávy ve tvaru `activate Alarm` pro spuštění a zprávy `deactivate Alarm` pro vypnutí. Pro SMS komunikaci je potřeba mít povolenou GSM komunikaci v nastavení. Příkaz je přijat pouze od nastaveného telefonního čísla.
- Přes webové prostředí. Je potřeba mít povolenou síťovou komunikaci a být v aplikaci přihlášen.

A.2.1 Upozornění na narušitele

Při spuštění poplachu je několik možností jak systém bude reagovat. Nastavení závisí zcela na uživateli.

Siréna

Aplikace umožňuje při poplachu spustit sirénu, která hraje dokud není alarm vypnut. Tuto možnost využijte pouze v případě připojení externí sirény v



Obrázek A.2: Menu Android aplikace

podobě reproduktoru, který generuje dostatečný hluk.

SMS

Je-li povolena GSM komunikace a je nastaveno kontaktní telefonní číslo, je možné odeslat v případě poplachu SMS zprávu. Aby byla SMS odeslána, je nutné ji v nastavení povolit `Settings => Communication => GSM => Sms types => Send alarm`.

Volání

Pro zavolání v případě poplachu je nutné nastavit telefonní číslo v `Settings => Communication => GSM => Phone number` a povolit hovory v `Settings => Alarm => Make call`. Důležité upozornění: při hovoru je siréna ztlumena.

E-mail

E-mailové upozornění probíhá prostřednictvím systémového serveru. Pro obdržení e-mailového upozornění je tedy nutné být v Android aplikaci přihlášen, mít nastavenou e-mailovou adresu ve webové aplikaci, mít zaškrtnuté políčko `Settings => Communication => Network => Additional communication => Alarm + Alarm position` a samozřejmě mobilní telefon musí být v době poplachu připojen k internetu. Pokud by nebyla zadána e-mailová adresa, událost se pouze zobrazí ve webovém prostředí.

Zprávy o lokaci

V případě poplachu je možné průběžně odesílat informace o lokaci zařízení pomocí e-mailových nebo SMS zpráv. Interval odesílání zpráv lze změnit v `Settings => Alarm => Message location interval`. V případě, že je již nastaveno e-mailové upozornění na poplach, jsou zprávy o lokaci automaticky odesílány také. U SMS zpráv je nutné povolit `Settings => Communication => GSM => Sms types => Send alarm location`.

A.2.2 Nastavení senzorů

Jak bylo zmíněno výše, je pro detekci narušitele použit pohybový a zvukový senzor. Pohybový senzor je dle výchozího nastavení aktivován. Deaktivovat jej lze v `Settings => Sensors => Allow Move sensor`. V případě nedostatečné citlivosti pohybového senzoru (poplach není spuštěn při požadovaném pohybu auta) je možné zvýšit citlivost v `Settings => Sensors => Move sensor => Sensor sensitivity`.

Zvukový senzor lze povolit v `Settings => Sensors => Allow Sound sensor`. Součástí nastavení je i citlivost senzoru `Settings => Sensors => Sound sensor => Sensor sensitivity` a i interval měření `Settings => Sensors => Sound sensor => Measure interval`. Interval měření určuje jak často bude kontrolováno překročení hladiny zvuku za poslední interval. To znamená, že bude-li překročena povolená hladina v polovině intervalu, poplach bude spuštěn až na jeho konci.

A.2.3 Nastavení alarmu

Alarm jako samotný obsahuje také několik nastavení, která jsou v `Settings => Alarm`.

Allow Alarm - Povolení či zakázání použití alarmu v aplikaci.

Start interval - Interval před spuštěním činnosti alarmu. Během tohoto intervalu nelze spustit poplach. Jedná se o dobu určenou pro opuštění vozidla po aktivaci alarmu.

Alert interval - Interval po zaznamenání narušitele než bude spuštěn poplach. Jedná se dobu určenou pro deaktivaci alarmu při nastoupení do vozidla.

Message location interval - Interval odesílání zpráv o lokaci při poplachu.

Make call - Povolení či zakázání hovorů při poplachu.

Siren - Povolení či zakázání sirény při poplachu.

A.3 Kniha jízd

Zaznamenávání knihy jízd je povoleno pouze pokud je povolena síťová komunikace a uživatel je přihlášen v Android aplikaci. Během zaznamenávání knihy jízd není potřeba internetové připojení, a data jsou ukládána v zařízení, dokud se zařízení opět nepřipojí k internetu například prostřednictvím Wi-Fi sítě. Pro zaznamenávání polohy je také nutné v aplikaci povolit přístup k lokaci zařízení. Přístup k lokaci lze povolit v `Settings => Sensors => Allow Location sensor`. Nastavení senzoru také obsahuje interval získávání jednotlivých pozic. Čím delší interval, tím bude zobrazení trasy na mapě méně přesné.

Zaznamenávání knihy jízd lze stejně jako alarm spustit několika způsoby:

- Z hlavní obrazovky tlačítkem `Tracker`.
- Pomocí SMS zprávy ve tvaru `activate Tracker` pro spuštění a zprávy `deactivate Tracker` pro vypnutí. Pro SMS komunikaci je potřeba mít povolenou GSM komunikaci v nastavení. Příkaz je přijat pouze od nastaveného telefonního čísla.
- Přes webové prostředí. Je potřeba mít povolenou síťovou komunikaci a být v aplikaci přihlášen.

Je-li zaznamenávání trasy spuštěno, obsahuje spouštěcí tlačítko informaci o ujeté vzdálenosti v rámci aktuální trasy. Zaznamenané trasy, které zatím nebyly synchronizovány se serverem a nejsou tedy přístupné přes webové prostředí, jsou vypsány na obrazovce `Status` přístupné přes menu aplikace.

A.3.1 Nastavení knihy jízd

Nastavení vytváření knihy jízd je v `Settings => Tracker`.

Allow Tracker - Povolení či zakázání použití knihy jízd v aplikaci.

Not moving distance - Minimální vzdálenost od poslední zaznamenané pozice, která je potřeba urazit, aby byla pozice uložena. Senzor zařízení není přesný a při nastavení nulové vzdálenosti se vozidlo jeví jako v neustálém pohybu.

Timeout - Interval, po kterém je zaznamenávání vypnuto, pokud se vozidlo přestalo pohybovat.

A.3.2 Problém se záznamem trasy

Není-li možné zaznamenat trasu, a při jízdě ukazuje spouštěcí tlačítko nulovou vzdálenost, lze předpokládat, že nelze získat polohu zařízení. Problém může být způsoben nastavením telefonu. V nastavení telefonu přejděte do nastavení polohy a zapněte vysokou přesnost získávání polohy.

A.4 Přihlášení do aplikace

Přihlášení probíhá na obrazovce `Login` přístupné přes menu aplikace. Pro přihlášení je nutné mít povolenou síťovou komunikaci a být připojen k internetu povoleným způsobem (komunikace přes mobilní data může být zakázána viz kapitola A.7.4). Přihlašovací proces:

1. Zadání správných přihlašovacích údajů a stisknutí tlačítka `LOGIN`.
2. Proběhne-li přihlášení v pořádku, zobrazí se seznam aut vlastněných uživatelem. Je nutné auto vybrat nebo stisknout tlačítko `CREATE NEW CAR`.
3. V případě výběru existujícího automobilu, je proces přihlášení kompletní a zobrazí se obrazovka pro odhlášení uživatele.
4. V případě výběru možnosti vytvoření nového automobilu se otevře okno pro zadání názvu nového automobilu. Je-li automobil po potvrzení úspěšně vytvořen na serveru, přihlašovací proces je kompletní a zobrazí se obrazovka pro odhlášení uživatele.

Odhlášení uživatele probíhá na stejné obrazovce jako přihlášení. Pro odhlášení stačí stisknout tlačítko `LOGOUT`. Pozor, při odhlášení se smažou veškerá pořízená data, která ještě nebyla nahrána na server. Je doporučeno před odhlášením provést synchronizaci se serverem.

A.5 Ovládání prostřednictvím SMS

Aplikaci lze ovládat vzdáleně pomocí SMS zpráv. Pro vzdálené ovládání je nutné mít vyplněné kontaktní telefonní číslo v `Settings => Communication => Phone number`, mít povolenou GSM komunikaci v `Settings => Communication => Allow GSM communication` a povoleny příkazy v `Settings => Communication => Sms types => Command ...`

Command tool switch - Aktivuje příkazy pro vzdálené zapnutí a vypnutí alarmu a knihy jízd.

Command send status - Příkaz na získání aktuálního stavu aplikaci (seznam zapnutých nástrojů, stav baterie, stav nabíjení, stav power save módu).

Příkazy jsou provedeny pouze, pokud byli odeslány z kontaktního telefonního čísla. Povoleny jsou následující textové příkazy:

activate Alarm - Zapne alarm.

deactivate Alarm - Vypne alarm.

activate Tracker - Zapne zaznamenávání knihy jízd.

deactivate Tracker - Vypne zaznamenávání knihy jízd.

info - Zařízení odešle zpět zprávu o aktuálním stavu.

A.6 Power save mód

Je-li mód úspory elektrické energie povolen, aktivuje automaticky při poklesu stavu baterie pod hladinu nastavenou v `Settings => Power-save mode => Critical level`. Při poklesu baterie pod nastavenou mez se aplikace přepne do úsporného režimu s latencí až jedné minuty. Povolení a zakázání módu je v `Settings => Power-save mode => Allow power save mode`. Mód úspory elektrické energie provede při své aktivaci následující akce:

- Siréna je deaktivována.

- Zvukový senzor alarmu je zakázán.
- Interval mezi zprávami s pozicí při alarmu je prodloužen na 10 minut.
- Kniha jízd je zakázána.
- Synchronizace nahraných tras se serverem je zakázána.

A.7 Nastavení aplikace

V této části bude popsáno kompletní nastavení aplikace po kategoriích, které jsou v aplikaci.

A.7.1 Alarm

Allow Alarm - Povolení či zakázání použití alarmu v aplikaci.

Start interval - Interval před spuštěním činnosti alarmu. Během tohoto intervalu nelze spustit poplach. Jedná se o dobu určenou pro opuštění vozidla po aktivaci alarmu.

Alert interval - Interval po zaznamenání narušitele než bude spuštěn poplach. Jedná se dobu určenou pro deaktivaci alarmu při nastoupení do vozidla.

Message location interval - Interval odesílání zpráv o lokaci při poplachu.

Make call - Povolení či zakázání hovorů při poplachu.

Siren - Povolení či zakázání sirény při poplachu.

A.7.2 Tracker

Allow Tracker - Povolení či zakázání použití knihy jízd v aplikaci.

Not moving distance - Minimální vzdálenost od poslední zaznamenané pozice, která je potřeba urazit, aby byla pozice uložena. Senzor zařízení není přesný a při nastavení nulové vzdálenosti se vozidlo jeví jako v neustálém pohybu.

Timeout - Interval, po kterém je zaznamenávání vypnuto, pokud se vozidlo přestalo pohybovat.

A.7.3 Sensors

Move sensor

Senzor použitý pro detekci narušitele při alarmu.

Allow Move sensor - Povolení či zakázání použití senzoru.

Sensor sensitivity - Nastavení citlivosti pohybového senzoru. Menší hodnota znamená vyšší citlivost.

Sound sensor

Senzor použitý pro detekci narušitele při alarmu.

Allow Sound sensor - Povolení či zakázání použití senzoru.

Sensor sensitivity - Nastavení citlivosti zvukového senzoru. Menší hodnota znamená vyšší citlivost.

Measure interval - Interval měření hladiny zvuku. To znamená, že bude-li překročena povolená hladina v polovině intervalu, poplach bude spuštěn až na jeho konci. Delší interval šetří baterii zařízení.

Location sensor

Senzor je použitý pro vytváření knihy jízd.

Allow Location sensor - Povolení či zakázání použití senzoru.

Location update interval - Nastavení intervalu snímání aktuální polohy zařízení. Čím delší interval, tím bude zobrazení trasy na mapě méně přesné.

Sensor accuracy - Požadovaná přesnost senzoru. Při použití knihy jízd je doporučeno používat pouze `High accuracy`.

A.7.4 Communication

GSM

Allow GSM communication - Povolení či zakázání komunikace prostřednictvím SMS zpráv a voláním.

Phone number - Nastavení kontaktního telefonního čísla, na které budou odesílána upozornění a od kterého budou přijímány příkazy.

Sms types - Nastavení, která upozornění budou odesílána a které příkazy budou přijímány:

- Alarm send tool switch - Notifikace o zapnutí či vypnutí alarmu
- Trakcer send tool switch - Notifikace o zapnutí či vypnutí vytváření knihy jízd.
- Send alarm - Notifikace poplachu.
- Send alarm location - Průběžné zprávy s polohou při poplachu.
- Send battery notification - Notifikace o přechodu do úsporného módu.
- Send power connected - Notifikace o připojení k externímu zdroji energie.
- Send power disconnected - Notifikace o odpojení od externího zdroje energie.
- Command tool switch - Přijímání příkazů pro zapnutí či vypnutí alarmu a knihy jízd.
- Command send status - Přijímání příkazu pro odeslání aktuálního stavu zařízení.

Network

Allow network communication - Povolení či zakázání komunikace prostřednictvím internetu.

Server url - URL adresa serveru se kterým aplikace komunikuje. Výchozí adresa je `https://carsecurity.mooo.com`.

Mobile data synchronization - Povolení komunikace se serverem prostřednictvím mobilních dat.

Additional communication - Nastavení, která upozornění budou odesílána na server.

- Tool state changed - Zapnutí či vypnutí alarmu nebo knihy jízd.
- Alarm + Alarm position - Notifikace poplachu a průběžné odesílání polohy během poplachu.
- Battery state changed - Notifikace o připojení nebo odpojení zařízení od elektrické energie a upozornění na přechod do úsporného módu.

Synchronization interval - Interval určuje, jak často se bude aplikace synchronizovat se serverem.

A.7.5 Power-save mode

Allow power save mode - Povolení či zakázání přechodu do úsporného režimu.

Critical level - Maximální hladina baterie při které je aplikace v úsporném režimu. Při vzestupu nad nastavenou mez se úsporný režim automaticky vypne. Úsporný režim je nezávislý na tom, zda se zařízení nabíjí či nikoliv.

B Uživatelská příručka Webové aplikace

Webová aplikace slouží převážně pro správu dat získaných pomocí Android aplikace, bez které nemá použití webové aplikace význam. Dále musí být v Android aplikaci povolena síťová komunikace a uživatel musí být v aplikaci přihlášen. Webová aplikace je dostupná na adrese:
`https://carsecurity.mooo.com`.

B.1 Správa uživatele

Registrace

Před přihlášením musí být uživatel již registrován do systému. Registrace se provádí na adrese `/register`. Na stránku registrace se lze dostat také tlačítkem `Register` z přihlašovací stránky.

Uživatelské heslo musí být minimálně 8 znaků dlouhé. Při zadání kratšího hesla nebude uživatel registrován.

Přihlášení

Přihlašovací obrazovka se nachází na adrese `/login`. Přihlašovaný uživatel musí být v systému registrován a přihlášení probíhá pomocí uživatelského jména a hesla.

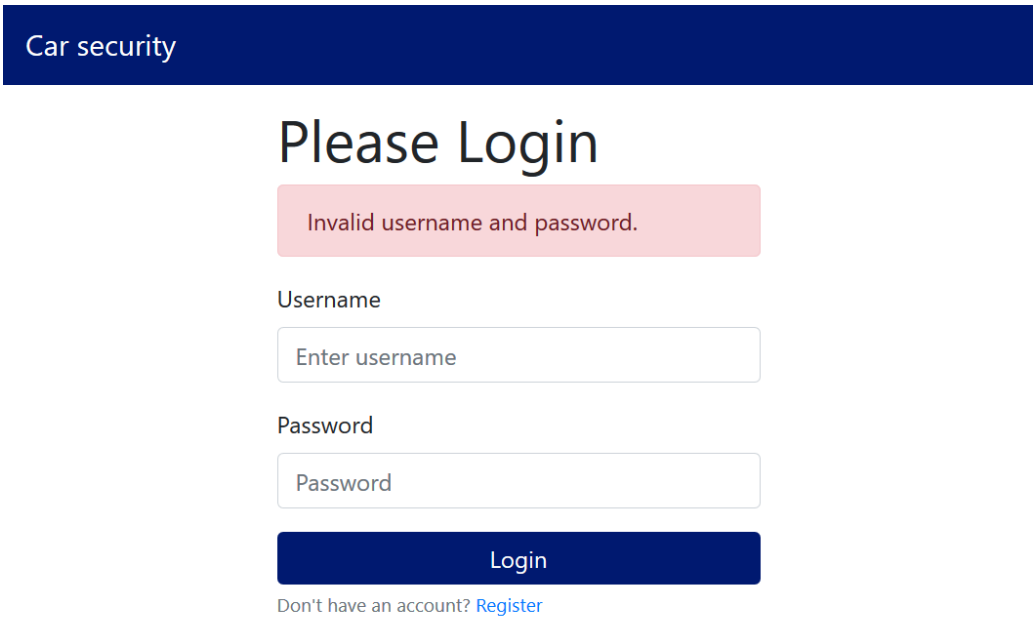
V případě chybně zadaných přihlašovacích údajů se zobrazí zpráva s chybou (viz obrázek B.1).

Odhlášení

Přihlášený uživatel je automaticky odhlášen po půl hodině nečinnosti. Pro okamžité odhlášení je možné použít tlačítko `Logout` umístěné v navigačním panelu (viz kapitola B.2).

Nastavení uživatele

Veškerá nastavení uživatele, která jsou systémem umožněna, se nacházejí na stránce `/settings`.



Obrázek B.1: Přihlašovací obrazovka webové aplikace

Změna e-mailové adresy - E-mailovou adresu pro příjem notifikací lze změnit přepsáním hodnoty v textovém poli `e-mail` a následným stiskem tlačítka `Update e-mail`. V případě problémů se zobrazí chybová zpráva.

Změna přihlašovacího hesla - Změna hesla se vyvolá stisknutím tlačítka `Update password`. V nově otevřeném okně vyplňte původní a nové heslo. Po potvrzení je heslo změněno. V případě problémů se na obrazovku vypíše chybová zpráva.

Odstranění uživatele

Uživatelský účet lze odstranit ze stránky `/settings` pomocí tlačítka `Delete User`. Před odstraněním se zobrazí výzva k potvrzení odstranění.

S uživatelským účtem se odstraní veškeré vytvořené záznamy a uživatelské údaje již nepůjdou obnovit.

B.2 Navigace v aplikaci

Pro navigaci v aplikaci slouží navigační panel v horní části obrazovky viz obrázek B.2. Ovládací prvky navigačního panelu se zobrazí až po přihlášení a jsou po celou dobu přihlášení neměnné. Tlačítka navigačního panelu:

Obrázek B.2: Navigační panel webové aplikace na běžných monitorech.

Car security - Úvodní obrazovka se statusem připojených zařízení.

Events - Seznam událostí ze všech aut.

Cars - Seznam vlastněných automobilů.

Routes - Seznam všech vytvořených cest.

Settings - Uživatelské nastavení.

Logout - Odhlásí aktuálně přihlášeného uživatele a zobrazí přihlašovací obrazovku.

Na menších obrazovkách je zobrazen rozbalovací navigační panel viz obrázek B.3. Pro rozbalení a sbalení stiskněte tlačítko v pravé části navigačního panelu s ikonou třech vodorovných čar.

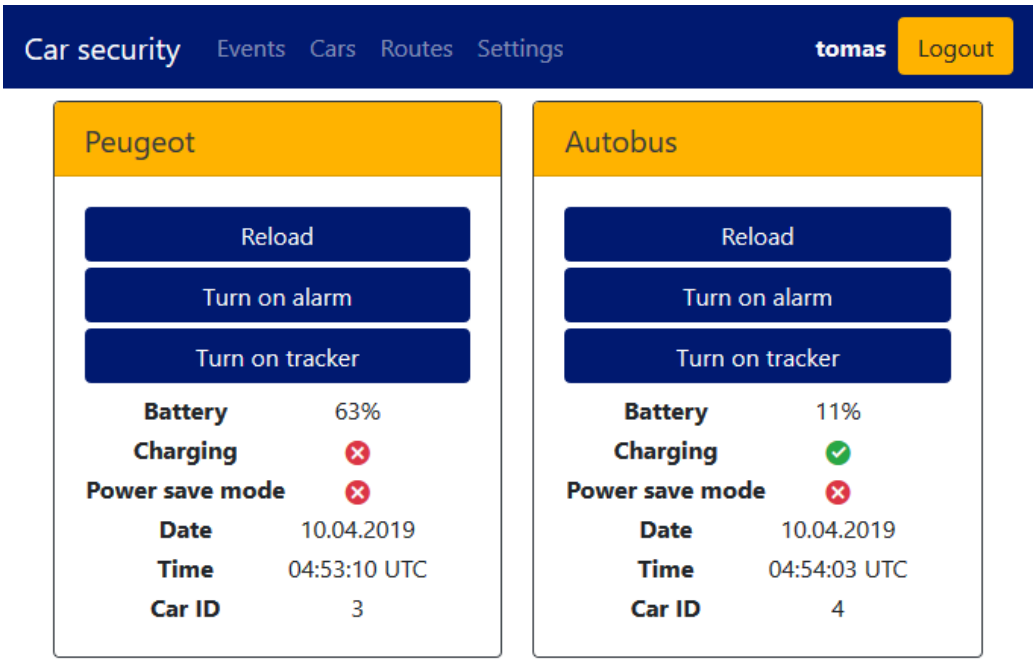
Obrázek B.3: Navigační panel webové aplikace na mobilních zařízení a malých obrazovkách.

B.3 Status připojených zařízení

Obrazovka se nachází na adrese `/` a obsahuje informace o aktuálně přihlášených zařízeních. Na obrazovce se nachází seznam aut, je-li k autu přihlášeno zařízení, které je aktuálně připojené k internetu, zobrazí se informace o tomto zařízení viz obrázek B.4. Nelze-li navázat spojení se zařízením, zobrazí se informační zpráva. Status každého zařízení obsahuje:

- Procentuální stav baterie
- Indikaci nabíjení
- Indikaci power save módu
- Datum získání statusu
- Čas získání statusu
- Identifikační číslo automobilu

Kromě statusu lze u připojeného zařízení zapínat či vypínat nástroje tlačítky `Turn on alarm`, `Turn off alarm`, `Turn on tracker` a `Turn off tracker`. Je-li nástroj zapnutý, má tlačítko zelenou barvu.



Obrázek B.4: Obrazovka se statusem připojených zařízení

B.4 Správa aut

Na adrese `/car` se nachází obrazovka obsahující seznam vlastněných aut. Nové auto lze přidat stisknutím tlačítka `+`. Pro úpravu automobilu stiskněte tlačítko s ikonou pera v pravém horním rohu každého automobilu. Zde je možné upravit název a poznámku automobilu. Kromě úprav se zde nachází i tlačítko pro smazání vozidla. S vozidlem se smažou i vytvořené trasy a události.

B.5 Události

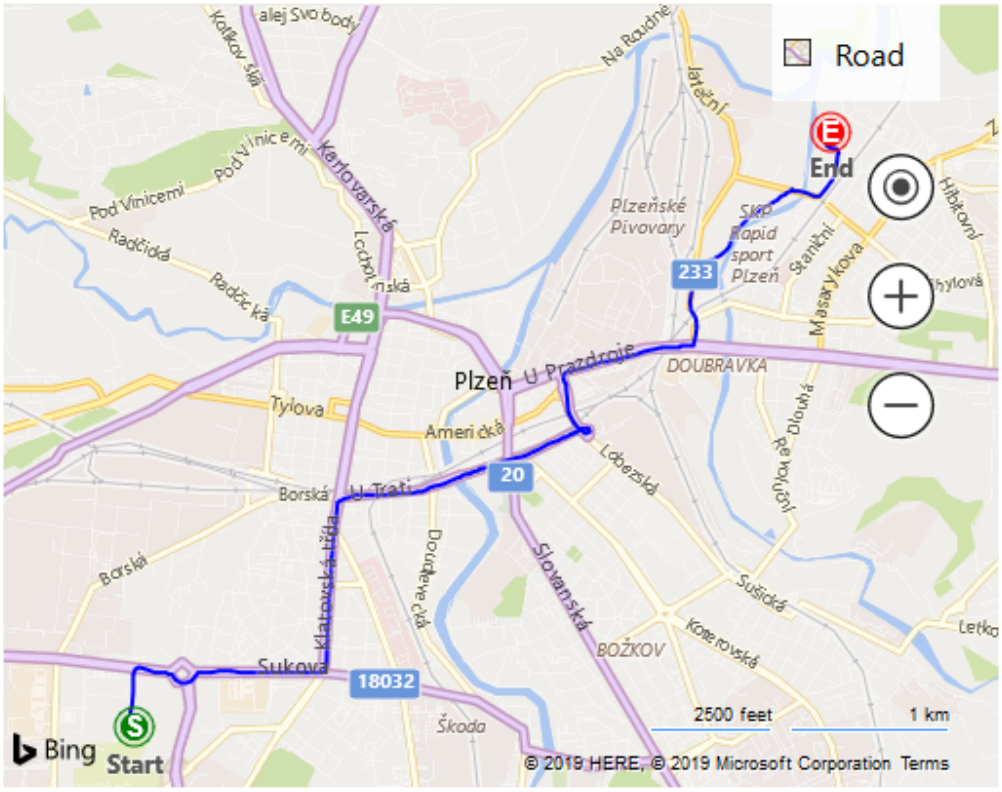
Události nalezneme na adrese `/event`. Zde se nachází veškeré události, které jsou povoleny sdílet v Android aplikaci. U každé události je možné upravit poznámku tlačítkem pera v pravém horním rohu. Pomocí tlačítka `Filter by Car` lze filtrovat zobrazené události dle jednotlivých automobilů. Stejným tlačítkem lze filtr deaktivovat. Události, které se zde mohou nacházet:

- Připojení/odpojení od elektrické energie.
- Spuštění/ukončení vytváření knihy jízd.
- Spuštění/ukončení alarmu.
- Spuštění poplachu.
- Informace o poloze při poplachu.
- Přepnutí do power save módu.

B.6 Kniha jízd

Přehled knihy jízd je na adrese `/routes`. Každá trasa obsahuje přehled a náhled trasy. Pro filtrování tras podle jednotlivých aut stiskněte tlačítko `Filter by Car`. Stejným tlačítkem lze filter deaktivovat. Kliknutím na náhled trasy se otevře stránka s detailním zobrazením trasy.

Detailní zobrazení trasy obsahuje interaktivní mapu s trasou (viz obrázek B.5), graf nadmořské výšky, graf rychlosti a podrobnosti o trase. Ve spodní části obrazovky se nachází tři tlačítka. Tlačítko `Download GPX` stáhne soubor s trasou ve formátu GPX, který lze nahrát do navigace nebo do mapových portálů třetích stran. Tlačítkem `Update note` otevřeme formulář, ve kterém můžeme doplnit vlastní poznámku o trase. Posledním tlačítkem `Remove`



Obrázek B.5: Ukázka zobrazení trasy na interaktivní mapě.

`route` odstraníme trasu. Před odstraněním budete vyzváni k potvrzení odstranění trasy.

C Instalační manuál systému

C.1 Instalace Android aplikace

Aplikace není dostupná přes `Google Play`[1]. Aby bylo možné aplikaci nainstalovat, je nutné v mobilním telefonu povolit instalaci z neznámých zdrojů. Povolení neznámých zdrojů se nachází v `Nastavení => Zabezpečení => Správa zařízení => Neznámé zdroje`. Je možné, že v některých verzích Androidu se umístění tohoto nastavení mírně liší.

Instalační soubor `car_security.apk` je nutné nahrát libovolným způsobem do paměti mobilního telefonu. Pomocí průzkumníka souborů v mobilním telefonu otevřeme nahraný soubor a pokračujeme dle pokynů na obrazovce. Po instalaci je aplikace možné okamžitě používat.

Doporučuji z bezpečnostních důvodů po instalaci opět zakázat instalaci z neznámých zdrojů. Na již nainstalované aplikace nemá změna nastavení žádný vliv.

C.1.1 Konfigurace aplikace

V případě konfigurace aplikace je následně nutný její překlad.

Změna portů serverových aplikací

Porty lze změnit v souboru config.properties, který se nachází ve složce `app/src/main/Assets`.

communication.network.server.port - Obsahuje port datového serveru.

communication.network.auth.server.port - Obsahuje port autorizačního serveru.

Adresa serveru lze měnit za běhu v nastavení aplikace.

C.1.2 Překlad aplikace

Pro překlad je potřeba mít nainstalované:

- Java 1.8 nebo vyšší

[1] Google Play je oficiální obchod s aplikacemi pro platformu Android.

- Android SDK

Spustitelný `apk` soubor je generován pomocí `Gradle`. V kořenovém adresáři se nachází soubor `gradlew` ve verzi pro `Windows` i pro `Linux`. Pro vytvoření `apk` napíšeme pouze příkaz `gradlew assembleDebug`. Vygenerovaný `apk` soubor se nachází ve složce `app/build/outputs/apk/debug`.

C.2 Instalace serverové části systému

Konfigurace se provádí v souborech `application.properties`, které se nachází v každé serverové aplikaci v adresáři `resources`. V případě potřeby je možné upravovat i soubor `docker-compose.yml`.

V následujících podkapitolách bude stručně popsána konfigurace serverových aplikací a nakonec i jejich spuštění.

C.2.1 Požadavky systému

Pro instalaci a spuštění serverové části systému je potřeba mít nainstalované další aplikace třetích stran, které jsou uvedeny v následujícím seznamu.

Docker - Docker je použit pro překlad a spuštění jednotlivých aplikací v samostatném prostředí.

Docker Compose - Využíván pro spuštění databáze a všech aplikací.

C.2.2 Konfigurace URL adres

Datový server potřebuje znát URL adresu autorizačního serveru. Adresa se nastavuje v souboru `application.properties` pomocí dvou parametrů a musí vždy začínat HTTP nebo HTTPS.

oauth2.check.token.url - URL adresa koncového bodu autorizačního serveru pro ověření platnosti přístupového tokenu.

auth.server.url - URL adresa autorizačního serveru.

Webová aplikace potřebuje přístup k autorizačnímu i datovému serveru. Nastavení je opět v souboru `application.properties`. Adresa musí opět začínat HTTP nebo HTTPS.

auth.server.url - URL adresa autorizačního serveru.

rest.server.url - URL adresa datového serveru.

C.2.3 Konfigurace HTTPS

Všechny tři serverové aplikace využívají pro komunikaci HTTPS protokol, který je konfigurován v souboru `application.properties`. Pro použití je nutno vygenerovat vlastní ověřený certifikát. Z certifikátu vytvořit `keystore` a ten umístit do složky `resources` ve všech aplikacích. Podle vytvořeného certifikátu je nutné nastavit parametry v `application.properties`, které jsou uvozené příponou `server.ssl`. Aplikaci je možné použít bez HTTPS protokolu, ale je nutné tomu přizpůsobit konfiguraci URL adres.

C.2.4 Konfigurace MySQL databáze

Pro použití připravené databáze v podobě `Docker image` je potřeba zvolit cestu k adresáři, ve kterém se budou uchovávat data z databáze. Díky tomu se data neodstraní ani během kompletní obnovy `Docker image`. Tato konfigurace se nachází v souboru `docker-compose.yml`. Upravovaným parametrem je `services => mysql => volumes`. Upravíme pouze řádek, který končí na `:/var/lib/mysql`. Přesná podoba řádku bude:

– absolute_path_to_emtpy_dir:/var/lib/mysql

C.2.5 Užití vlastní MySQL databáze

Pro využití vlastní MySQL databáze je potřeba změnit v autorizačním serveru a datovém serveru adresu databáze a přihlašovací údaje. Parametry se upravují v souboru `application.properties`. Upravované parametry začínají předponou `spring.datasource`. V existující databázi musí existovat schémata definovaná v souboru `docker/init/01_database.sql`. Nakonec je nutné upravit soubor `docker-compose.yml`.

1. Odstraníme konfiguraci MySQL. To znamená veškeré řádky patřící pod `services => mysql`.

2. Změníme adresu MySQL databáze u všech proměnných `WAIT_HOSTS`.

C.2.6 Konfigurace Firebase

Jelikož systém komunikuje se službou FCM, je nutné jej před prvním spuštěním serveru konfigurovat. Kromě konfigurace serveru je nutné přidat konfigurační soubor i do Android aplikace.

1. Vytvoříme účet ve Firebase konzoli:
   `https://console.firebase.google.com`.

2. Pod vytvořeným účtem vytvoříme nový projekt.
3. Ve vytvořeném projektu je nutné přejít k `Settings => Service accounts => Firebase Admin SDK`.
   (a) Stáhneme konfigurační soubor, který získáme tlačítkem `Generate new private key`.
   (b) Z ukázkového kódu zkopírujeme URL adresu databáze.
4. Stažený konfigurační soubor umístíme do složky: `data-server/src/main/resources/`
5. V konfiguračním souboru datového serveru `application.properties` upravíme dva parametry:

   firebase.configuration.file - Název staženého souboru.

   firebase.database.url - Zkopírovaná URL adresa Firebase databáze.
6. Ve Firebase konzoli přejdeme opět do nastavení a přidáme aplikaci s `Package name`: `com.example.tomas.carsecurity`.
7. Stáhneme konfigurační soubor tlačítkem `google-services.json`.
8. Soubor umístíme v Android projektu do složky `app/` a aplikaci přeložíme.

C.2.7 Konfigurace e-mailového klient

Datový server umožňuje odesílat e-mailová upozornění. Je tedy potřeba buď mít vlastní e-mailový server nebo využít existující. Kompletní konfigurace se provádí pomocí souboru `application.properties` datového serveru. V této části se budeme zabývat pouze propojením s `Gmail` účtem, ale úpravou dalších parametrů konfiguračního souboru lze připojit i vlastní e-mailový server.

Veškeré konfigurační parametry jsou uvozeny pomocí `spring.mail`. V případě použití `Gmailu` nás zajímají pouze dva parametry:

spring.mail.username - Uživatelské jméno `Gmail` účtu.

spring.mail.password - Přihlašovací heslo k `Gmail` účtu.

C.2.8 Konfigurace Bing map

K `Bing Maps API` přistupuje jak datový server, tak webová aplikace. U obou je potřeba upravit parametr `bing.map.key` v konfiguračním souboru `application.properties`.

1. Přihlásíme se na portále `https://www.bingmapsportal.com` pomocí `Microsoft` účtu
2. Přejdeme do `My account => My Keys` a vytvoříme nový klíč. Do pole `Application type` vložíme `Website`.
3. Vytvořený klíč a vložíme do obou `application.properties` souborů do proměnné `bing.map.key`.
4. V datovém serveru je navíc nutné určit adresář pro ukládání statických náhledů tras. Do parametru `static.maps.upload.folder` přiřadíme absolutní cestu k existující složce.

C.2.9 Změna použitých portů

Aplikační porty není potřeba upravovat. V případě potřeby porty změnit je nutné upravit porty ve všech souborech `application.properties`, v souboru `docker-compose.yml`, ale i v konfiguraci Android aplikace.

C.2.10 Spuštění serverových aplikací

Pro spuštění se zadá v adresáři, kde se nachází soubor `docker-compose.yml`, příkaz `docker-compose up`. Na `Linuxu` může být potřeba přidat `sudo`. Pro spuštění na pozadí přidáme nakonec parametr `-d`.

D Uživatelské testovací scénáře

V této příloze jsou popsány testovací scénáře celého systému, kterými lze ověřit, zda systém obsahuje požadovanou funkcionalitu a zároveň se ověří základní funkčnost celého systému.

D.1 Prerekvizity

Během testů se předpokládá:

- Webová aplikace je dostupná na adrese: `https://carsecurity.mooo.com`
- Android aplikace je úspěšně nainstalována v mobilním telefonu s operačním systémem Android 4.0 (Ice Cream Sandwich) nebo vyšším.
- V systému jsou registrováni dva uživatelé `user1` a `user2` jejichž heslo je `12345678`.
- `user1` vlastní minimálně jedno auto a má nahráno několik cest a vytvořeno několik událostí.
- `user2` je prázdný uživatel bez aut a bez záznamů.

D.2 Vizuální kontrola Android aplikace

Předpokládá se, že aplikace je čerstvě nainstalována, veškeré služby jsou vypnuté a uživatel není přihlášen.

Výchozí obrazovka (Home)

Při spuštění aplikace se otevře obrazovka obsahující:

- Dvě modro zelená tlačítka s popisky `ALARM` a `TRACKER` v horní části obrazovky.
- V pravé spodní části obrazovky je tlačítko s ikonou nastavení, které po stisku otevře obrazovku s nastavením.

- Titulek stránky je `Home`.
- Vedle titulku stránky se nachází tlačítko se třemi vodorovnými čarami (Hamburger menu), které po stisku otevře postranní panel na levé straně obrazovky.

Postranní panel

Postranní menu obsahuje od shora dolů:

- Barevný panel s Android ikonou a textem `Car Security`.
- Menu položku `Home` s ikonou domu.
- Menu položku `Login` s ikonou obsahující obdélník se šipkou směřující do obdélníku.
- Menu položku `Settings` s ikonou nastavení.
- Menu položku `Status` s ikonou obsahující písmeno 'i' uvnitř kruhu.

Po stisku jednotlivých menu položek se otevře adekvátní obrazovka a postranní panel se uzavře.

Login obrazovka

Obrazovka obsahuje:

- Titulek s názvem `Login`.
- Vedle titulku stránky se nachází tlačítko se třemi vodorovnými čarami (Hamburger menu), které po stisku otevře postranní panel na levé straně obrazovky.
- V pravé spodní části obrazovky je tlačítko s ikonou nastavení, které po stisku otevře obrazovku s nastavením.
- Je-li uživatel odhlášen:
  - Dvě vstupní pole nadepsané `User name` a `Password`.
  - Vlastní text v poli `Password` není čitelný.
  - Pod textovými poli se nachází tlačítko s popiskem `Login`.
- Je-li uživatel přihlášen:

  - V horní části obrazovky se nachází text `You are logged in as a xy`. Míto `xy` je jméno přihlášeného uživatele.
  - Pod textem se nachází tlačítko s popiskem `Logout`.
  - V horní části postranního panelu přibyl text obsahující jméno přihlášeného uživatele a název vybraného vozu.

Obrazovka nastavení

- Neobsahuje tlačítko pro otevření postranního panelu.
- Obsahuje pět tlačítek, které otevírají vlastní podmenu:
  - Alarm
  - Tracker
  - Sensors
  - Communication
  - Power-save mode
- Každé tlačítko obsahuje svoji unikátní ikonu.
- Položky v podmenu jdou upravovat a po vrácení zpět na stánku zůstanou na nastavené hodnotě.

Obrazovka status

- Titulek s názvem `Status`.
- Vedle titulku stránky se nachází tlačítko se třemi vodorovnými čarami (Hamburger menu), které po stisku otevře postranní panel na levé straně obrazovky.
- V pravé spodní části obrazovky je tlačítko s ikonou nastavení, které po stisku otevře obrazovku s nastavením.
- V horní části obrazovky se nachází nadpis Status: ...
- Pod nadpisem je text `Local routes`.
- Nakonec stránka obsahuje seznam tras uložených v paměti telefonu (nesynchronizované trasy).
- Trasa obsahuje datum vytvoření a počet pozic.

- Pokud nejsou vytvořeny žádné trasy, je zobrazen text `No local route`.
- Seznam lze aktualizovat potažením směrem dolů.

D.3 Přihlášení a odhlášení v Android aplikaci

Pro tento test musí být telefon připojen k internetu prostřednictvím povoleného zdroje. Na obrazovce `login` se nachází přihlašovací formulář. Pokud není uvedeno jinak veškeré testy vycházejí z odhlášeného stavu.

Vytvoření nového auta

Po zadání uživatelského jména `user1` a hesla `12345678` se zobrazí seznam aut, které vlastní `user1`. Pod seznamem aut se nachází tlačítko `Create new car`. Seznam lze opustit, ale uživatel zůstane odhlášen. Po stisku tlačítka `Create new car` se zobrazí vstupní pole pro zadání nového názvu auta. Okno lze opět opustit, ale uživatel zůstane odhlášen. Po zadání a potvrzení nového názvu auta se objeví obrazovka pro odhlášení. V postranním panelu je správné uživatelské jméno a správný název auta. Přihlášený uživatel může vidět ve webové aplikaci nově vytvořené auto.

Výběr existujícího auta

Po zadání uživatelského jména `user1` a hesla `12345678` se zobrazí seznam aut, které vlastní `user1`. Seznam lze opustit, v takovém případě zůstane uživatel odhlášen. Vybereme jedno z existujících aut. Objeví se obrazovka pro odhlášení. V postranním panelu je správné uživatelské jméno a správný název auta.

Odhlášení

Přihlášený uživatel stiskne tlačítko `logout`. Objeví se přihlašovací obrazovka. V postranním panelu již není viditelné jméno přihlášeného uživatele ani název vybraného automobilu.

Prázdné přihlašovací údaje

Pole pro uživatelské jméno a heslo jsou prázdná nebo obsahují pouze bílé znaky. Po stisku tlačítka `login` se zobrazí chybová zpráva `Username and`

`password can not be empty`.

Nesprávné přihlašovací údaje

Po zadání neexistujícího uživatelského jména a hesla a stisku tlačítka `login`, se objeví chybová hláška `Invalid credentials`. Stejná chybová hláška se zobrazí i po zadání správného uživatelského jména, ale chybného hesla.

D.4 Alarm

Ovládání alarmu se nachází na hlavní obrazovce.

Spuštění alarmu

Základní spuštění

Nastavení před spuštěním alarmu bude:

- Sensors => Allow Move sensor => ON
- Communication => Allow GSM communication => OFF
- Alarm
  - Allow Alarm => ON
  - Start interval => větší než 0
  - Alert interval => větší než 0
  - Make call => OFF
  - Siren => ON

Alarm spustíme z hlavní obrazovky modrozeleným tlačítkem `Alarm`. Po spuštění tlačítko zezelená a v oznamovací liště se objeví nová notifikace, kterou nelze odstranit. Nad tlačítkem se objevil indikátor průběhu (progressbar), který načítá po dobu nastavenou v `Settings => Alarm => Start Interval`, po kterou nelze spustit poplach. Po uplynulé době progressbar zmizí. Zatřeseme telefonem a počkáme dobu, která je nastavena v `Settings => Alarm => Alert interval`. Po uplynulé době se spustí siréna (poplach) a tlačítko zčervená. Stisknutím tlačítka `Alarm` se poplach ukončí.

Změna intervalů

Celý proces zopakujeme s tím rozdílem, že libovolně změníme hodnoty v `Setting => Alarm => Start interval` a `Settings => Alarm => Alert interval`. Doby musí opět odpovídat těm nastaveným.

Zvukový senzor

V dalším testu použijeme místo pohybového senzoru zvukový senzor. Zvukový senzor aktivujeme pomocí `Settings => Sensors => Sound sensor => Allow Sound sensor => ON`. Spustíme alarm stejným způsobem jako v předchozích případech. Pro aktivaci poplachu, ale použijeme zvuk. V případě, že nelze aktivovat poplach, zkusíme změnit citlivost zvukového senzoru pomocí `Setting => Sensors => Sound sensor => Sensor Sensitivity` a test zopakujeme.

Upozornění na poplach

Pro následující testovací scénáře je možné libovolně aktivovat poplach. Scénáře se zabývají pouze upozorněním na poplach.

Hovor

V mobilním telefonu musí být umístěna platná SIM karta. V nastavení povolíme GSM komunikaci a zadáme kontaktní telefonní číslo:

- Settings => Communication => GSM => Allow GSM communication => ON
- Settings => Communication => GSM => Phone number => kontaktní telefonní číslo
- Settings => Alarm => Make call => ON

Po spuštění poplachu se otevře aplikace pro hovory a vytočí se kontaktní telefonní číslo.

SMS

V mobilním telefonu musí být umístěna platná SIM karta. V nastavení povolíme GSM komunikaci a zadáme kontaktní telefonní číslo. Vypneme funkci volání pokud je povolena:

- Settings => Communication => GSM => Allow GSM communication => ON

- Settings => Communication => GSM => Phone number => kontaktní telefonní číslo
- Settings => Alarm => Make call => OFF

Po spuštění poplachu je odeslána SMS zpráva na kontaktní telefonní číslo informující o poplachu.

Průběžné SMS s polohou

V mobilním telefonu musí být umístěna platná SIM karta. V nastavení povolíme SMS komunikaci a zadáme kontaktní telefonní číslo.

- Settings => Communication => GSM => Allow GSM communication => ON
- Settings => Communication => GSM => Phone number => kontaktní telfonní číslo
- Settings => Communication => GSM => SMS types => Send alarm location => ON
- Settings => Sensors => Location sensor => Allow Location sensor => ON
- Settings => Alarm => Message location interval => 60s

Po spuštění poplachu se bude odesílat SMS zpráva s aktuální polohou zařízení každou minutu. SMS zpráva je odeslána pouze pokud zařízení dokáže zjistit svoji polohu.

Průběžné zprávy s polohou

Mobilní telefon musí být připojen k internetu a uživatel musí být přihlášen. Pro povolení zpráv musí být nastaveno

- Settings => Communication => Network => Allow network communication => ON
- Settings => Communication => Network => Additional communication => Alarm + Alarm position => ON
- Settings => Sensors => Location sensor => Allow Location sensor => ON
- Settings => Alarm => Message location interval => 60s

Po spuštění poplachu se bude odesílat zpráva s aktuální polohou zařízení každou minutu. Zpráva je odeslána pouze pokud zařízení dokáže zjistit svoji polohu. Zprávy jsou viditelné ve webové aplikaci pod záložkou `Events`. Událost obsahuje odkaz na mapy se zvýrazněnou polohou zařízení.

Událost na serveru a e-mailové upozornění

V tomto testu musí být zařízení připojené k internetu a v zařízení musí být uživatel přihlášen. Dále je nutné aktivovat upozornění pro server:

- Settings => Communication => Network => Additional communication => Alarm

Pro e-mailové upozornění je nutné zadat svoji e-mailovou adresu v nastavení webové aplikace.

Po spuštění poplachu musí být po chvíli dostupná událost v záložce `Events`. Událost se neobjeví automaticky, stránku je nutné znovu načíst. Na zadanou e-mailovou adresu přišla zpráva informující o poplachu.

Siréna

Siréna byla používaná ze začátku testovacích scénářů, proto zde nebude zkoušeno, že siréna lze zapnout, ale že siréna lze také vypnout. V nastavení `Alarm => Siren => OFF`. Po spuštění poplachu tlačítko alarmu zčervená, ale siréna nehraje.

D.5 Tracker

Ovládání knihy jízd se nachází stejně jako ovládání alarmu na hlavní obrazovce. Pro použití knihy jízd musí být provedena následující nastavení:

- Settings => Sensors => Location sensor => Allow Location sensor => ON
- Settings => Tracker => Allow Tracker => ON

Zároveň musí být uživatel v Android aplikaci přihlášen. Během měření není potřeba připojení k internetu.

Spuštění knihy jízd

Kniha jízd se spustí stisknutím modro zeleného tlačítka na hlavní obrazovce s popiskem `Tracker`. Popisek se změní na naměřenou ujetou vzdálenost během

aktuální cesty a barva tlačítka se změní na zelenou. Do 500 metrů je délka uvedena v metrech. Od 500 metrů je délka uvedena v kilometrech.

Po připojení zařízení k internetu se nahraná trasa uloží na server nejpozději do doby uvedené v `Settings => Communication => Network => Synchronization interval`. Nahraná trasa je viditelná včetně náhledu ve webovém prostředí pod záložkou `Routes`. Po kliknutí na trasu se zobrazí interaktivní mapa s naměřenou trasu. Trasa je na mapě zvýrazněna a odpovídá skutečné ujeté trase. Pod mapou se nachází grafy obsahující nadmořskou výšku na trase a rychlost. Pod grafy se nachází dodatečné informace. Veškeré informace odpovídají skutečné trase s přihlédnutím na přesnost GPS senzoru v mobilním telefonu.

Export trasy

Ve webové aplikaci v zobrazení trasy lze stáhnout trasu tlačítkem `download GPX`. Stažený soubor je validní GPX formát a uvedená trasa odpovídá trase zobrazené na interaktivní mapě.

Automatické vypnutí knihy jízd

Je-li spuštěno nahrávání knihy jízd, je automaticky vypnuto po dobu bez pohybu nastavené v `Settings => Tracker => Timeout`. Důležité je během testu nastavit `Settings => Tracker => Not moving distance` na hodnotu větší než nula. Senzor v mobilním telefonu není přesný a při nastavení příliš malé hodnoty se zařízení nepřestane pohybovat.

D.6 Power-save mode

Power save mód musí být aktivován v `Settings => Power-save mode => Allow power save mode => ON`. Nastavení hladiny, ve které je mód automaticky aktivován, se provede v `Power-save mode => Critical level`.

Při poklesu hladiny baterie pod nastavenou hladinu se automaticky přepne do jedné minuty aplikace do power save módu. Při vzestupu hladiny baterie nad nastavenou hladinu se power save mód opět do jedné minuty deaktivuje.

Pro ověření správného fungování je potřeba provést následující testy během aktivovaného power save módu:

- Zapnout sirénu v nastavení aplikace a spustit poplach. Siréna nehraje.

- Povolit pohybový senzor a zakázat zvukový senzor. Při aktivaci alarmu se vypíše hláška, že nelze zapnout alarm.
- Zapnout v nastavení odesílání zpráv s polohou zařízení. Po spuštění poplachu jsou zprávy odesílány v 10 minutových intervalech.
- Zaznamenávání knihy jízd nelze zapnout a zobrazí se informativní zpráva.
- Vytvořené cesty, které jsou pouze v zařízení, se na server nenahrají. Události však ano. Po vypnutí power save módu se vše nahraje na server.

D.7 Ovládání pomocí SMS

Pro ovládání zařízení pomocí SMS zpráv musí být v telefonu platná SIM karta a musí být uplatněna tato nastavení:

- Settings => Communication => GSM => Allow GSM communication => ON
- Settings => Communication => GSM => Phone number => validní telefonní číslo ze kterého budou chodit příkazy
- Settings => Communication => GSM => Sms types =>
  - Comand tool switch => ON
  - Command send status => ON
  - Command send location => ON

Aplikace reaguje na SMS zprávy pouze od telefonního čísla, které je uvedeno v nastavení. Po odeslání zprávy z jiného telefonního číslo aplikace nijak nereaguje.

Aplikace provede akci pouze při příjmu následujících příkazů:

activate Alarm - po přijetí zprávy se aktivuje alarm. Pokud je již zapnutý, nic se nezmění.

deactivate Alarm - po přijetí zprávy se deaktivuje alarm. Pokud je již vypnutý, nic se nezmění.

activate Tracker - po přijetí zprávy se aktivuje vytváření knihy jízd. Pokud je již zapnuté, nic se nezmění.

deactivate Tracker - po přijetí zprávy se deaktivuje vytváření knihy jízd. Pokud je již vypnuté, nic se nezmění.

info - zařízení odešle zpět zprávu o aktuálním stavu systému.

D.8 Ovládání přes webovou aplikaci

Pro ovládání přes webovou aplikaci musí být uživatel přihlášen, telefon musí být připojen k internetu a musí být uplatněna tato nastavení:

- Settings => Communication => Network => Allow network communication => ON

Ve webové aplikace se odesílají příkazy z hlavní obrazovky. Každé registrované vozidlo obsahuje vlastní panel pro odesílání příkazu.

- Tlačítkem `Reload` odešle příkaz na získání aktuálního stavu zařízení, které se zobrazí pod tlačítkem.
- Tlačítkem `Turn on Alarm` se spustí v mobilním zařízení alarm a tlačítko zezelená.
- Tlačítkem `Turn off Alarm` se vypne v mobilním zařízení alarm a tlačítko zmodrá.
- Tlačítkem `Turn on Tracker` se spustí v mobilním zařízení vytváření knihy jízd a tlačítko zelená.
- Tlačítkem `Turn off Tracker` se vypne v mobilním zařízení vytváření knihy jízd a tlačítko zmodrá.

Může se stát, že zařízení neodpoví na první pokus o odeslání příkazu. Úsporné režim Android zařízení totiž mohou zabraňovat okamžité komunikaci se zařízením.

D.9 Synchronizace pouze přes Wi-Fi

Nastavení komunikace pouze prostřednictvím mobilních dat je v `Settings => Communication => Network => Mobile data Synchronization`. Je-li nastavení vypnuto, aplikace nekomunikuje se serverem pokud není dostupné Wi-Fi připojení. Trasy ani události nejsou synchronizovány a příkazy přes webovou aplikaci nereagují.

D.10 Vizuální kontrola webové aplikace

Webová aplikace je dostupná a čitelná jak v počítači na velké obrazovce, tak na mobilním telefonu přes webový prohlížeč. Pokud na jakékoliv stránce nejsou žádná data k zobrazení, je zobrazena alespoň informativní zpráva.

V horní části obrazovky se nachází lišta s názvem aplikace `Car security` a pokud je uživatel přihlášen, obsahuje i navigační menu s položkami `Events`, `Cars`, `Routes` a `Settings`. Na levé straně se nachází uživatelské jméno a tlačítko pro odhlášení.

Obrazovka přihlášení (/login)

Obrazovka obsahuje dvě vstupní pole. Jedno nadepsané `Username` a druhé `Password`. Pod těmito vstupními poli se nachází přihlašovací tlačítko `Login`. Pod přihlašovacím tlačítkem je drobný nápis s odkazem pro zobrazení stránky pro registraci.

Obrazovka registrace (/register)

Obrazovka obsahuje tři vstupní pole. Jedno nadepsané `Username`, druhé `Password` a třetí `Password confirmation`. Na stránce se nachází informační text o minimální délce hesla. Pod vstupními texty je tlačítko s popiskem `Register`. Pod tlačítkem je drobný nápis s odkazem pro zobrazení přihlašovací obrazovky.

Úvodní obrazovka (/)

Obrazovka obsahuje seznam všech aut uživatele s tlačítkem `Reload`. Pokud je zařízení dostupné, zobrazí se ke každému zařízení informace o zapnutých nástrojích, stavu baterie, stavu připojení k elektrické energii a stav úsporného režimu.

Obrazovka událostí (/event)

Obrazovka událostí obsahuje veškeré události vytvoření všemi automobily seřazené od nejnovějšího po nejstarší. Události lze filtrovat podle automobilu. U události lze změnit poznámku a událost lze zcela odstranit. Každá poznámka obsahuje jméno auta, datum, čas, poznámku a název.

Obrazovka aut (/car)

Obrazovka obsahuje seznam všech automobilů uživatele. Názvy automobilů se mohou shodovat. Na obrazovce lze vytvořit nový automobil. U každého automobilu je zobrazen počet jízd a událostí, poznámka a jméno auta. Jméno auta a poznámka lze změnit. Automobil lze odstranit.

Obrazovka cest (/route)

Obrazovka obsahuje veškeré cesty vytvořené všemi automobily. Cesty lze filtrovat podle automobilu. Každá cesta obsahuje náhled trasy, název auta, datum, čas, délku, průměrnou rychlost, čas jízdy a poznámku. Na každou cestu lze kliknout a otevřít detail cesty.

Obrazovka cesty (/route/XY)

Obrazovka cesty obsahuje interaktivní mapu s vyobrazenou celou cestou, graf nadmořské výšky v průběhu cesty, graf rychlosti v průběhu cesty, název auta, datum, čas, délku cesty, průměrnou rychlost, čas jízdy a poznámku. Cestu lze odstranit nebo stáhnout ve formátu GPX. Poznámku o trase lze upravit.

Lze přistoupit pouze k cestám, které byly vytvořeny autem aktuálně přihlášeného uživatele.

Obrazovka nastavení (/settings)

Na obrazovce s nastavením je zobrazeno uživatelské jméno, které nelze změnit, textové pole pro zadání e-mailové adresy, tlačítko pro uložení nově zadané e-mailové adresy, tlačítko pro otevření okna pro změnu uživatelského hesla a tlačítko pro odstranění uživatele.

D.11 Přihlášení do webové aplikace

Na přihlašovací obrazovce do webové aplikace (`/login`) vyzkoušíme zadat několik kombinací vstupů.

Špatné uživatelské jméno - Do pole `Username` zadáme neexistující uživatelské jméno. Do pole `Password` vložíme libovolné heslo. Po stisku tlačítka `Login` se zobrazí chybová hláška s textem: `Invalid username and password.`

Špatné heslo - Do pole `Username` zadáme existující uživatelské jméno například `user1`. Do pole `Password` vyplníme libovolné, ale nesprávné heslo. Po stisku tlačítka `Login` se zobrazí chybová hláška `Invalid username and password.`

Prázdné uživatelské jméno a heslo - Vstupní pole necháme prázdná a stiskneme tlačítko `Login`. Zobrazí se upozornění na vyplnění obou prázdných polí.

Úspěšné přihlášení - Do pole `Username` zadáme uživatelské jméno `user1`. Do pole `Password` vyplníme heslo `12345678`. Po stisku tlačítka `Login` je uživatel úspěšně přihlášen do aplikace.

D.12 Registrace přes webovou aplikaci

Na obrazovce pro registraci (`/register`) do webové aplikace vyzkoušíme zadat několik kombinací vstupů:

Prázdná pole - Všechna nebo některá pole necháme nevyplněna. Po stisku tlačítka `Register` se zobrazí upozornění na vyplnění prázdných polí.

Bílé znaky - Vyplníme všechny vstupy libovolnými bílými znaky. Po stisku tlačítka `Register` se zobrazí upozornění: `Please fill all inputs!`.

Existující uživatel - Zadáme uživatelské jméno `user1` a jako heslo zvolíme `87654321`. Po stisku tlačítka `Register` se zobrazí upozornění: `Invalid input data or user already exists.`

Krátké heslo - Vyplníme libovolné uživatelské jméno. Obě hesla zvolíme `123`. Po stisku tlačítka `Register` se zobrazí upozornění: `Password is too short. Minimum is 8 characters.`

Neshodné heslo - Zvolíme libovolné uživatelské jméno a libovolná hesla, která se ale liší. Po stisku tlačítka `Register` se zobrazí upozornění: `Passwords do not match!`.

Úspěšná registrace - Vyplníme neexistující uživatelské jméno a zvolíme libovolné heslo o minimální délce osmi znaků, které neobsahuje bílé znaky. Po stisku tlačítka `Register` se zobrazí přihlašovací stránka s upozorněním: `You have been successfully registered. You can login now.`

D.13 E-mailové notifikace

Pokud je uživatel přihlášen v Android aplikaci, ve které je povolena síťová komunikace, jsou povolena upozornění v `Settings => Communication => Network => Additional communication`, mobilní zařízení je připojeno k internetu a ve webové aplikaci je nastavena e-mailová adresa, dostává následující upozornění do e-mailové schránky:

- Spuštění poplachu,
- Průběžné zprávy o poloze při poplachu,
- Připojení k externímu zdroji energie,
- Odpojení od externího zdroje energie,
- Zapnutí power save módu.