TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Mobilní taktický asistent pro výcvik Aktivních záloh

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Ondřej Dlabola**
Vedoucí práce: Ing. Igor Kopetschke

Liberec 2018

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Mobile tactical assistant app for Army reserves training

Master thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information Technology

Author: **Bc. Ondřej Dlabola**
Supervisor: Ing. Igor Kopetschke

Liberec 2018

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Dlabola**
Osobní číslo:	**M15000161**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Mobilní taktický asistent pro výcvik Aktivních záloh**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Uveďte současný stav vytváření a předávání situačních hlášení při výcviku Aktivních záloh.
2. Pokud to bude možné v závislosti na utajované skutečnosti, zpracujte v rešerši aktuální přehled používaných technologií Armády České republiky.
3. Navrhněte zefektivnění této činnosti za využití moderních a dostupných mobilních technologií s ohledem na bezpečnost
4. Implementujte svůj návrh v podobě aplikace pro platformu Android za využití jazyka Kotlin,
5. Pokuste se získat zpětnou vazbu v praktickém nasazení při výcviku

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **40 - 60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] Statically typed programming language for modern multiplatform applications [online]. Prague, Czech Republic: JetBrains, 2017 [cit. 2017-09-21]. Dostupné z: https://kotlinlang.org/
[2] Kotlin and Android [online]. California, USA: Google, 2017 [cit. 2017-09-21]. Dostupné z: https://developer.android.com/kotlin/index.html
[3] Příručka vojáka AČR. 3. vydání. Velitelství výcviku Vojenská akademie Vyškov, 2009.

Vedoucí diplomové práce: **Ing. Igor Kopetschke**
Ústav nových technologií a aplikované informatiky

Datum zadání diplomové práce: **19. října 2017**

Termín odevzdání diplomové práce: **14. května 2018**

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2017

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 10.5. 2018

Podpis:

Abstrakt

Tato práce se zabývá výcvikem vojenských jednotek v oblasti předávání zpráv a hlášení. Jde především o zjednodušení a zefektivnění přípravy standardizovaných hlášení používaných v ozbrojených složkách. Konkrétně se práce zaměřuje na oblast záložních jednotek, které mají omezený čas a prostor pro výcvik.

Cílem práce je zefektivnit a zrychlit přípravu hlášení při výcviku a tím umožnit procvičování dalších tématických oblastí. Zpráva seznamuje čtenáře se základy z vojenského prostředí, nezbytnými pro pochopení problematiky, popisuje návrh a implementaci vhodného řešení.

Jako řešení je navržena aplikace pro operační systém Android. Aplikace je implementována pomocí programovacího jazyka Kotlin a využívá funkce chytrých zařízení k zjednodušení vytváření hlášení. V závěru jsou popsány zkušenosti uživatelů z testování při výcviku Aktivních záloh ozbrojených sil České republiky.

Vytvořená aplikace usnadňuje výcvik tím, že uživatele zbavuje nutnosti ručně hledat a získávat informace, které dokáže chytré zařízení získat automaticky. Tím mu dává prostor soustředit se na další úkoly a zrychlit vytvoření a odeslání hlášení, čímž také získává více času pro procvičení dalších dovedností

Klíčová slova: Android, Kotlin, Aktivní záloha, MGRS, standardizovaná hlášení, Anko, KOIN

Abstract

This thesis topic is the training of military units in the communication and signals. It is focused on simplifying of preparation of the standardized reports used in armed forces. The focus is further narrowed on the reserve units and their training, which is greatly limited by time.

The aim of the thesis is to make the preparation of report more efficient and simpler and let more time to be spent on another exercised subjects. This paper makes it's reader acquainted with the necessary basic knowledge from the military training and describes proposed solution and suitable implementation.

The proposed solution is to implement application for Android operation system. This application is written in Kotlin programming language and utilizes smart device features to simplify preparation of reports. The user experiences from the military exercise of Czech republic Army reserves are described in the conclusion of the paper.

Implemented application releases user from necessity to obtain and prepare the data for report, that can be easily automatically prepared by the smart device, and simplifies his training duties. Hence the user can prepare the report quicker and concentrate more on another tasks of his training.

Keywords: Android, Kotlin, Active Reserve, MGRS, standardized reports, Anko, KOIN

Obsah

Seznam obrázků

Seznam zdrojových kódů

Seznam zkratek

AČR	Armáda České republiky
AZ	Aktivní záloha
DTG	date time group
IED	improvised explosive device
MGRS	military grid reference system
OPZHN	ochrana proti zbraním hromadného ničení
OSČR	Ozbrojené síly České republiky
UXO	unexploded ordnance

Úvod

Každý občan České republiky může dobrovolně zažádat o zařazení do Aktivních záloh a převzít tak na sebe výkon branné povinnosti. Pokud splňuje požadovaná kritéria a existuje pro něj místo, je přijat a zařazen ke vhodné jednotce. Jeho úkolem je poté připravovat se k obraně vlasti a v případě potřeby ji bránit. Tuto povinnost jsem na sebe dobrovolně převzal i já. Zkušenosti, které jsem při výcviku získal, jsem se rozhodl propojit se znalostmi a dovednostmi nabytými při studiu informatiky a využít je ke zpracování práce, jejíž zprávu nyní čtete.

Při službě v Aktivních zálohách jsem byl cvičen mimo jiné i v komunikaci pomocí radiostanic a předávání hlášení o stavu své jednotky. V rámci výcviku jsem si uvědomil, že by šlo efektivně nahradit dosud používané postupy v sestavování hlášení a zjednodušit jejich přípravu. Tím bych mohl sobě i svým kolegům výcvik usnadnit a zbavit se monotónní rutinní části přípravy, kterou lze výhodně nahradit prací stroje. V mém případě by tímto strojem mohlo být chytré zařízení - telefon, tablet nebo chytré hodinky, které by nahradilo tužku s papírem a mapu.

Tuto myšlenku jsem se rozhodl realizovat pomocí aplikace pro chytrý mobilní telefon v druhé polovině roku 2017. V tomto roce společnost Google oznámila, že bude podporovat programovací jazyk Kotlin, jako oficiální pro vytváření aplikací na mobilní operační systém Android. Tato zpráva mě zaujala, protože jsem již předtím sledoval dění okolo tohoto jazyka. Kotlin se pasuje do pozice nástupce jazyka Java, se kterým mám jisté zkušenosti, a lákalo mě vyzkoušet si ho. Proto jsem se rozhodl využít jej pro realizaci své myšlenky.

V této písemné zprávě bych rád čtenáře seznámil se svými zkušenostmi z Aktivních záloh a se všemi nezbytnými informacemi nutnými k pochopení hlášení využívaných při výcviku a jejich předávání. Dále bych chtěl věnovat část zprávy popisu požadovaných funkcí vytvořené aplikace a tomu, proč jsem se rozhodl pro operační systém Android. Velkou část budu věnovat také návrhu aplikace, použití jazyka Kotlin a frameworkům využitým při implementaci. V závěrečné části uvedu zkušenosti získané při testování aplikace na vojenském cvičení jednotky Aktivních záloh.

1 Zprávy ve vojenském prostředí

V první části této zprávy bych chtěl čtenáře seznámit s její cílovou oblastí, tedy se službou v Aktivních zálohách a se zkušenostmi, které byly podnětem pro mou práci. Pro přiblížení problematiky bude třeba vysvětlit alespoň hrubé rysy výcviku a komunikace v bezpečnostních složkách.

Pro komunikaci jsou často využívána standardizovaná hlášení, která budou v této kapitole popsaná. S nimi souvisí také způsob jejich předávání a mechanismy umožňující předání zprávy ve formě srozumitelné jejímu adresátu a zároveň ideálně nesrozumitelné další straně, která by mohla zprávu odchytit. I tato témata se pokusím vysvětlit v této kapitole.

V závěru kapitoly bych rád popsal dosud využívané postupy přípravy hlášení a existující řešení, která umožňují jejich zjednodušení.

1.1 Aktivní zálohy

Aktivní zálohy jsou oficiální součástí ozbrojených sil České republiky. Jedná se o složku tvořenou občany ČR, kteří se rozhodli z vlasteneckých či různých jiných důvodů dobrovolně převzít brannou povinnost. Tato složka byla vytvořena ze dvou důvodů. Prvním důvodem je potřeba ozbrojených sil mít vycvičenou a připravenou zálohu pro doplnění stavů vojenských útvarů v případě ohrožení státu nebo ve válečném stavu. Druhým důvodem je právě snaha umožnit občanům a vojákům v záloze účastnit se cvičení a připravovat se k plnění úkolů ozbrojených sil nad rámec jejich běžné povinnosti.

V rámci Aktivní zálohy (dále též pouze AZ) jsou zařazeni na pevně dané místo v jedné z krajských pěších rot nebo v záložních jednotkách, které postupně vznikly u každé jednotky Armády ČR. Krajské pěší roty mají plnit funkci teritoriálních jednotek, určených ke střežení důležitých objektů ve svém kraji a ke spolupráci se státní policií i ostatními složkami integrovaného záchranného systému. Jednotky záloh u stálých vojenských útvarů jsou určeny především k doplnění jejich stavu v případě potřeby a plní stejné úkoly jako jejich mateřské útvary. V době míru, kdy není třeba jednotky AZ nasazovat k plnění jejich primárních úkolů, je jejich hlavní náplní příprava a výcvik pro plnění těchto úkolů. Kromě toho mohou být povolány k řešení krizových situací (např. živelné katastrofy apod.).

1.1.1 Výcvik

Pro lepší pochopení služby a aktivit, které vykonává běžný příslušník aktivních záloh, se nyní pokusím popsat, co vše vykonává v mírovém stavu. Zde budu vycházet ze své vlastní zkušenosti a ze zkušeností mých vlastních kolegů. Jako příslušník krajské jednotky sice nemám zkušenosti od jednotek AZ u bojových jednotek a nedokáži předat jejich zkušenosti. Věřím, že pro pochopení mojí motivace k zpracování tohoto tématu to není nutné. Zkušenosti z fungování aktivních záloh za jiného než mírového stavu zde nebudu popisovat. Jednak je nikdo v naší zemi nemá a navíc to pro mou práci rovněž není důležité.

Standardní mírová aktivita vojáka v AZ se sestává především z částečně pravidelných cvičení a z různých akcí pro veřejnost, propagačních či jiných. Důležitá pro nás v této práci jsou především cvičení. Voják v záloze má ze zákona dané penzum dnů, na které smí být povolán v průběhu jednoho roku ke cvičení. Dosavadní praxe je zatím taková, že většina vojáků nedokáže toto penzum vyčerpat. Jednak proto, že ne vždy a na všechna cvičení musí být povolán každý voják. Také ze strany vojáka mohou nastat důvody, pro které se nemůže zúčastnit vojenských cvičení. Ať už jimi jsou zdravotní důvody, osobní nebo jiné závažné důvody, ze cvičení se lze omluvit.

1.1.2 Charakteristika vojáků AZ

V optimálním případě je voják povolán na dvě velká cvičení v průběhu roku, každé v délce zhruba jednoho týdne. V obzvláště příznivém případě může voják stihnout více cvičení, ale není to pravděpodobné. Pokud bychom uvažovali pravidelné rozložení cvičení v průběhu roku, tak se voják dostane na cvičení každých šest měsíců nebo v předchozích letech jen jednou za rok (to samé, pokud se z nějakých závažných důvodů nezúčastní jednoho cvičení).

Dalším kritériem, které je třeba uvažovat, jsou samotní vojáci. Ke službě v aktivní záloze se nyní může přihlásit každý, kdo splňuje následující kritéria. Předně musí dosáhnout věku 18 let (naopak vrchní hranice je okolo 60 let, zde není úplně striktní), musí být zdravotně způsobilý a mít minimálně středoškolské vzdělání ukončené výučním listem. Dále musí být trestně bezúhonný a pro úplnost - předložit čestné prohlášení, že nepodporuje žádné nenávistné hnutí. Pokud splňuje toto vše a projde základním výcvikem, může být přijat. Je tu samozřejmě také nutná podmínka, že jej ozbrojené síly potřebují.

Reálné složení vojáků v AZ je díky těmto kritériím poměrně pestré. Věkové složení skupiny se pohybuje v celém rozmezí od čerstvě plnoletých vojáků až po vojáky na hranici aktivní služby kolem věku šedesáti let, kdy už je na jednotlivém posouzení, zda bude jejich služební závazek prodloužen nebo ne. Zastoupení mužů a žen je v AZ silně nakloněno ve prospěch mužů, ale objevují se i ženy, toužící se do služby zapojit.

Podobně pestré, jako věkové rozložení vojáků, je i jejich spektrum profesí a úroveň nejvyššího dosaženého vzdělání. Velká většina vojáků má dokončené úplné středoškolské vzdělání s maturitou, ale významný je i počet vojáků s pouhým výučním listem. Oproti tomu je počet vysokoškolsky vzdělaných vojáků nepoměrně nižší.

S tím souvisí i ohromná variabilita v jejich civilních profesích. Tak se v AZ objevují lidé, kteří jsou truhláři, řidiči autobusů, projektanti, konstruktéři, podnikatelé, učitelé, programátoři a mnozí další.

Zájemci o službu v AZ mají také často rozdílnou motivaci. Najdou se zde dobrodružné povahy, které touží zkusit si něco zajímavého, nového. Jsou tu lidé, kteří s nostalgií vzpomínají na základní službu, dále lidé se silným vztahem k armádě, vlastenci s touhou sloužit své zemi i zájemci o vstup do armády, kteří berou AZ pouze jako dočasnou přestupnou zkušenost. Od zavedení nových legislativních změn zákonem č. 45/2016 Sb. S platností od poloviny roku 2016 mohou být motivací pro službu v AZ i finanční podmínky. Další skupinou lidí v AZ jsou vojáci z povolání, kteří po ukončení posledního závazku jsou povinni po nějakou dobu ještě sloužit v AZ.

1.1.3 Specifika výcviku AZ

Předchozí popis fungování a charakteristiky slouží k představení variability vojáků, se kterou je nutné se potýkat při výcviku AZ. Oproti profesionálním vojákům musí AZ bojovat s tím, že její příslušníci jsou nejen vojáky, ale také občany se svými civilními profesemi. Jejich výcvik je tímto limitován, protože kromě vojenských zkušeností, návyků a znalostí si též musí udržovat své vlastní profesní znalosti a návyky. Vojenský výcvik se ve svém zaměření povětšinou nepotkává s civilními znalostmi a zkušenostmi. Ať již se jedná o taktiku, střeleckou přípravu, spojovací přípravu, výcvik pro ochranu před zbraněmi hromadného ničení nebo třeba pořadovou přípravu, nic z toho nemá průnik s civilním životem. Zdravotní a topografická příprava jsou jediné dvě oblasti, kde snad může dojít k určitému průniku, ale i tyto oblasti jsou ve vojenském pojetí specifické a odlišné od civilních.

Dalším faktorem ovlivňujícím výcvik je nestejná úroveň vycvičenosti vojáků v jednotce, která je ovlivněna jednak fluktuací vojáků a dále i frekvencí cvičení a možností se z nich za daných podmínek omluvit. To v praxi přináší situace, kdy je třeba pravidelně procvičovat základní znalosti a dovednosti vojáka. Aktuální rámec výcviku s tím počítá ve svém tříletém cyklu. Postupně se každý cyklus procvičují nejprve dovednosti vojáka jednotlivce, následně fungování na úrovni družstva a v konci cyklu fungování na úrovni čety. Tyto úrovně na sebe navazují a dovednosti z předchozích úrovní jsou nutné pro navázání na dalších úrovních.

V praxi to ovšem znamená, že člověk, který přijde do AZ v druhé nebo třetí části cyklu, nemusí mít zkušenosti pro kvalitní fungování v rámci aktuálního výcviku a po další dva roky je mít nebude. Zde je také rozdíl v základním výcviku, který musí vykonat každý zájemce o službu v AZ. Mladší ročníky zájemců, kteří neprošli základní vojenskou službou, nemají zkušenosti žádné, a proto jsou posíláni do Kurzu základní přípravy, který je zkrácenou variantou kurzu pro profesionální vojáky. Zde jsou většinou vycvičeni tak, že je snazší je začlenit do fungování již existující jednotky. Druhou skupinou zájemců jsou muži, kteří základní vojenskou službou prošli a v jejím rámci prošli i základním výcvikem. Tato povinnost byla v ČR zrušena ke konci roku 2004 a tak i poslední vojáci, kteří jí prošli, mají zkušenosti ze základního výcviku již skoro 15 let staré. Za tu dobu se mnohé návyky a postupy změnily a je

pro ně vhodné projít výcvikem na úrovni jednotlivce pro doplnění nových návyků a postupů.

Pokud odhlédneme od všech možných překážek a obtíží, kterými je nerovnoměrná vycvičenost, zkušenosti, motivace, intelekt, ale i množství cvičení, dostáváme se k samotnému vojenskému cvičení. Při předpokladu dvou cca týdenních cvičení za rok, dostáváme 14 dní na výcvik jednotlivce v prvním roce tříletého cyklu a dále pak porůznu procvičování těchto schopností v rámci výcviku v následujících dvou letech tohoto cyklu. Ani to není čistý čas použitelný pro výcvik, neboť do tohoto času se započítávají různé administrativní úkony spojené s výcvikem vojáka v AZ, jeho přeprava do a z výcvikového prostoru a různý čas na ošetření materiálu v průběhu cvičení. Nezřídka se též stává, že se v rámci cvičení připravuje ukázka výcviku pro armádní nebo politické činitele, která čas na samotný výcvik dále zkracuje.

Takto se lze dostat na cca 7-8 dní čistého času pro praktický výcvik jednotlivce za rok. Do tohoto času se musí směstnat časově náročný střelecký výcvik, taktická, spojovací, topografická, zdravotní příprava a výcvik v ochraně proti zbraním hromadného ničení (OPZHN). Z toho všeho vyplývá, že je nezbytné být při nácviku jednotlivých dovedností a schopností maximálně efektivní a ideálně využívat všechny dostupné prostředky pro jeho zjednodušení.

1.2 Spojovací příprava a používané formáty

Jednou z oblastí, ve kterých je voják cvičen, je spojovací příprava. Tento název v realitě AZ znamená především nácvik navazování spojení a předávání zpráv pomocí radiostanic. Při komunikaci pomocí radiostanic je nutné dodržovat daná pravidla. Jednak proto, že se jedná o otevřené médium a na stejné vysílací frekvenci se mohou snažit vysílat jiní uživatelé (proto je nutné nezahltit tento kanál) a zároveň zde může poslouchat kdokoliv (proto je nutné zprávy šifrovat).

Dle Příručky vojáka [1] má být komunikace pomocí jakéhokoliv prostředku včasná, věrohodná a utajená. Včasná znamená, že se dokáži spojit v daném časovém intervalu. To často závisí na okolním prostředí, ale i na ostatních účastnících radiového provozu. To, že má být utajená, je zdůrazněno na radiostanicích pomocí známého „Pozor! Nepřítel naslouchá!“a je tedy třeba neprozrazovat při vysílání žádné důležité informace. Pokud je třeba takové informace předat, je nutné je kódovat a šifrovat. Poslední zbývající bod je věrohodnost spojení. Zde se armáda odlišuje od názvosloví používaného u bezpečnostních praktik v IT, kde je věrohodností myšlena především možnost ověřit původ informace. V armádě je zde myšlena potřeba dokázat reprodukovat přenášenou zprávu s dostatečnou přesností.

Pro zrychlení a zpřesnění komunikace se využívají různé standardizované postupy. Jednak jsou využívány různé přesně dané formáty zpráv (ve vojenském prostředí často označovaných jako hlášení), určené k předání určité informace, o kterých je pojednáno dále v této kapitole. Dále jsou používány sjednocené formáty pro předávání konkrétních informací jako pozice nebo datum a čas. V současné době jsou tyto postupy v AČR i v AZ sjednoceny s postupy používanými v NATO. To umožňuje kooperaci se spojenci a v rámci zahraničních misí to je jediná možnost.

Příkladem mohou být třeba mise v Afghánistánu, kam jsou vysílány různé jednotky národů zastoupených v NATO a navzájem se svými specializacemi musí podporovat. Pokud tedy budou třeba čeští vojáci potřebovat povolat zdravotnickou pomoc (medevac), je jisté, že pro ně poletí buď americké nebo britské jednotky, protože to budou jediné jednotky s touto specializací v oblasti. Bez společného jazyka by se nedomluvili a bez sjednocených postupů by mohlo dojít k nedorozumění či prodlení, které v takové situaci může být fatální.

1.2.1 Formát dat

Zvyklosti v zapisování času a data se liší v různých zemích světa. V rámci vojenského prostředí se používá zápis zvaný jako date time group (DTG), někdy též military date time group nebo hovorově „Zulu time“. Je to zápis, který má následující formát:

$$ddhhmm(Z)MMMyy \tag{1.1}$$

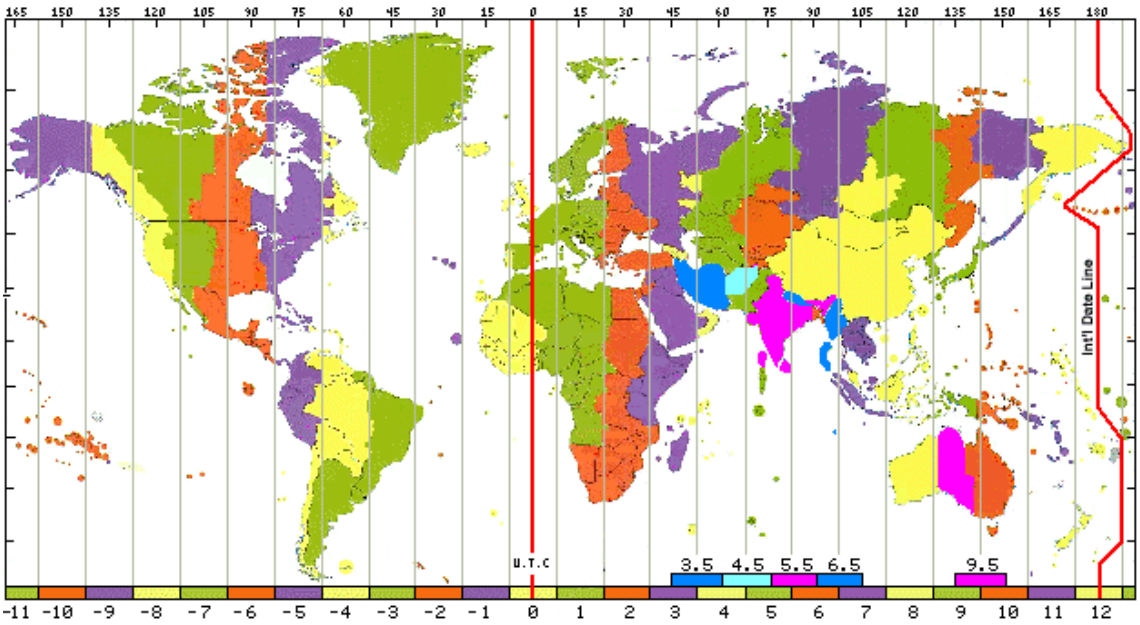
V tomto formátu je několik skupin, které znamenají:

- dd - den (01-31)
- mm - minuta (00-59)
- hh - hodina (00-23)
- (Z) - časové pásmo (A-Z)
- MMM - měsíc, vyjádřený zkratkou slova (JAN-DEC)
- yy - rok (00-99)

Pro příklad by tedy datum 23. února 2018 v čase 11:45 v pásmu z (např. v Londýně) odpovídalo následujícímu zápisu:

$$231145(Z)FEB18 \tag{1.2}$$

Z předchozího formátu stojí za zmínku právě časové pásmo, pro které se používá vojenské značení. To se sestává z 25 písmen anglické abecedy od a do Z, kdy je vynecháno J. Každé písmeno značí jedno pásmo, pásma odpovídají koordinovanému světovému času (UTC), viz obrázek 1.1. Jsou stanovena jako hodiny, o které je čas v daném pásmu napřed nebo opožděn oproti Greenwichskému hlavnímu času (GMT), který odpovídá pásmu a (posun +0h). Východně od pásma A, rostou pásma po hodinách od B (posun +1) až po M (posun +12). Západně od GMT klesá čas v pásmech od N (posun -1h) až po z (posun -12h). Písmeno J se nepoužívá z historických důvodů, neboť panovala obava, že se bude snadno plést s písmenem I. Pásma M a Z spolu sousedí.



Obrázek 1.1: Časová pásma UTC, převzato z [4]

Pokud bychom chtěli stejný čas zachytit v místním čase v Praze, přičetli bychom jednu hodinu (pásmo A, posun +1h) a zaznamenali, že se jedná o čas v pásmu A:

$$231245(A)FEB18 \tag{1.3}$$

Do tohoto logického a poměrně snadno srozumitelného systému se míchá ještě změna letního a zimního času v oblastech, kde je zaveden. V takovém případě se změna projevuje nikoliv přičtením hodiny (pro přechod na letní čas) nebo odečtením (přechod na zimní čas), ale změnou pásma. Pro příklad použijeme datum 23. června 2018 v čase 11:45. Čas Zulu (nulový posun) se nemění, ale pokud budeme vyjadřovat místní čas pro Londýn, již nejsme v pásmu Z, ale v pásmu A (letní čas je posunutý o 1 hodinu):

$$231245(A)JUN18 \tag{1.4}$$

Pro doplnění zde ještě uvedu, že letní čas v ČR se v tomto případě bude udávat v pásmu B, takto:

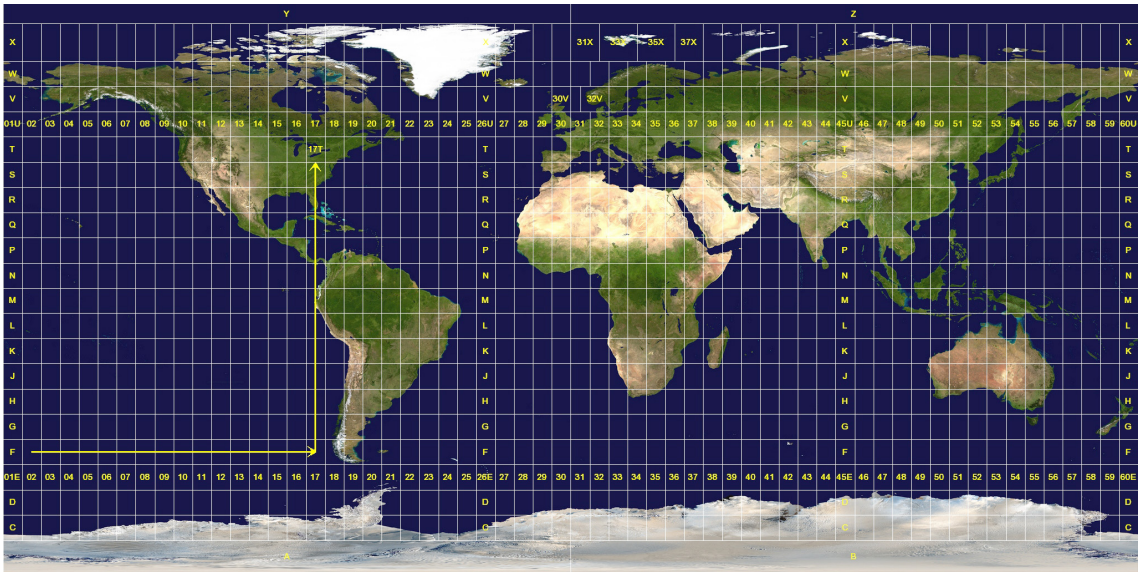
$$231345(B)JUN18 \tag{1.5}$$

1.2.2 Formát souřadnic

Pro jednoznačné určení místa na povrchu Země jsou v běžném životě především využívány zeměpisné souřadnice, a to zeměpisná šířka a zeměpisná délka. Tyto souřadnice vyjadřují úhly vzhledem k rovníku (počátku zeměpisné šířky) a vzhledem k nultému poledníku (počátek zeměpisné délky). Pokud uvažujeme Zemi jako kouli

nebo alespoň elipsoid, umožňují nám ještě za pomoci výšky určit naši pozici s dostačující přesností. Tyto souřadnice jsou využívány v civilních systémech a aplikacích pracujících s GPS a většina lidí je na ně již plně zvyklá.

Ve vojenském prostředí, mluvíme-li o státech sdružených v NATO, se používá jiný souřadnicový systém, nazvaný MGRS (military grid reference system). Jeho počátky se datují někdy do období druhé světové války. MGRS vychází z Univerzálního transverzálního Mercatorova zobrazení a stejně jako ono využívá zobrazení země jako částí elipsoidu do roviny. Toto zobrazení dělí zemi do šedesáti zón a ty dále rozděluje pomocí pásem na jednotlivé oblasti. Tyto oblasti jsou označovány jako čtverce a mají rozměr 100 x 100 kilometrů. Každý čtverec lze jednoznačně určit podle označení jeho zóny a pásma.



Obrázek 1.2: Zóny MGRS, převzato z [5]

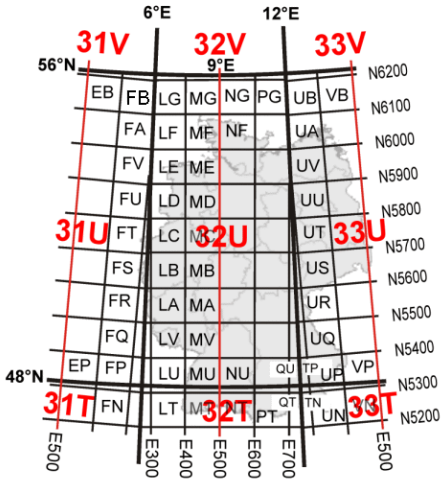
Pro podrobné určení pozice se dále určuje východní a severní souřadnice od počátku, kterým je jiho-západní roh čtverce. Tyto souřadnice poté tvoří dvojici n-tic, kde první n-tice určuje východní souřadnici a druhá severní souřadnici. Každá n-tice může mít dvě až osm cifer, které udávají přesnost. Jedna cifra, v každé souřadnici, znamená přesnost v řádu desítek km, zatímco pět cifer znamená přesnost v řádu metrů. Obě souřadnice musí mít vždy stejnou přesnost (stejný počet cifer). Při snižování přesnosti v systému MGRS se nezaokrouhluje, pouze se nejnižší cifry nepoužívají.

Pro příklad uvedu souřadnici sochy sv. Václava na Václavském náměstí v Praze:

$$33UVR59194765 \tag{1.6}$$

Na této souřadnici lze demonstrovat jednotlivé části, ze kterých se skládá. Předně je zde označení stokilometrového čtverce: $33UVR$, ve kterém $33U$ označuje zónu a VR pásmo. Tento čtverec má svůj počáteční bod zhruba 10 km jižně od Rokycan.

Údaj 59194765 dále popisuje, v jaké vzdálenosti od tohoto bodu se naše hledané místo nachází. Socha sv. Václava je ve vzdálenosti cca 59 km a 190 m směrem na východ a 47 km a 650 m směrem na sever s přesností na desítky metrů od počátečního bodu.



Obrázek 1.3: Ukázka značení 100 km čtverců, převzato z [6]

Tento souřadnicový systém má oproti standardní zeměpisné šířce a výšce jednu výraznou výhodu. Tou je jednoduchá možnost odečítání vzdálenosti dvou bodů v mapách a především určování jejich souřadnic. MGRS má výhodu v tom, že uživatel pracuje s mapou, která obsahuje souřadnicovou síť v jednotkách vzdálenosti a nikoliv se souřadnicemi danými úhlovými mírami. To zároveň umožňuje snadno spočítat vzdálenost dvou bodů za pomoci Pythagorovy věty.

1.2.3 Kódování zpráv

Při komunikaci pomocí radiového spojení přichází na řadu také otázka kódování a šifrování zpráv. Radiové spojení samo o sobě není nikterak zabezpečené a tak kdokoliv, kdo má naladěný stejný kmitočet jako vysílající, může poslouchat bez omezení všechny přenášené zprávy. Z toho důvodu je nutné šifrovat zprávy nebo je alespoň posílat zakódované. V běžné praxi AZ se nepoužívá šifrování, ale pouze kódování, kdy jednotlivým informacím jsou přiřazeny kódy a obě strany spojení znají kódová slova.

Tento systém funguje pro kódování významů zpráv, ale nepomůže se zakódováním různých číselných údajů. Například, pokud je třeba předat informace o počtu pozorovaných objektů nebo třeba souřadnice pozice. Jeden z postupů, na který jsem v praxi narazil, se nazývá RAMROD. Pokoušel jsem se k němu dohledat referenční materiál, ze kterého by mohlo být potvrzeno jeho užívání i v jiných jednotkách, ale nebyl jsem úspěšný. Tento postup využívá desetipísmenného kódovacího slova, které znají obě komunikující strany. Žádné z písmen se ve slově neopakuje a každému je

přiřazena číslice podle jeho pořadí, například takto:

$$\begin{array}{c} ALGORITMUS \\ \downarrow \\ 0123456789 \end{array} \tag{1.7}$$

Pokud je třeba zakódovat jakékoliv číslo, jednoduše se ve zprávě nahradí číslice za příslušné znaky a druhá strana je o této záměně informována. Případně pokud ví, že se jedná o číselný údaj, ani nemusí být informována. Výhodou tohoto systému je jeho jednoduchost a to, že lze čísla takto kódovat téměř zpaměti. Pro příklad souřadnice sochy sv. Václava zakódované pomocí kódovacího slova ALGORITMUS budou vypadat následovně:

$$\begin{array}{c} 33UVR59194765 \\ \downarrow \\ OOUVRISLSRMTI \end{array} \tag{1.8}$$

Druhou oblastí, která se blízce dotýká komunikace pomocí radiostanic, je věrohodnost. Kvalita rádiového spojení je silně závislá na různých faktorech, jako je poloha vysílajícího a přijímajícího, terén mezi nimi, povětrnostní podmínky atd. Často dochází k tomu, že kvalita není ideální a je nutné zprávy předávat znovu, případně je hláskovat. Hláskování lze s výhodou využít i pokud je třeba předat jednotlivé znaky. Samotný jeden znak by mohl v radiovém provozu zaniknout nebo splynout s předchozím či následujícím slovem a proto se i v této situaci využívá hláskování.

Pro hláskování existují standardní abecedy, jejichž slova jsou zvolena tak, aby nebyla jednoduše zaměnitelná mezi sebou a zároveň aby byla dobře rozpoznatelná i při nízké kvalitě spojení. NATO a anglofonní armády využívají standardní anglickou abecedu, s jejímž užitím se lze setkat i v AČR. Existuje i její český protějšek, který má slova pro čistě české hlásky. Poměrně vhodně lze ilustrovat použití hláskovací abecedy při předávání souřadnic MGRS, kdy již dříve uvedené souřadnice 33UVR59194765 budou pomocí radiostanice předávány jako „tři tři UNIFORM VICTOR ROMEO pět devět“ atd. V případě české hláskovací abecedy budou vypadat následovně „tři tři Urban Václav Rudolf pět devět“.

Obě hláskovací abecedy jsou v plném znění uvedeny v příloze k této zprávě.

1.3 Standardizovaná hlášení

Jak již bylo zmíněno na začátku předchozí podkapitoly, pro předávání informací existují standardizované postupy a formáty. Dále se tato podkapitola věnovala způsobu kódování a formátu jednotlivých informací. V rámci spojovací přípravy se vojáci učí používat ucelená hlášení, která znalosti z předchozí kapitoly využívají a dále je rozšiřují. Hlášení mají různý účel - předat informace o stavu jednotky, vyžádat si

zdravotnickou či jinou pomoc, informovat nadřízený stupeň o nastalé situaci a mnohé další.

Tato hlášení mají přesně daný tvar a posloupnost a některé informace jsou dokonce předávány pouze pomocí ustanovených kódů. Důvody pro používání takových hlášení jsou maximální efektivita (zkrácení vysílací doby) a zároveň snaha o předání všech nezbytných informací. Jasně daný formát pomáhá k tomu, že se nezapomene na žádnou podstatnou informaci, i když je vysílající pod stresem. Dobře je to třeba vidět na žádosti o zdravotnickou pomoc - Medevac 9-liner viz dále. Zároveň to usnadňuje komunikaci mezi jednotkami, útvary a řídícími stupni z různých spojeneckých armád.

V následující části budou popsána vybraná hlášení, která jsou nejčastěji procvičovaná při výcviku AZ a potenciálně by mohla být i nejčastěji používaná při reálném nasazení.

1.3.1 SALUTE

První hlášení, které zde bude popsáno, je označováno jako SALUTE. SALUTE je zpravodajské hlášení, které zasílá jednotka nadřízenému stupni, pokud jej potřebuje informovat o zpozorovaném protivníkovi. V tomto hlášení se snaží co nejlépe popsat zpozorované nepřátelské síly a jejich činnost. Jedná se o akronym z počátečních písmen anglických slov size, activity, location, uniform, time a equipment. Tato slova označují jednotlivé informace obsažené v hlášení.

- Size - velikost jednotky, popisuje počet kombatantů, vozidel a techniky
- Activity - činnost, kterou protivník v době pozorování provádí
- Location - pozice protivníka, udává se v MGRS
- Uniform - stejnokroje, vlajky, nášivky, označení jednotek, vozidel - vše, podle čeho je možné lépe určit protivníkovu jednotku
- Time - čas, kdy byl protivník zpozorován, ve formátu DTG
- Equipment - vybavení nesené protivníkem, zbraně, kterými disponuje atp.

1.3.2 SALTR

SALTR je hlášení, které nejen svým názvem souvisí s předchozím SALUTE. Vzniklo jako původně částečné hlášení, které mohlo být posíláno po odeslání SALUTE k doplnění či upřesnění předchozího hlášení. V praxi se význam těchto dvou hlášení začíná prolínat a mohou být využívána ve stejné roli. SALTR je rovněž akronym. Skládá se z anglických slov size, activity, location, time, request/remark. Oproti SALUTE se zde nehlásí uniform a equipment, které mohou být zahrnuté v size, a naopak zde přibývá request/remark, který je určen pro přidání doplňujících poznámek nebo žádostí k nadřízenému stupni.

- Size - velikost jednotky, popisuje počet kombatantů, vozidel a techniky
- Activity - činnost, kterou protivník v době pozorování provádí
- Location - pozice protivníka, udává se v MGRS
- Time - čas, kdy byl protivník zpozorován, ve formátu DTG
- Request/remark - doplňující poznámky nebo žádost k nadřízenému stupni

1.3.3 Sitrep

Označení sitrep je zkratkou z situation report a popisuje účel tohoto hlášení. Jedná se o informace o stavu jednotky posílané nadřízenému stupni. Sitrep se odesílá obvykle buď v pravidelných, předem určených intervalech, nebo si jej může vyžádat velící v situaci, kdy je třeba potvrdit aktuální stav jednotky. Sitrep se sestává z následujících informací:

- Time - čas, kdy byl protivník zpozorován, ve formátu DTG
- Unit status - stav jednotky udávaný pomocí barevného kódu (viz následující odstavec)
- Enemy situation - stav a činnost, kterou aktuálně provádí protivník
- Own situation - stav a činnost, kterou provádí moje jednotka
- Location - pozice mojí jednotky, udávaná v MGRS
- Following actions - zámysl, činnost, kterou mám v plánu

Pro předání informace o aktuálním stavu mojí jednotky se používá barevný kód. Každá barva označuje procentuální rozsah, který udává velikost bojeschopné části jednotky. Ztráta bojeschopnosti může být způsobená počtem zraněných či padlých, ale například i ztrátou nebo poškozením kritického vybavení a materiálu. Stupnice jde od zelené až po černou a v hlášení jsou předávána anglická slova pro tyto barvy.

- Green (100-90%) - jednotka je plně bojeschopná
- Amber (90-75%) - jednotka utrpěla ztráty, ale je schopná dokončit zadaný úkol
- Red (75-60%) - jednotka utrpěla ztráty v takovém rozsahu, že dokáže úkol splnit jen s obtížemi
- Black (60% a méně) - jednotka není schopna pokračovat v plnění úkolu

1.3.4 Medevac 9-liner

Všechna předchozí hlášení mají sice danou strukturu, ale ta se pro různá nasazení může měnit a být různě upravována a doplňována. To je vidět ostatně i na SALUTE a SALTR. Oproti nim je tu Medevac 9-liner. Toto hlášení má strukturu přesně danou a je nezbytné ji dodržovat. Jedná se o hlášení podávané při potřebě vyžádání zdravotnické pomoci. Název je odvozen ze složeniny slov medical evacuation (zdravotnický odsun) a 9-liner, což je označení pro devítibodové hlášení.

Medevac 9-liner by měl obsahovat všechny nezbytné informace k tomu, aby mohla být vyslána zdravotnická pomoc. Jedná se o informace stran počtu zraněných, vážností zranění a požadovaného záchranného materiálu a nářadí. Protože jde o vojenské hlášení, obsahuje rovněž informace o taktické situaci a příslušnosti zraněných. Toto vše je shrnuto do devíti bodů:

Line 1 - místo vyzvednutí, zadané pomocí souřadnic MGRS

Line 2 - volací znak a radiová frekvence, na které lze jednotku kontaktovat

Line 3 - počet pacientů, podle závažnosti jejich zranění

- **A** - Urgent - vyžaduje chirurgický zásah v průběhu transportu
- **B** - Urgent surgical - život ohrožující zranění, které lze stabilizovat pro transport
- **C** - Priority - vážné zranění neohrožující bezprostředně život
- **D** - Routine - transport zraněného, kterého není možné transportovat vlastními prostředky jednotky
- **E** - Convenience - lehké zranění nebo transport k běžnému ošetření

Line 4 - požadované speciální vybavení

- **A** - None - žádné
- **B** - Hoist - jeřáb/naviják
- **C** - Extraction equipment - vyprošťovací vybavení
- **D** - Ventilation - zařízení pro zajištění dýchání

Line 5 - počet pacientů podle jejich mobility

- **A** - Ambulatory - samostatně pohybliví
- **L** - Litter - pacienti na nosítkách

Line 6 - bezpečnostní situace v místě vyzvednutí

- **N** - v okolí není žádný protivník
- **P** - možný výskyt protivníka v okolí - přibližte se obezřetně
- **E** - protivník v okolí - přibližte se obezřetně

X - protivník v okolí/probíhající kontakt s protivníkem - je nutný ozbrojený doprovod

Line 7 - způsob označení místa vyzvednutí

A - Signalizační panel

B - Pomocí pyrotechnického signálu (světlice, oheň atp.)

C - Kouřový signál

D - Žádný signál

E - Jiné označení - zde je vhodné doplnit jaké

Line 8 - počet pacientů podle jejich národnosti a statutu

A - vojáci US armády

B - američtí civilisté

C - vojáci koaličních armád

D - ostatní civilisté

E - váleční zajatci

Line 9 - ohrožení zbraněmi hromadného ničení v místě vyzvednutí

N - jaderné zamoření

B - biologické zamoření

C - chemické zamoření

V případě použití v době míru se v tomto hlášení v Line 6 udává počet a typ jednotlivých zranění a v Line 9 se udává charakteristika terénu v okolí místa vyzvednutí. Celé hlášení se předává pouze formou „Line - příslušný údaj“. Například by předání tohoto hlášení mohlo znít následovně: „...mám připravený Medevac 9-liner. Line 1 - 33UVR45789632, Line 2 - 465.125 Mhz, Dagger 5, Line 3 - Bravo 1, Charlie 2“ atd.

1.3.5 UXO/IED 9-liner

Druhým frekventovaným devítibodovým hlášením je tzv. UXO/IED 9-liner. Akronym UXO pochází z anglického unexploded ordnance a označuje nevybuchlou munici jakéhokoliv druhu, od pěchotní munice, přes granáty až po minometné miny, rakety a dělostřelecké granáty, nalezené při pohybu v terénu. Akronym IED je opět z angličtiny a skládá se ze slov improvised explosive device, tedy improvizované výbušné zařízení, nástraha. Toto hlášení se používá v případě odhalení výbušniny v terénu a obsahuje informace o nálezu, jeho okolnostech a o dalším postupu jednotky. Opět je vše shrnuto do devíti bodů:

Line 1 - čas nálezu udávaný pomocí DTG

Line 2 - identifikátor jednotky, volací znak a místo, kde došlo k nálezu

Line 3 - volací znak a radiová frekvence, na které lze jednotku kontaktovat

Line 4 - druh nalezené výbušniny

- svržená - bomby, kazetová munice, zařízení pro rozprášení látek
- vystřelená - střely, minometné miny, rakety, puškové granáty
- umístěná - protipěchotní a protitankové miny, nástrahy
- hozená - granáty, výbušky

Line 5 - prostředky a zdroje důležité pro dokončení úkolu, které jsou ohroženy

Line 6 - byla nalezena kontaminace zbraněmi hromadného ničení? Jaké látky byly identifikovány?

Line 7 - jak ohrožuje nebo omezuje nález splnění úkolu

Line 8 - ochranná opatření a prostředky, které byly zavedeny

Line 9 - doporučená priorita hrozby

- okamžitá
- nepřímá
- podružná
- žádná

1.4 Hlášení při výcviku aktivních záloh

Všechna tato uvedené hlášení jsou využívána v AZ a vojáci se je učí, aby je dokázali používat při dalších částech výcviku a při případném ostrém nasazení. Standardně jsou vojáci nejdříve s jednotlivými hlášeními seznámeni a je jim vysvětlen jejich význam i jednotlivé body každého hlášení. Dále se učí je předávat při výcviku ve spojovací přípravě a posledním stupněm výcviku je, že jsou tato hlášení zakomponována do navazujícího komplexního výcviku. Zde již se předpokládá jejich znalost a schopnost vojáků je adekvátně použít v rámci aktuální situace při vykonávání dalších povinností.

1.4.1 Modelová situace

Vzorovým příkladem komplexního nácviku je například modelová situace, kdy se malá jednotka AZ pohybuje terénem a provádí průzkumnou a hlídkovací činnost. Během této činnosti je napadena protivníkem pomocí nástřelu z lehkých zbraní. Jednotka opětuje palbu a velitel po okamžitém zhodnocení situace dává povel ke stažení a udává směr a vzdálenost. V průběhu stahování je jeden z vojáků raněn

a musí být evakuován. Jednotka se stáhne do bezpečného místa, zaujme obranné postavení a provede ošetření raněného kolegy. Spolu s tím vydává velitel povel spojařovi, aby předal nadřízenému stupni hlášení o napadení (zde je vhodný Sitrep) a vyžádal zdravotnickou evakuaci raněného (Medevac 9-liner). Při takovém cvičení uplatňují vojáci své zkušenosti a znalosti z oblasti střelecké a taktické přípravy, dále pak ze zdravotnické a v závěru také spojovací přípravy.

Od okamžiku napadení se vše odehrává ve značné rychlosti a pod tlakem, způsobeným stresem z napadení a nutností adekvátně reagovat. Každý z vojáků je v takové chvíli především „střelcem“, který musí spolu s kolegy odvrátit a potlačit útok nepřítele. Dále pak odbornosti zdravotníka a spojaře mají povinnosti plynoucí z jejich specializací. Obdobně velitel musí, kromě role „střelce“, především velet a vydávat povely všem vojákům. Pokud by on v danou chvíli nefungoval, vše se může rozpadnout v úplný chaos. Pokud by byl vyřazen z boje (těžce raněn nebo zabit), musí jeho zástupce převzít velení a vést jednotku. Podobně je třeba nahradit kteroukoliv další odbornost v rámci jednotky (např. spojař, kulometník atd.), pokud je vykonávající voják vyřazen z boje.

1.4.2 Příprava hlášení v terénu

Z toho vyplývá, že je třeba, aby všichni vojáci měli naučené základní znalosti a dokázali zastupovat více funkcí, než jen pouze svoji. Znalost spojovací přípravy a předávání hlášení je jednou z nich. Ve standardní praxi AZ nyní příprava hlášení vypadá následovně. Velitel potřebuje předat hlášení nadřízenému stupni. Proto si k sobě zavolá spojaře a předá mu informace, které potřebuje předat.

Spojař si v tuto chvíli vše poznamená na papír a následně si připraví všechny nezbytné informace pro dané hlášení. Pokud je součástí hlášení datum, musí je převést do formátu DTG. Pokud se v hlášení vyskytuje informace o aktuální pozici, je třeba si připravit její souřadnice ve tvaru MGRS. To obvykle znamená vyhledat a odečíst je z papírové mapy. Pokud jsou v rámci cvičení používána kódová označení a RAMROD, je třeba připravené hlášení podle nich zakódovat. Toto opět probíhá za použití papíru a tužky. Ve chvíli, kdy je spojař připraven, naváže spojení s nadřízeným stupněm a zakódované hlášení mu předá. Některé části hlášení jsou předávány pomocí hláskování, obzvláště v případě horší kvality spojení.

1.4.3 Zjednodušení přípravy hlášení

Velká část činnosti spojaře při přípravě hlášení je poměrně rutinní záležitost, od vyhledávání pozice v mapě, přes sepsání hlášení ve standardním tvaru až po zakódování číslic pomocí RAMRODu. Zkušený spojař by měl být schopný zvládat vše bez potřeby hledat si formáty hlášení atp. Realita AZ v tomto ohledu klade vyšší nároky na osobní iniciativu, protože není možné v rámci vymezených cvičení vycvičit všechny vojáky dostatečně ve všech odbornostech, ideálně ještě tak, aby si vše pamatovali. Proto velká část lidí používá různé vlastní poznámky a konkrétně standardizovaná hlášení jsou v nich často obsažena, stejně jako tvar data ve formátu DTG. Všichni

vycvičení vojáci dokáží zakódovat číslice ve zprávě pomocí RAMRODu a měli by být schopni určit souřadnice své pozice z mapy.

Velkou část úkonů, vykonávaných při přípravě hlášení, je možné zjednodušit při použití moderních technologií. Souřadnice aktuální pozice lze snadno zjistit za pomocí GPS lokátoru, který umí určit souřadnice v systému MGRS. Různé informace jako formáty hlášení nebo data lze mít uložené např. v chytrém telefonu. Datum ve formátu DTG může být automaticky zformátované vhodným programem atp. Všechny tyto činnosti jsou více či méně složité operace nad vstupními daty, která jsou zadaná uživatelem nebo daná jeho reálnou situací (kde se nachází, jaký je čas a datum, atd.).

2 Existující použitelná řešení

Zkušenosti nabyté na cvičeních AZ spolu se znalostmi z oblasti informatiky mě vedly k zamyšlení, zda by nebylo možné přípravu hlášení automatizovat a tím zefektivnit. Vhodný přístroj nebo aplikace by mohly samy uživateli připravit části, které je jinak nutné složitě připravovat. Určitě by mělo být snadné zformátovat datum do formátu DTG nebo zakódovat číslice pomocí RAMROD. Získávání zeměpisné pozice je dnes dostupné pomocí GPS v celé řadě chytrých zařízení a i příprava hlášení by mohla být upravena do podoby formuláře, který je následně uživateli upraven do formy, kterou již může použít dále. S těmito úvahami jsem se pokusil nalézt již existující řešení, které by bylo možné využít.

2.1 Volně dostupné aplikace

Jako první směr hledání jsem zvolil aplikace dostupné pro chytrá zařízení (telefony, tablety, případně hodinky). Výhodou takového řešení je snadná dostupnost hardware. Většina lidí již dnes vlastní alespoň smartphone a pro příslušníky AZ platí to samé. Svoje hledání jsem omezil na operační systémy Android zastřešený společností Google a iOS od firmy Apple.

2.1.1 Aplikace pro Android

Výhodou operačního systému Android je jeho otevřenost a široká dostupnost. Zařízení s tímto systémem se pohybují v celém cenovém spektru a tak je dostupný téměř komukoliv. Díky tomu jej využívají ve svých telefonech vlastníci napříč sociálním spektrem, od chudších až po zámožné. Příslušníci AZ toto spektrum poměrně věrně kopírují a často mají telefony s tímto systémem.

Výraznou vlastností této platformy je také její otevřenost, která umožňuje téměř komukoliv vytvořit vlastní aplikaci a za nízký poplatek ji umístit na oficiální obchod Google Play. Velká většina zařízení rovněž disponuje GPS přijímačem a umožňuje uživateli snadné určení vlastní pozice. S těmito vlastnostmi jsem očekával, že výsledek mého hledání nemůže být neúspěšný.

Při hledání aplikací jsem zkoumal pouze oficiální distribuci skrze Google Play, protože toto je doporučený způsob získávání nových aplikací. Zároveň je nabídka Google Play do jisté míry kontrolovaná společností Google a aplikace v ní by neměly obsahovat škodlivý kód. Aplikace jsem hledal pomocí klíčových slov jako: *report*,

military, *army*, *tactical*, *assistant*, *medevac*9 − *liner*, *ied/uxo*9 − *liner*, *sitrep*, *MGRS* atp. Výsledek hledání by se dal rozdělit do cca tří skupin.

První skupina nalezených aplikací byly aplikace navigačního charakteru, které umožňovaly práci se souřadnicovým systémem MGRS. Na výběr je od jednoduchých aplikací, které pouze zobrazují souřadnice v tomto systému a další přidružené geolokační údaje, až po komplexní navigační aplikace s možností přidávání vlastních bodů do mapy a práci s nimi pomocí MGRS souřadnic nebo pomocí standardních údajů o zeměpisné šířce a délce. Za všechny zde uvedu pouze několik příkladů:

- Grid GPS [7] - jednoduchá aplikace zobrazující pouze souřadnice
- Koord Konverter [8] - konvertor souřadnic s jejich zobrazením v mapě
- Land Nav Assistant [9] - komplexní aplikace pro navigaci s možností MGRS
- Personal Eye System [10] - aplikace pro navigaci a sdílení mapy včetně možnosti zaznamenávání zájmových bodů

Druhá skupina aplikací fungovala spíše jako kapesní průvodce a slovníky. Tyto aplikace uživateli nabízely různé přehledy vojenských akronymů, zkratek, termínů a hesel. Některé také obsahují popis různých operačních postupů a další znalosti, které může voják potřebovat. Pro účely mojí rešerše je důležité, že obsahovaly popis jednotlivých formátů hlášení. Jako poměrně slušné řešení se mi jevily např. tyto:

- Military Acronym Reference Guide [11] - slovník vojenských akronymů
- Marine Corps Pocket Knowledge [12] - kapesní příručka pro příslušníky americké námořní pěchoty, vytvořená jejími příslušníky
- Army Leader Smart Cards [13] - informace uspořádané ve formě stručných karet, obsahující jednotlivá hlášení

Třetí skupina aplikací, která se nejvíce blížila mým představám, již zastupovala samotná vojenská hlášení. Podařilo se mi objevit pár aplikací, které umožňovaly přímo sestavení hlášení. Bohužel jsem nalezl pouze aplikace pro Medevac 9-liner - žádost o lékařskou evakuaci, žádná další hlášení se mi nepodařilo nalézt, a i těch bylo pomálu:

- Army Reports [14] - aplikace pro přípravu Medevac 9-liner, autor má v plánu doplnit hlášení SALUTE
- 9-Liner [15] - další aplikace pro přípravu Medevac 9-liner

2.1.2 Aplikace pro iOS

Druhým nejrozšířenějším operačním systémem pro chytrá zařízení je iOS. I přes nižší zastoupení na trhu je stále velmi oblíbenou volbou a oproti Androidu je více uzavřený a restriktivní ve vztahu k vývojářům. Oproti Androidu je třeba splnit více kritérii k tomu, aby mohl vývojář umístit svou aplikaci na distribuční platformu - App Store. Aplikace jsou více kontrolovány, aby koncový uživatel nebyl zahlcen množstvím nekvalitních či dokonce podvodných produktů.

Zařízení se systémem iOS jsou vyráběna pouze samotným Applem a pohybují se ve vyšší cenové hladině. Nicméně, pro svoji prestiž a oblíbenost u uživatelů, jsou zastoupena i ve vrstvách s nižšími příjmy. Podobně jako u Androidu i tato zařízení jsou používána příslušníky AZ, oproti Androidu však v nižší míře, stejně jako v běžné populaci. Pro svoje hledání jsem uvažoval pouze aplikace z oficiálního App Store a hledal jsem pod stejnými hesly jako v případě Androidu.

Výsledky hledání byly rozvrstveny obdobně jako u systému Android a některé aplikace byly pro oba systémy dokonce shodné. Pouze počet nalezených aplikací byl, nikterak překvapivě, nižší.

I na App store se objevila skupina navigačních aplikací umožňujících práci se souřadnicovým systémem MGRS:

- GPS Utility [16] - aplikace pro převod mezi souřadnicovými systémy
- Tactical NAV [17] - komplexní navigace v souřadnicovém systému MGRS s možností zadávání vlastních bodů a značek
- MilGPS [18] - navigace v souřadnicovém systému MGRS, možnost přidávání vlastních bodů

Rovněž aplikace v podobě příruček jsou zde zastoupeny:

- Army Ranger Handbook [19] - příručka pro americké Rangers
- Military Terms & Acronyms [20] - slovníček vojenských zkratek a akronymů
- Army Leader Smart Cards [21] - vojenská příručka, obsahující hlášení

Pro sestavování samotných hlášení se mi nepodařilo nalézt na App store žádného zástupce. Jediný, opět pro hlášení Medevac 9-liner, byl z App Store stažen a je dostupný pouze z jiných zdrojů:

- 9 Line MEDEVAC Training App [22] - aplikace pro sestavení hlášení Medevac 9-liner

2.2 Nástroje a aplikace používané ozbrojenými složkami

Druhým směrem hledání byl pokus nalézt existující řešení používaná v profesionální armádě. Zde jsem se rozhodl zmenšit okruh plánovaného hledání pouze na AČR. Výsledky hledání z mezinárodního pole by pro použití v AZ, kde se lze s jistotou spolehnout pouze na znalost českého jazyka, neměly moc význam. Cílem bylo pokusit se zjistit, zda AČR využívá systém či aplikaci, která by nějakým způsobem pomáhala jednotlivým vojákům v plnění jejich povinností v poli.

Z osobních zkušeností mých ani mých kolegů o žádném takovém nevím. Armáda v rámci základního výcviku využívá pouze tištěné příručky vojáka [1] a zbytek výkladu probíhá formou monologu. Z veřejně dostupných zdrojů lze dopátrat, že jsou v rámci armády zaváděny systémy na různých úrovních, ať již velitelské nebo vozidlové. V rámci soukromé komunikace s lidmi z Univerzity obrany v Brně se podařilo zjistit, že byl vyvíjený systém s názvem BADIAN pro sesednuvšího vojáka. Systém byl vyvíjen společností Delinfo, která je součástí skupiny ICZ Group, ale o jeho reálném použití v armádě se mi nepodařilo zjistit více.

2.3 Zhodnocení nalezených existujících řešení

Celkově se mi v rámci rešerše použitelných nástrojů nepodařilo najít žádný, který by se mi jevil jako vhodný pro použití v rámci výcviku AZ. Nástroj, který bych považoval za vhodný, by měl umět automaticky získávat údaje o geografické poloze a čase a dokázat je uživateli připravit ve standardizovaném formátu. Dále by měl dokázat nabídnout alespoň nejčastěji využívaná vojenská hlášení, aby si uživatel nemusel pamatovat všechna včetně podrobností. Pro rozumné použití v rámci AZ je také nezbytné, aby takový nástroj byl lokalizován do českého jazyka (podobný požadavek by platil pravděpodobně i pro univerzální použití v armádě). Nástroj, který by splňoval všechna tato kritéria, bych považoval za rozumně použitelný.

Ve skupině civilně dostupných aplikací se podařilo najít více možných použitelných nástrojů, žádný z nich však nesplňoval všechna kritéria. Hlavním vyřazujícím kritériem by pro použití v AZ byl univerzálně český jazyk. Žádné z nalezených řešení nebylo v českém jazyce. Při odhlédnutí od tohoto kritéria se nejblíže dostaly aplikace pro přípravu hlášení Medevac 9-liner, které byly funkčně blízko, ale jednalo se pouze o jediný druh hlášení. Zbylé nalezené možnosti mi nepřišly nijak zvláště atraktivní pro to, aby mohly samy o sobě nahradit zavedenou praxi ručně připravovaných psaných poznámek. Jisté usnadnění by zde představovaly aplikace pro určení polohy disponující souřadnicovým systémem MGRS, ale v tomto případě bych oproti aplikaci na chytrém telefonu nebo hodinkách asi zvolil specializovanou navigaci, která tuto funkci zvládá též.

V oblasti profesionálních vojenských nástrojů by bylo možné najít vhodné nástroje. Obzvláště pokud, stejně jako u civilních aplikací, polevíme z podmínky českého jazyka. Zde je bariérou pro využití v AZ jejich dostupnost. Tím, že se jedná o specifické, úzce zaměřené nástroje roste jejich cena. Ta roste i s tím, že se často

jedná nejen o aplikace, ale i o konkrétní hardware, podléhající přísným standardům a normám. Cena a dostupnost je z pohledu AZ problematická, vezmeme-li v úvahu to, že je někdy problém pro vojáky v AZ zajistit i běžné výstrojní součástky (viz vlastní autorova zkušenost s obuví - tuto poznámku si dovolím sem vložit pro pozorného čtenáře).

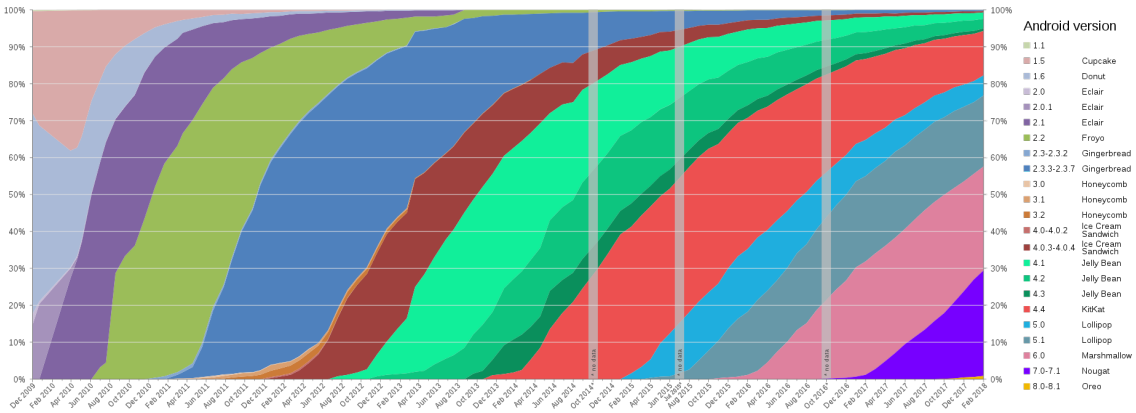
Rešerše s poměrně negativními výsledky mi potvrdila smysluplnost zadání, nad kterým jsem přemýšlel. Napadlo mě využít rozšířenost chytrých telefonů s dostatečnými technickým prostředky a propojit ji se svými zkušenostmi z obou oborů, jak vojenského, tak technického - informatiky, a vytvořit aplikaci, která by splňovala dříve uvedená kritéria. Výhodou tohoto řešení by byla naprostá míra splnění požadavků, které vnímám jako potřebné, a zároveň dostupnost řešení, kdy mnou dodaná aplikace by mohla běžet v rámci hardware, který si může kdokoliv snadno opatřit nebo jej již vlastní.

3 Návrh aplikace

V této kapitole bych rád popsal prostředky, které jsem se rozhodl využít pro realizaci svého řešení. Postupně bych chtěl popsat něco o výběru cílových zařízení a jejich operačního systému. Dále se zmínit o programovacím jazyce, který byl jedním z důležitých důvodů vedoucích k mému rozhodnutí, a poté bych chtěl popsat z uživatelského hlediska aspekty, které by hotové řešení mělo splňovat, ale nebude zde ještě popsána jejich implementace. Od této kapitoly dále také budu o výsledném řešení psát jako o aplikaci, protože to je hlavní část mé práce.

3.1 Volba systému

První podstatnou volbou před implementací aplikace bylo rozhodnutí, který z aktuálně používaných operačních systémů zvolit a vytvořit pro něj aplikaci. V úvahu připadaly systémy Android a iOS, jakožto dva nejpoužívanější systémy. Okrajově by bylo možné zvážit ještě systém Windows Phone v jeho nejnovější verzi Windows 10. Technické prostředky nutné pro implementaci (především polohovací systém) mají zařízení se všemi třemi systémy a použitelné by byly všechny. Pro aplikaci jsem zvolil Android ze dvou důvodů. Objektivně je to nejrozšířenější operační systém, který i druhý iOS stále předbíhá minimálně dvojnásobně v počtu uživatelů. Toto se projevuje i v jednotce AZ, ve které sloužím. Druhým důvodem pro volbu jsou prostředky, které Android nabízí vývojářům aplikací a se kterým jsem již před touto prací pracoval a o kterých se též zmíním dále.



Obrázek 3.1: Používané verze systému Android podle roku, převzato z [23]

Spolu s volbou Androidu bylo nutné zvážit, pro kterou jeho verzi budu aplikaci vyvíjet. Obecně platí, že novější verze systému nabízejí více možností a lépe vyladěné API. Naopak při volbě starší verze je možné obsáhnout více zařízení, protože jednotlivé verze jsou zpětně kompatibilní a aplikace pro starší verzi systému lze většinou provozovat i na novějších verzích systému. Pro přehled je zajímavé se podívat na to, jak jsou jednotlivé verze systému zastoupeny mezi uživateli. Poměrně přehledné porovnání lze nalézt například v grafu viz 3.1. Jako vhodný kompromis mezi aktuálností API a množstvím cílových zařízení jsem zvolil verzi 4.4 (Kitkat). Android v této a vyšších verzích používá kolem 90% všech jeho uživatelů.

3.2 Volba zařízení

Jeden z prvních nápadů, který jsem při počátečních fázích promýšlení aplikace zvažoval, byla implementace na chytré hodinky (smart watches). Představa vojáka, který má na zápěstí asistenční program, který je doslova po ruce a stále dostupný, byla lákavá. Zároveň by to mohlo být smysluplné využití prostředků, které takový hardware nabízí. Po zvážení různých aspektů jsem od této myšlenky upustil a přiklonil se ke standardnějšímu řešení.

Důvody, proč jsem opustil myšlenku na využití smart watches, bych zde rád vysvětlil. Jako hlavní plus vidím to, že hodinky by nabízely vysokou pohotovost, uživatel je má stále na ruce a nemusel by je hledat někde po kapsách či ve výstroji, což je vlastnost, kterou vojáci v průběhu výcviku ocení. Jejich nízké rozměry a hmotnost by vojáka nijak nadměrně nezatěžovaly. Prostě by jen mohly nahradit standardní hodinky a byl by to další evoluční krok ve vylepšování výstroje.

Jejich rozměry bohužel zatím mají velký vliv na jejich další vlastnosti, které jsou pro použití v taktickém prostředí (i výcvikovém) negativní. Předně malé rozměry určují maximální velikost displeje a i s velkým rozlišením neumožňují zobrazení většího množství informací najednou. S vhodným designem aplikace by se tato slabina dala pravděpodobně odstranit. Větší nevýhodou by bylo obtížné zadávání delších textů, které se mohou v hlášeních vyskytnout, ale na které zatím hodinky nejsou připravené.

Jako největší nevýhodu vidím aktuálně nízkou kapacitu baterií, na kterou si uživatelé u telefonů již do určité míry zvykli a v případě možnosti každodenního dobíjení ji akceptují. To ovšem není vždy možné v podmínkách výcviku a voják by tak buď musel řešit další logistiku ohledně pravidelného dobíjení (pomocí power banky, adaptéru do vozidla, atp.) nebo počítat s tím, že má hodinky pouze na omezenou dobu a dále se bez nich musí obejít. Obejít se bez hodinek považuji pro vojáka za neakceptovatelné. Navyšovat množství věcí, o které musí speciálně pečovat, mi nepřipadá jako rozumné, a proto jsem určil za cílové zařízení chytrý telefon. To také koresponduje s tím, že chci použít hardware, který většina lidí již vlastní.

Samozřejmě v případě Androidu od verze systému 4.0 není třeba rozlišovat mezi tablety a telefonem. Aplikace od této verze lze spustit na telefonech i tabletech. Jediným rozdílem zůstává velikost obrazovky a případné využití této skutečnosti může být výhodou, pokud je aplikace např. optimalizovaná pro tablety. Mnou vyví-

jenou aplikaci tedy bude možné používat i na tabletech, i když v první verzi určitě neplánuji cílenou optimalizaci pro různé velikosti obrazovek.

3.3 Programovací jazyk Kotlin

Dalším důvodem, který mě vedl k volbě Androidu jako cílové platformy, byl programovací jazyk, ve kterém budu aplikaci vytvářet. Shodou okolností byl v květnu 2017 na konferenci Google I/O oznámen nový oficiálně podporovaný jazyk pro platformu Android. Ke stávajícím jazykům, kterými jsou Java a C++, přibyl Kotlin, který se pokusím alespoň lehce přiblížit v následujících odstavcích. Kotlin je zajímavý více věcmi, ale pro mě především tím, že navazuje na Javu a tvrdí o sobě, že odstraňuje některé její nedostatky. Jako programátor již mám nějaké zkušenosti s psaním aplikací v Javě a toto tvrzení upoutalo mojí pozornost.

Psát aplikace pro Android lze nejen v oficiálně podporovaných jazycích, ale i v dalších, pro které existují překladače. Podstatnou výhodou oficiálně podporovaných jazyků však je větší komunita a také kvalitní vývojové prostředí Android Studio, dodávané Googlem, které značně usnadňuje vývoj a zároveň je neplacené. Nechci zde dělat reklamu společnosti Google, pouze popisuji další důvody, proč jsem se rozhodl aplikaci implementovat s těmito nástroji.

3.3.1 Představení jazyka

Programovací jazyk Kotlin je vyvíjen společností JetBrains, známou pro vývojové prostředí Idea. Je vyvíjen cca od roku 2010 a veřejnosti byl poprvé představen v roce 2011. Původní záměr vývojářů bylo používat jazyk, který bude mít všechny jimi požadované funkce. Jediným vhodným adeptem byla Scala, která trpí poměrně pomalou kompilací. Proto se autoři Kotlinu rozhodli vytvořit vlastní jazyk, který bude dostatečně odolný pro nasazení na profesionální úrovni. Výsledkem jejich práce je staticky typovaný programovací jazyk, který může být kompilován pro spuštění na Java Virtual Machine nebo do JavaScriptu. Do budoucna se počítá, že by měl mít i stabilní možnost překladu přímo do strojového kódu.

Jedním z cílů, které si autoři stanovili, je jeho kompatibilita s Javou, tak aby již existující kód napsaný v Javě mohl být postupně, po částech přepisován do Kotlinu a zůstával stále funkční. Touto vlastností se snažili zjednodušit proniknutí Kotlinu do firemní sféry. Při kompilaci pro JVM tak lze využít jeho kompatibility s Javou, a používat knihovny psané pro Javu v Kotlinu i naopak.

Dalším z cílů bylo vytvořit moderní, efektivní jazyk, který budou vývojáři rádi používat. Jako hlavní charakteristiky, které by měly vývojáře nalákat k použití Kotlinu, uvádí jeho autoři na své stránce [2] tyto čtyři vlastnosti:

1. Stručnost - jazyk odstraňuje nutnost psát množství efektivně prázdného kódu, který pouze obaluje reálnou funkcionalitu. Dobrým příkladem je snazší vytváření jednoduchých objektů (POJO), singletonů nebo třeba možnost neukončovat řádky středníkem.

2. Bezpečnost - svým návrhem zabraňuje tomu, aby vznikaly různé dříve časté chyby - například z Java nechvalně proslulý *NullPointerException*. Objekty v Kotlinu musí být defaultně nenullové a tato vlastnost je kontrolována již při kompilaci kódu. Dalším programátorským chybám je předcházeno tím, že je kladen důraz na neměnnost objektů.

3. Interoperabilita - Kotlin je kompatibilní s knihovnami v Javě, JavaScriptu nebo s Androidem.

4. Použitelný s různými nástroji - kód je možné psát a kompilovat v kterémkoliv vývojovém prostředí pro Javu nebo i přímo z příkazové řádky.

Samozřejmě toto nejsou jediné vlastnosti Kotlinu, je jich mnohem více a ve stručnosti se je pokusím zde popsat, bez jakéhokoliv pořadí. Kotlin umožňuje stanovit defaultní hodnotu argumentů funkcí. Argumenty funkcí lze pojmenovat a pojmenování využít nejen pro přehlednost kódu, ale i při přiřazování spolu s defaultními hodnotami. Kotlin má možnost psát funkce vyšších řádů (takové funkce, které mají jako argument jiné funkce). Umožňuje vracet více než jednu hodnotu z funkce - toto se nazývá destrukturalizační deklarace. Rozšiřující funkce umožňují rozšířit již existující komponenty o další funkcionalitu bez nutnosti dědění atp. Funkce mohou obsahovat lokální funkce, platné pouze v jejich rozsahu, nebo naopak je možné použít funkce vytvořené pouze jedním výrazem v zkráceném zápisu. Oproti Javě byly posíleny a zjednodušeny možnosti použití lambd.

Existuje zde chytré přetypování, kdy za určitých okolností není třeba deklarovat přetypování a překladač i vývojové nástroje si dokáží typ odvodit samy. Ochrana před používáním null hodnoty je řešena tak, že uživatel musí explicitně uvést, že chce, aby ji daná proměnná mohla nabývat. Nechci a ani se zde nemohu věnovat detailnímu popisu všech funkcí, a proto pouze uvedu odkaz na dokumentaci jazyka [24].

3.3.2 Porovnání s Javou

Protože jsem měl dosud zkušenosti pouze s Javou, zajímalo mě srovnání obou jazyků, tím spíše, že autoři Kotlinu deklarovali, že jejich jazyk je lepší a odstraňuje část nedostatků Javy. Proto bych zde rád uvedl několik, případů ve kterých se oba jazyky významně liší.

První nedostatek adresovaný autory Kotlinu je snaha o odstranění nebezpečí null referencí. Jedná se o slabé místo mnoha programovacích jazyků, kdy pokus o práci s objektem, který odkazuje na null, končí výjimkou (v Javě nechvalně proslulá NPE - NullPointerException). V Kotlinu je tento problém řešen statickou kontrolou při překladu a tím, že v Kotlinu běžné datové typy a objekty implicitně nemohou odkazovat na null. Pokud uživatel potřebuje mít objekt odkazující na null, musí to explicitně definovat a kompilátor jej dále v kódu varuje a nutí vyřešit problémy, které by s tím mohly být spojené. Nedovolí objekt s null referencí propagovat dále. Pro kontrolu null zavádí Kotlin i několik operátorů. Je to operátor pro bezpečné volání - Safe operator - ?., který vrací hodnotu nebo null, ale nespadne s NPE. Dále

zavádí tzv. Elvis operator - ? :, který rozšiřuje Safe operator tím, že v případě null výsledku vrací jinou zvolenou hodnotu. Poslední operátor spojený s null referencí je tzv. !! operator, který umožňuje datový typ s možností null převést na datový typ bez možnosti null reference. Toto je také jeden z mála případů, kdy může v Kotlinu dojít k vyvolání NPE výjimky (další případy mohou nastat při propojení s Java funkcemi nebo při explicitním vyhození NPE výjimky).

Kotlin dále odstraňuje slabinu Javy v podobě možnosti použití tzv. raw typů. Ty se v Javě mohou vyskytnout, kdykoliv programátor použije generický typ bez zadání parametru typu. Pro příklad vytvoření objektu typu List, bez udání datového typu pro objekty v něm obsažené. Tato anomálie v Java vznikla snahou o udržení zpětné kompatibility při zavedení generických typů. Kotlin raw typy prostě nemá.

Další rozdíl je v polích, která jsou v Kotlinu invariantní, tedy neumožňují přiřazení pole s typem Array<String> do pole Array<Any> (Any je obdobou datového typu Object v Java). Tím se předchází chybám, které by překladač neodhalil a mohly by vzniknout až při běhu programu. Kotlin se také odlišuje tím, že má opravdové typy pro funkce, oproti Javě. Také explicitně definuje projekci typů v rámci generických parametrů a dědičnosti mezi nimi, nepoužívá checked exceptions, nemá primitivní datové typy (všechno v Kotlinu je objekt), ternární operátor, statické pole a další věci. Oproti Javě samozřejmě Kotlin přináší nové možnosti, které jsem vyjmenoval v počátku této podkapitoly a nebudu se jim v rámci této práce detailněji věnovat, protože bych se snažil více méně o přepis dokumentace jazyka, kterou lze nalézt online [24].

3.4 Gradle

S rozhodnutím vytvořit aplikaci na platformě Android s pomocí jazyk Kotlin souvisí i volba automatizace sestavení. Vzhledem k tomu, že jsem využil Android Studio jako vývojové prostředí, je přirozenou volbou Gradle, jako systém pro automatizaci sestavení. Android Studio jej využívá defaultně a protože nemám speciální požadavky na sestavení, nemám ani žádný důvod, proč jej rovněž nevyužít. Jako krátké představení uvedu, že Gradle vychází z konceptů použitých v Apache Ant a Apache Maven a ty dále rozšiřuje. Soubory popisující postup a způsob sestavení v Gradle jsou psány pomocí jazyka vycházejícího z Groovy a umožňují například sestavení více projektů, inkrementální sestavení a další.

3.5 Vyžadovaná funkcionalita

Dosud jsem se věnoval v této kapitole spíše jednotlivým prostředkům a základům, na kterých bude aplikace postavena, ale nepopsal jsem, co od ní očekávám a jak by měla fungovat. To bych rád shrnul nyní v podobě popisu funkcionalit, tak jak je bude moci uživatel použít.

3.5.1 Hlášení

Hlavní oblastí, kterou uživateli aplikace nabídne, je příprava vojenských hlášení. Uživatel by měl mít možnost zvolit si jedno ze standardizovaných hlášení, které chce připravit. Aplikace mu nabídne formulář pro dané hlášení, který bude obsahovat všechny jeho položky. Vyplní jej a následně dostane hlášení připravené v takové formě, aby je mohl přímo předat dál. Uživatel si nemusí pamatovat význam všech položek a případná kódová označení v hlášeních, protože je aplikace obsahuje a přirozeně ho vede v jejich vyplňování.

Často se v rámci hlášení udává pozice uživatele v souřadnicovém systému MGRS. Tento údaj bude mít uživatel předvyplněný aplikací, která jej získá pomocí GPS modulu v zařízení, na kterém běží. Aplikace automaticky zajistí převod ze zeměpisných souřadnic ve formátu zeměpisné délky a šířky do systému MGRS. Pokud uživatel potřebuje zadat jinou než svoji aktuální pozici, může tak učinit za pomoci mapové aplikace Google Maps, ve které si zvolí požadovanou pozici a po potvrzení ji aplikace převede do MGRS a vyplní do formuláře.

Podobně, jako jsou automaticky připraveny údaje o pozici, bude uživateli připravena informace o čase a datu. Tu aplikace automaticky získá ze systémového času telefonu, zakóduje do tvaru DTG a předvyplní do formuláře hlášení. Při zakódování bude také vycházet z údaje o časové zóně, který si uživatel v aplikaci nastaví.

Po sestavení hlášení bude mít uživatel možnost jej transformovat. První možností bude zakódovat čísla obsažená v hlášení pomocí RAMRODu. Pro zakódování se využije klíčové slovo, které si uživatel může nastavit. Druhou možností transformace bude převedení celého hlášení na vyhláskovaný tvar. Uživatel si bude moci v aplikaci zvolit preferovanou hláskovací abecedu.

3.5.2 Stavový panel

Aby měl uživatel přehled o čase a své pozici, bud aplikace obsahovat stavový panel. Na stavovém panelu budou uvedeny údaje o uživateli (volací znak, vysílací frekvence), o aktuální pozici (zde bude pozice uvedena v MGRS i ve standardním tvaru zeměpisné výšky a šířky), o přesnosti pozice a o tom, jak je tento údaj aktuální, a také datum a čas v běžném tvaru i ve formátu DTG.

3.5.3 Uživatelská nastavení

Aplikace umožní uživateli nastavit si údaje nutné pro tvorbu hlášení, které jsou po dobu jeho působení v terénu či na misi konstantní. Jsou to informace o sobě (volací znak a vysílací frekvence), kódovací slovo pro RAMROD, preferovaná časová zóna a hláskovací abeceda (výběr z NATO a české hláskovací abecedy). Dále si bude moci zvolit jazyk, ve kterém se bude aplikace zobrazovat (angličtina nebo čeština).

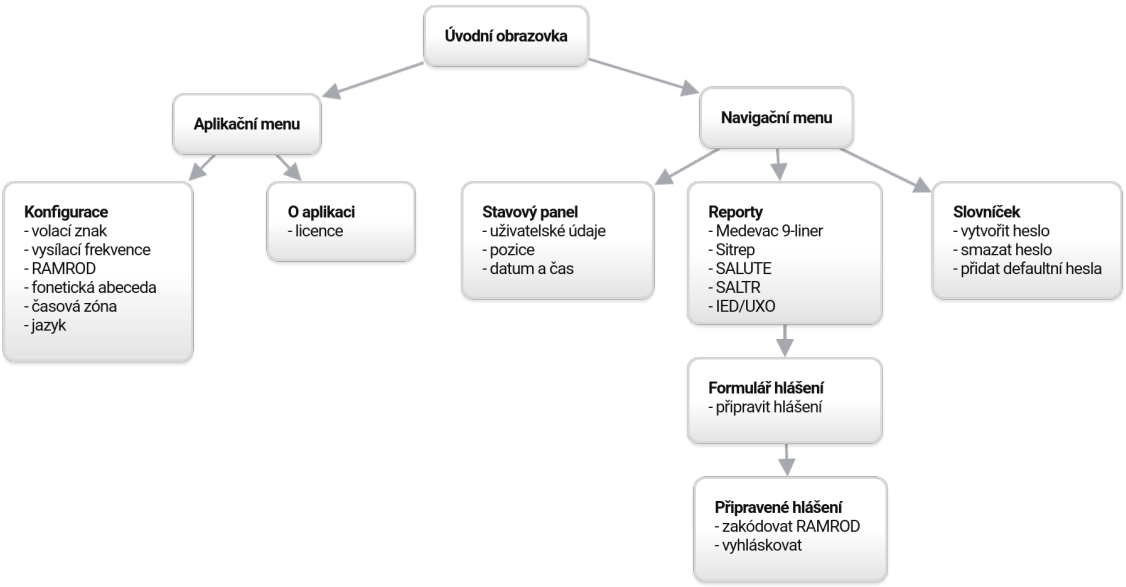
3.5.4 Slovníček

Doplňkovou funkcí aplikace bude slovníček akronymů a termínů používaných při sestavování hlášení nebo obecně ve vojenském prostředí. Ve slovníčku bude možné

vytvářet nová hesla a případně také hesla mazat. Bude sloužit uživateli k poznamenání nových akronymů či termínů, se kterými se při výcviku setká.

3.6 Návrh uživatelského prostředí

Návrh uživatelského prostředí mé aplikace využívá standardní prvky Android aplikací, které jsou definované aktuálně používaným Material designem. Mezi ně patří navigační menu aplikace, vysouvané od okraje obrazovky, aplikační menu v pravém horním rohu a plovoucí tlačítka - tzv. floating action button. Pro zvolení barev jsem využil nástroje navrhující barevnou paletu odpovídající Material designu.



Obrázek 3.2: Přehled obrazovek v aplikaci s možnými akcemi

V rámci aplikace se uživatel může pohybovat v rámci několika možných cest. Po spuštění aplikace se uživatel dostane na úvodní obrazovku, ze které může otevřít pomocí tlačítka v pravém horním rohu aplikační menu nebo pomocí tlačítka v levém horním rohu navigační menu. Aplikační menu umožňuje otevřít konfiguraci aplikace, ve které lze konfigurovat všechna uživatelská nastavení a nastavení aplikace, a nebo stránku s informacemi o aplikaci a licencích. V otevřeném navigačním menu je v horní části umístěný stavový panel, dále následují odkazy na jednotlivá hlášení a jako poslední položka je odkaz na slovníček. Po kliknutí na jednotlivé položky se uživateli sbalí navigační menu a otevřou se mu jednotlivá hlášení nebo slovníček. Ve slovníčku může procházet jednotlivá hesla, přidávat nová pomocí plovoucího tlačítka, mazat hesla z jejich kontextové nabídky a také obnovit všechna výchozí hesla. V rámci hlášení lze vyplnit formulář za pomocí různých vstupních prvků a následně jej pomocí plovoucího tlačítka nechat zpracovat. Tím se uživatel dostane na nový pohled, který je společný pro všechna hlášení. Zde je hlášení ve tvaru, ve kterém

se bude předávat dále, a také plovoucí tlačítka pro zakódování hlášení pomocí RAMRODu a pro jeho vyhláskování.

V rámci formuláře některých hlášení lze otevřít dialogové okno se vstupním prvkem pro zadání času a data. Také lze u některých hlášení zadat pozici z mapy pomocí akce vyvolané příslušným tlačítkem. V rámci slovníčku je při vytváření nového hesla používáno dialogové okno pro zadání hesla a jeho popisu.

3.7 Zabezpečení

Při návrhu aplikace jsem uvažoval i nad tím, zda by aplikace neměla být zabezpečená proti zneužití při případném použití v boji. V úvahu by připadal zaheslovaný přístup do aplikace a ukládání všech údajů, které aplikace obsahuje nebo vygeneruje v zašifrované podobě, případně také ověřování toho, že aplikace nebyla upravena třetí stranou. K těmto myšlenkám mě vedly případy, kdy chytré telefony s nainstalovanými aplikacemi mohly vést či vedly k vyzrazení informací a skutečností bojových operací a následně pak k ohrožení těchto operací i jejich uživatelů.

Prvním příkladem je náhodné odhalení vojenských objektů skrze data z aplikace zaznamenávající sportovní aktivitu uživatele. Podle článku [25] mohlo dojít k vyzrazení objektů, užívaných americkými bezpečnostními složkami či jinými spojeneckými armádami v Afghánistánu i jinde na světě kvůli grafice znázorňující trasy a místa, na kterých sportovali uživatelé aplikace Strava. Dalším příkladem může být údajné zničení velké části dělostřeleckých jednotek ukrajinských ozbrojených sil v bojích proti separatistům. Vojáci v těchto jednotkách si dle článku [26] prý zjednodušovali výpočty dat pro vedení palby pomocí aplikace pro smart phone, která byla upravená ruskými hackery a vyzrazovala jejich pozice. Posledním příkladem může být také sesbírání dat z ukořistěných přístrojů. Toho například využívá americká armáda v boji proti terorismu, jak popisuje tento článek [27].

Při zvážení těchto a jiných dalších případů vychází jako nejlepší možnost prostě smart phone v ostrém nasazení nepoužívat. Bylo by možné omezit a hlídat jeho používání a data, která jsou z něj odesílána nebo přijímána, ale rizika a množství hrozeb, které by bylo třeba hlídat, pravděpodobně převyšují výhody takto získané. Oproti tomu využití aplikace na smart phone v rámci výcviku AZ, kdy nedochází k práci s utajovanými informacemi, je možné a nikterak neohrožuje uživatele ani prováděný výcvik. Proto jsem se rozhodl omezit zaměření své aplikace pouze pro výcvikové účely a spolu s tím neřešit problém bezpečnosti.

4 Implementace a otestování v praxi

Předchozí části zprávy popsaly teoretické podklady, volbu platformy a obecný návrh aplikace. Tato kapitola obsahuje již konkrétní popis použitých prostředků a toho, jak je celá aplikace naprogramována. Nejprve se zde zaměřím na popis implementace uživatelského prostředí, včetně ukázek z aplikace. Dále popíšu, jak jsou implementované jednotlivé funkcionality a v poslední části uvedu zkušenosti a reakce uživatelů z testování aplikace na vojenském cvičení AZ.

4.1 Zobrazovací část

Při návrhu zobrazovací části jsem vycházel ze standardního stylu používaného v oficiálních aplikacích od Google ve stylu Material. Tomu jsem podřídil návrh rozložení menu, ovládacích prvků a všech ostatních komponent. Spojením mých představ o rozložení komponent a funkcionalit (viz 3.2) s Material stylem jsem vytvořil nejprve v ruce náčrt všech obrazovek aplikace a ten postupně převedl do aplikace.

4.1.1 Návrh pomocí XML

První implementovaná verze využívala standardní řešení používané při programování Android aplikací, kterým je kombinace popisu vizuálních prvků v xml souboru (viz příklad kódu 1) a obslužné třídy. Tyto dvě části se generují pro každou aktivitu v aplikaci. Př této práci mi dost pomohlo vývojové prostředí, které dokáže vygenerovat obě části pro běžná rozložení ovládacích prvků. Následně je možné je pomocí grafického prostředí doplnit o další prvky a tak poměrně snadno, bez složitého psaní kódu, navrhnout uživatelské prostředí. Tímto způsobem jsem vytvořil první funkční návrh, který obsahoval vše nezbytné.

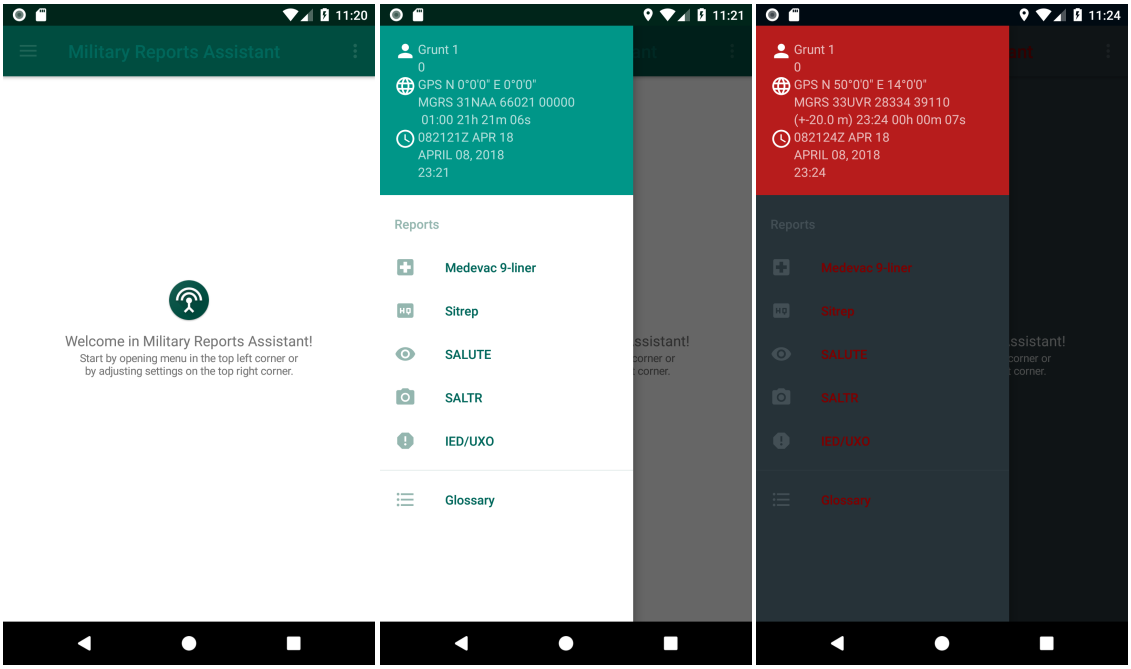
Pro hlavní aktivitu jsem použil rozložení s vysouvacím navigačním menu a s aplikačním menu v liště. Tato obrazovka šla navrhnout poměrně snadno, protože většina rozložení je již v defaultní verzi, kterou dokáže vytvořit vývojové prostředí. Musel jsem pouze doplnit a upřesnit položky jednotlivých menu a vytvořit návrh pro stavový panel v navigačním menu. Pro konfiguraci aplikace, odkazovanou z aplikačního menu, jsem využil standardní Android komponentu nazvanou PreferenceScreen s položkami pro všechny konfigurovatelné prvky. Výhodou je snadná konfigurace a vzhled, který je konzistentní se zbytkem systému a pro uživatele tak známý a přehledný.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/saltr_label_location"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/report_location" />

    <LinearLayout
        android:id="@+id/saltr_content_location"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <EditText
            android:id="@+id/saltr_value_location"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:hint="@string/report_location_hint"
            android:inputType="textPersonName" />

        <Button
            android:id="@+id/saltr_button_location"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/report_location_button" />
    </LinearLayout>
</LinearLayout>
```

Zdrojový kód 1: Ukázka komponenty definované pomocí XML

Zajímavější byl návrh aktivit pro jednotlivá hlášení, kde bylo potřeba vše rozmyslet s ohledem na pohodlí uživatele a intuitivní ovládání. Pro splnění mých představ bylo třeba použít poměrně složitou strukturu z horizontálních a vertikálních prvků (layout), do kterých byly na různá místa vloženy textové inputy, popisky a další prvky. Základním prvkem byl vertical layout, protože chci, aby se uživatel mohl hlášením pohodlně posouvat nahoru a dolů v průběhu vyplňování. V něm jsou umístěny další layouty, reprezentující jednotlivé položky hlášení, vždy s popiskem

a inputem (případně více prvky) a několika dalšími layouty, které pomáhají umístit prvky na správné místo. Na závěr jsem ještě přidal plovoucí akční tlačítko, které umožňuje přechod do aktivity se zpracovaným hlášením. Do aplikace jsem implementoval všech pět hlášení zmíněných v předchozích kapitolách a každé z nich má svou vlastní aktivitu.



Obrázek 4.1: Snímky obrazovky zobrazující hlavní stránku, navigační menu v denních barvách a navigační menu v nočních barvách

Aktivita pro zobrazení zpracovaného hlášení byla řádově jednodušší. Jejím hlavním prvkem je víceřádkový textový input, ve kterém je vypsáno připravené hlášení. Dále tato aktivita obsahuje plovoucí akční tlačítko pro zakódování textu hlášení pomocí RAMROD a druhé tlačítko pro jeho převod na vyhláskovanou verzi.

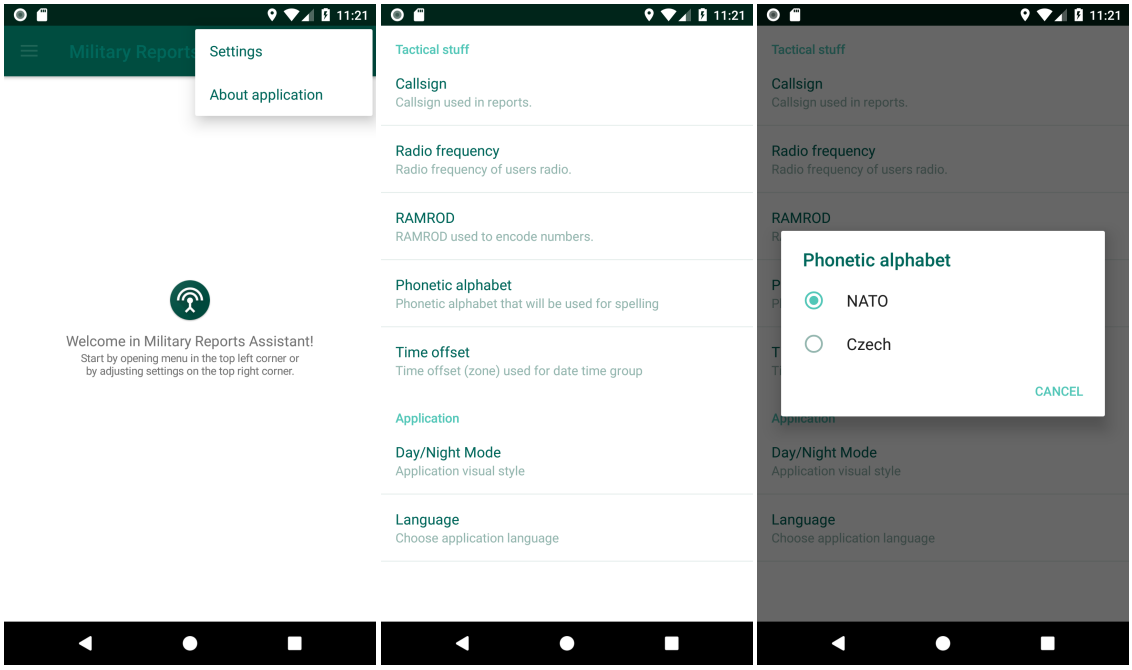
Slovníček je z pohledu uživatelského prostředí také spíše jednoduchý. Celou plochu aktivity vyplňuje list view, ve kterém se zobrazují jednotlivé položky slovníčku. Pro zobrazení položek jsem si připravil adaptér, který je zobrazuje ve formě vertikálního layoutu obsahujícího dva popisky (v prvním je název hesla a ve druhém je jeho vysvětlení).

V aplikace se ještě vyskytuje dialogové okno s licencemi a dialog pro zadání času a data, který popíšu v další části.

I když jde o poměrně jednoduchý a přímočarý proces, brzo jsem zjistil, že pro implementaci aktivit s jednotlivými hlášeními to není úplně praktické. Rozložení, které jsem navrhl, obsahovalo pro každou jednotlivou položku hlášení několik prvků a při ručním poskládání se xml soubor rychle rozrůstal. Přitom se často jednalo o opakující se prvky jen s pozměněným názvem.

4.1.2 Návrh pomocí Anko DSL

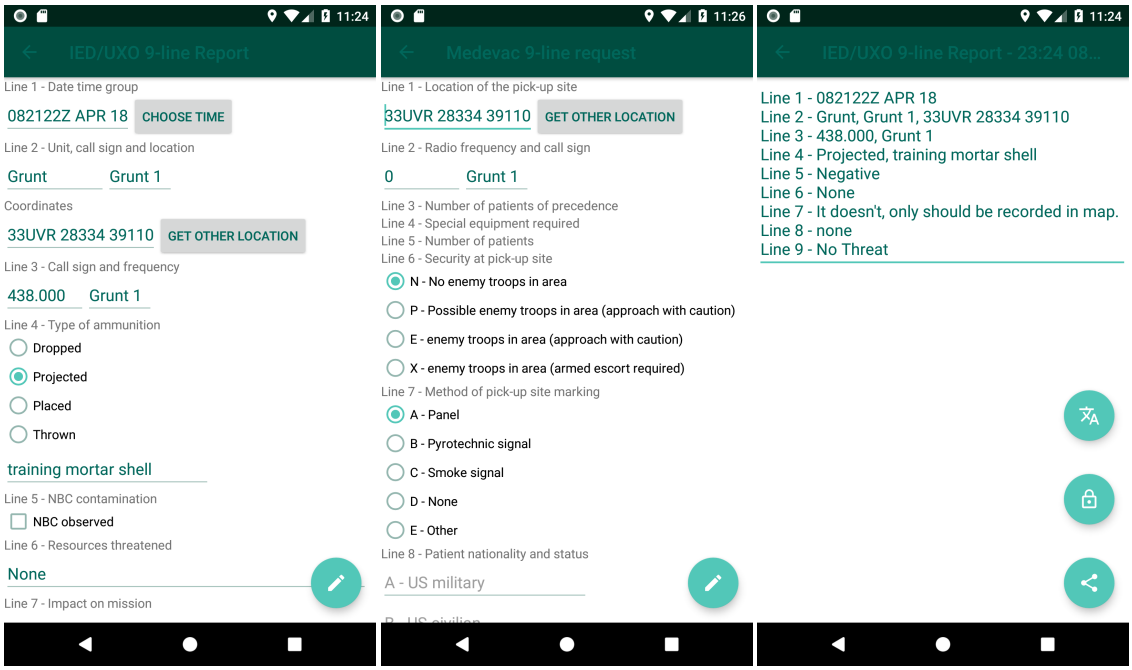
Při přípravě uživatelského rozhraní a vzhledu všech plánovaných aktivit jsem zjistil, že mnohokrát opakuji ty samé prvky, jen s drobnými obměnami. Spolu s tím mi vadilo, že celý návrh vzhledu je psaný pomocí xml. Jazyk xml se používá pro svoji popisnost a snadnou čitelnost, která s sebou nese velké množství nadbytečných informací. To se projevilo i v mém kódu, kdy každý další řádek formuláře zvyšoval neúměrně množství kódu. Proto jsem se rozhodl zkusit použít framework, na který jsem narazil při hledání informací k použití Kotlinu na Androidu. Jmenuje se Anko (ANdroid + KOtlin) a je vyvíjen společností JetBrains, která stojí i za Kotlinem. Část tohoto frameworku, která mě zaujala, se jmenuje Anko Layouts (více viz [28]) a dává programátorovi možnost, jak vytvořit layout aktivity pomocí kódu napsaného v Kotlinu. Ne, že by něco takového nebylo možné v Javě a bez tohoto frameworku, ale Anko má vlastní doménově specifický jazyk, pomocí kterého to výrazně usnadňuje a výsledek je dobře čitelný a udržovatelný.



Obrázek 4.2: Snímky obrazovky zobrazující aplikační menu a obrazovku s nastavením aplikace a změnu nastavení hláskovací abecedy

Další významnou výhodou tohoto řešení je zrychlení při inicializaci aktivit. Anko Layouts nemusí parsovat XML popis aktivity, ale vytváří ji přímo z kódu a tím odpadá právě pomalé parsování XML. Další výhodou je snadné vytvoření vlastních komponent. Důvod, kvůli kterému jsem se ho rozhodl vyzkoušet. Samozřejmě Anko Layout má i své nevýhody, například nemá úplně vyladěný náhled vytvářeného layout. Také neumožňuje jeho tvorbu sestavováním z komponent v grafickém režimu, jako je tomu u XML. Nicméně, po přepsání XML layoutů do kódu v Kotlinu, považuji tuto transformaci za lepší řešení. Pro často se opakující prvky layoutů jsem si vytvořil vlastní komponenty (typicky řádek hlášení s popiskem a textovým vstup-

ním polem), které dále používám, ušetřil jsem tím mnoho kódu a i řešení bez XML je čistší a přehlednější. Pro porovnání uvádím komponentu pomocí XML (viz kód 1) a komponenta pomocí Anko (viz kód 2). Rozdíl by byl ještě znatelnější při porovnání jednotlivých návrhů uživatelského prostředí, kde použití komponent v Anko šetří místo oproti XML, které opakuje ten samý kód. Bohužel se mi nepodařilo přepsat celé uživatelské prostředí do Kotlinu a nakonec tak v XML zůstalo navigační menu, aplikační menu a okno s konfigurací. Pokoušel jsem se je též přepsat, ale nepodařilo se mi najít způsob, jak je implementovat za pomocí Anko Layouts. Je možné, že bude v budoucnu autory doplněn.



Obrázek 4.3: Snímky obrazovky zobrazující přípravu hlášení, skryté řádky při přípravě hlášení a hlášení připravené k předání

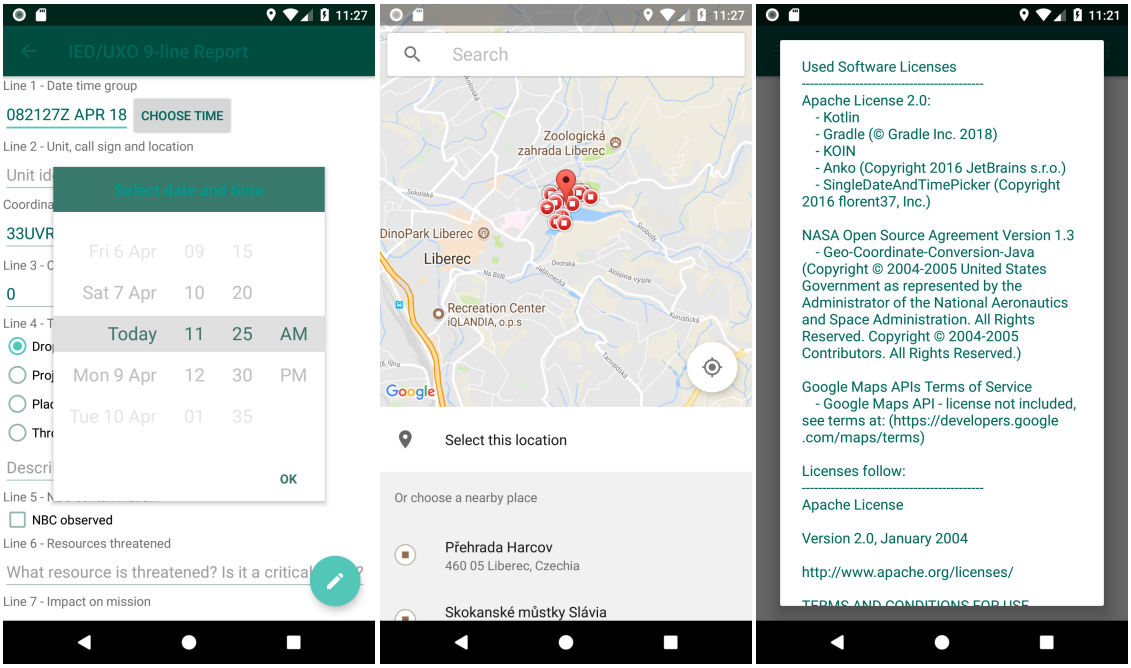
Nemá smysl zde popisovat detailně, jak aplikace vypadá. K tomu lépe poslouží okolní snímky obrazovky z aplikace samotné. Jedná se o snímky pořízené již ve verzi s použitím Anko Layouts. Raději bych zde zmínil ještě několik dalších implementovaných funkcí, které se týkají vzhledu. Oproti původnímu plánu se mi navíc podařilo implementovat změnu barev používaných aplikací v závislosti na tom, zda je den či noc. Noční barvy by neměly uživatele oslňovat a proto jsou zvolené tmavé barvy a odstíny červené, jak je vidět na snímku obrazovky. Také jsem díky komponentám vytvořeným pro části hlášení mohl snadno implementovat rozbalování a skrývání položek hlášení. Po klepnutí na název položky se celá skryje, pouze její název zůstává vidět. Podobně po klepnutí na název ve skrytém stavu se rozbalí a uživatel s ní může pracovat.

```
verticalLayout {
        textView(label) {
            textResource = label
            textSize = 20f
        }.setOnClickListener {
            if (hideableContent.visibility == View.VISIBLE) {
                hideableContent.visibility = View.GONE
            } else {
                hideableContent.visibility = View.VISIBLE
            }
        }

        linearLayout {
            hideableContent = this
            valueEdit = editText {
                hintResource = valueHint
            }
            locationButton = button(R.string.report_location_button)
            locationButton.setOnClickListener { getLocationFromGMaps() }
        }
        setPadding(0,0,0, 30)
}
```

Zdrojový kód 2: Komponenta uživatelského prostředí definovaná s pomocí frameworku Anko

Pro usnadnění zadávání času a data u některých hlášení jsem využil práci Florenta Chapignyho, který vytvořil funkční a pěkné dialogové okno pro jejich zadávání. Po klepnutí na tlačítko se otevře dialogové okno, ve kterém lze pomocí posouvání prstem nastavit měsíc, den, hodinu a minuty. Tyto údaje se poté vloží do příslušného textového pole. Před vložením je ještě kóduji do formátu DTG, ale o tom budu psát až v další části. Více k popisu této komponenty, viz její stránka na Githubu [29]

Podobně jsem zjednodušil získávání jiné než své aktuální pozice z mapy pomocí standardního PlacePickeru od Google [30]. Zde se po klepnutí na příslušné tlačítko otevře náhled s mapou, ve které uživatel posouváním vybere hledanou pozici a poté ji potvrdí. Aplikace od Place pickeru získá údaje v podobě zeměpisné délky a šířky a sama si je převede do systému MGRS. Oproti DateTime pickeru, který bylo snadné implementovat, jsem zde měl trochu problémy s nastavením. Google pro použití tohoto prvku vyžaduje zaregistrování aplikace ve své online vývojářské konzoli. Zde pak lze obdržet token, který je nezbytný pro zprovoznění samotného pickeru. Problematické v tomto řešení bylo dopátrat se vůbec, co je třeba kde zaregistrovat a jaký token použít. Po vložení správného tokenu do aplikace již vše fungovalo bez problémů.



Obrázek 4.4: Snímky obrazovky zobrazující DateTime picker, Place picker a dialogové info s informacemi o aplikace

Spolu s tím jsem ještě musel řešit uchování tokenu mimo verzovací systém. Při vývoji používám veřejný repozitář na svém účtu na Githubu a k tokenu, který by byl přímo v kódu aplikace, by se mohl dostat kdokoliv a začít jej využívat pro sebe. Celý problém jsem vyřešil definováním proměnné v config souboru Gradle na svém lokálním počítači. Při kompilaci aplikace je token do aplikace vložen a lze používat Place picker a přitom token zůstává mimo veřejnosti přístupný repozitář.

4.2 Popis služeb a výpočetní části

Podstatnou částí aplikace, která zůstává uživateli skryta, je servisní vrstva. Ta zajišťuje veškerou funkcionalitu a to, aby spolu vše spolupracovalo a fungovalo. V mé aplikaci tato vrstva zajišťuje několik různých oblastí, které jsem se od sebe rozhodl oddělit pro lepší čitelnost kódu, přehlednost a hlavně také pro to, aby bylo možné některé části použít na více místech.

Do této části aplikace spadají třídy určené pro obsluhu jednotlivých aktivit, které zajišťují to, že všechny prvky navržené v rámci uživatelského prostředí budou dělat to, co mají. Tlačítka budou spouštět zamýšlené funkce, vstupní prvky se budou chovat a formátovat podle návrhu atp. Zároveň propojuje jednotlivé aktivity mezi sebou a určuje, které akce vedou k vyvolání dalších aktivit, a zajišťuje dotažení potřebných servisních tříd.

Další větší oblast jsou servisní třídy, které zajišťují různé oblasti výpočtů, transformace dat a práce s prostředky jako je databáze či nastavení aplikace. V aplikaci mám servisní třídu pro práci s daty a časem, se souřadnicemi, s databází, dále třídu

zajišťující překlad aplikace, třídu pro kódování a také třídu pro práci s nastavením aplikace.

Kromě těchto dvou skupin je zde ještě několik nesourodých tříd, které slouží různému účelu. Samozřejmě je zde hlavní třída, která řídí, co se děje po startu aplikace. Také jsem vytvořil několik tříd obsahujících různé konstantní hodnoty (hláskovací abecedy, základní obsah slovníčku, různé formátovací znaky) využívané v aplikaci. Specifickými třídami je pomocná třída pro práci s databází, třída popisující vkládání závislostí a listener pro získávání údajů o pozici. Všechny tyto specifické třídy ještě popíšu dále.

4.2.1 Oddělení zobrazovací části od služeb

Už při prvním návrhu rozhraní aplikace jsem byl Androidem přirozeně směřován k rozdělení aplikace na více vrstev. Android standardně využívá pro popis uživatelského rozhraní xml a pro jejich ovládání třídy napsané v Javě. Nejjednodušší aplikace psaná tímto způsobem by tak mohla mít dvě vrstvy podle typu kódu - zobrazovací v xml a řídící v Javě. Moje rozhodnutí použít framework Anko mi umožnilo zbavit se téměř všech xml souborů a nahradit je třídami napsanými v Kotlinu. Přicházím tak o rozdělení podle jazyka, ale stejně jsem zachoval rozdělení na tyto dvě vrstvy a přidal jsem ještě servisní vrstvu.

Zobrazovací vrstvu jsem popsal v předchozí části a servisní bude popsána dále, nyní stručně zmíním, k čemu slouží vrstva obsluhující uživatelské rozhraní. Její funkce je velmi podobná při použití xml i Anko. V obou případech jsou v ní třídy pro každou jednotlivou aktivitu aplikace, které zajišťují vytvoření aktivity a určují, co se bude dít v návaznosti na povely uživatele. Vždy při spuštění aktivity musí být nejprve vytvořeno její rozhraní (view). Při použití xml se parsuje soubor a na jeho základě se vytváří rozhraní. V mojí aplikaci, při použití Anko, se pouze rozhraní vytváří podle toho, jak je vytvořené v třídě Kotlinu. Dále se zde navazují různé akce na jednotlivé prvky pomocí listenerů, rozhraní se plní daty a já zde také nechávám dotáhnout závislosti na jednotlivé servisní třídy.

Většina akcí, které v aktivitě mohou nastat, mají své vlastní metody, odpovídající typu aktivity (např. *onCreate*, *onBackPressed*, *onPropertyChanged* atp.). Tyto metody lze přepsat (override) a doplnit o vlastní požadované akce. Například v hlavní aktivitě využívám *onNavigationItemSelected* k otevření dalších aktivit (lze vidět v ukázce kódu 3), podle zvolené položky v navigačním menu v levém vysouvacím panelu. V této vrstvě jsem také vytvořil abstraktní třídu pro aktivy hlášení. Aktivity zastupující jednotlivá hlášení využívají stejné služby a všechny také potřebují funkci pro zpracování zadaných dat a jejich předání do aktivity zpracovaného hlášení, která je pro všechna hlášení stejná. Právě tyto společné části jsem vložil do abstraktní třídy, ze které dědí všechny aktivity hlášení. Také jsem zde vytvořil abstraktní funkci *getData*(), která je implementována ve všech hlášeních a slouží k získání dat z formuláře do hlášení.

```
override fun onNavigationItemSelected(item: MenuItem): Boolean {
    when (item.itemId) {
        R.id.nav_medevac -> {
            startActivity(Intent(this, MedevacReport::class.java))
            return true
        }

        ...

        R.id.nav_glossary -> {
            startActivity(Intent(this, Glossary::class.java))
            return true
        }
    }

    mainUI.drawerLayout.closeDrawer(GravityCompat.START)
    return true
}
```

Zdrojový kód 3: Ukázka části obslužné třídy, starající se o navigační menu

4.2.2 Závislosti pomocí KOIN

Servisní vrstva mé aplikace obsahuje třídy s funkcemi používanými na více místech. Je to logický krok k zpřehlednění kódu a k zamezení opakování stejných funkcí ve více třídách. Spolu s ním přichází nutnost propojit servisní třídy s třídami, ve kterých budou použité. Od začátku jsem plánoval využít dependency injection, abych nevytvářel těsné závislosti mezi třídami. Také jsem chtěl prozkoumat možnosti, které aktuálně Android ve spojení s Kotlinem nabízí. Kromě známých a více rozšířených frameworků, jako je Spring nebo Dagger2, které lze použít, jsem narazil také na KOIN. KOIN je framework pro dependency injection v Kotlinu a je to poměrně jednoduché a pragmatické řešení. Více se o něm lze dočíst v jeho repozitáři na Githubu [31].

```
val myModule: Module = applicationContext {
    bean { PreferencesServiceImpl(androidApplication().baseContext)
        as PreferencesService }
    bean { DateTimeServiceImpl(get()) as DateTimeService }
    bean { LocationServiceImpl(androidApplication().baseContext, get())
        as LocationService }
    bean { EncodingServiceImpl(get()) as EncodingService }
    bean { LocaleServiceImpl(get()) as LocaleService }
    bean { DatabaseServiceImpl(androidApplication().baseContext, get())
        as DatabaseService }
}
```

Zdrojový kód 4: Závislosti jednotlivých tříd definované v seznamu pro KOIN

V aplikaci jsem musel pro použití tohoto frameworku definovat v Gradle jeho závislost a po startu jej spustit spolu se seznamem jednotlivých bean, které má inicializovat. V tomto seznamu se vytváří jednotlivé třídy, které budou předávány jako závislosti, spolu s rozhraním, místo kterého mají být vloženy do dalších tříd. Uvádím jej jako ukázku z kódu 4. Pak už v rámci aplikace v jednotlivých třídách ovládajících rozhraní používám tyto třídy, spolu s informací o tom, že budou automaticky vloženy pomocí frameworku, jak lze vidět v kódu 5.

```
private val dateTimeService: DateTimeService by inject()
private val locationService: LocationService by inject()
private val preferencesService: PreferencesService by inject()
private val localeService: LocaleService by inject()
```

Zdrojový kód 5: Použití tříd zpravovaných frameworkem KOIN pro dependency injection

4.2.3 Lokalizace

Aplikaci jsem se rozhodl lokalizovat do anglické a české verze. Česká verze je nezbytná pro použití v AZ a v anglickém jazyce jsem celou aplikaci vyvíjel a pokud ji chci dát k dispozici veřejnosti, je to vlastně standard. Android Studio nabízí užitečný nástroj pro překlad, který umožňuje snadno doplnit překlad pro další jazyky. Po instalaci aplikace na cílové zařízení se automaticky zvolí jazyk podle toho, jaký používá zařízení. Pokud aplikace nenabízí překlad pro zvolený jazyk, použije se defaultní. Ve své aplikaci jsem jako defaultní jazyk zvolil angličtinu.

Jako doplňkovou funkcionalitu jsem chtěl uživateli nabídnout možnost si zvolit jazyk nezávisle na systémovém jazyku v očekávání, že to nebude nic složitého. Paradoxně se ukázalo, že Android není této možnosti nakloněn a oproti lokalizaci

podle systémového jazyka to byl nakonec poměrně netriviální úkol. Pro tuto funkcionalitu jsem si vytvořil položku v konfiguraci, kde si uživatel jazyk zvolí, a také službu, která obstará změnu jazyka. Obtížné zde bylo najít, co všechno je třeba v kontextu aplikace změnit. Po vyzkoušení několika různých variant se mi podařilo najít správnou, ale i tak bylo nutné některé části (názvy aktivit, navigační menu, ...) ještě překládat ručně v rámci kódu, protože Android si se změnou jazyk za běhu aplikace moc poradit neumí.

4.2.4 Získávání informací o pozici

Další důležitou službou v mé aplikaci je určování uživatelovy pozice a její převod do systému MGRS. Tato funkcionalita se opakuje ve většině hlášení a také ji využívám ve stavovém panelu v navigačním menu. Navíc se převod do MGRS používá při získávání pozice z PlacePickeru, ten totiž vrací pozici, stejně jako standardní systémová služba, ve formátu zeměpisné šířky a délky.

Pro získání informací o aktuální (nebo poslední známé) pozici uživatele využívám systémovou službu Androidu. Použití je poměrně jednoduché, jediné, co je třeba splnit, je získat od uživatele povolení k používání této služby. K tomu mám v aplikaci dialog, který se otevře po prvním spuštění a žádá o udělení práv. Pokud má aplikace tato práva, lze již snadno získat *LocationManager* z kontextu aplikace a s jeho pomocí si vyžádat získávání informací o pozici.

```
1  private fun getLocationFromGMaps() {
2      val api = GoogleApiAvailability.getInstance()
3      val code = api.isGooglePlayServicesAvailable(activity)
4      if (code == ConnectionResult.SUCCESS) {
5          val builder = PlacePicker.IntentBuilder()
6
7          val currentGPS = locationService.getCurrentGPS()
8          val bottomLeft = LatLng(currentGPS.latitude - CORNER_POINT_DISTANCE,
9              currentGPS.longitude - CORNER_POINT_DISTANCE)
10         val topRight = LatLng(currentGPS.latitude + CORNER_POINT_DISTANCE,
11             currentGPS.longitude + CORNER_POINT_DISTANCE)
12         builder.setLatLngBounds(LatLngBounds(bottomLeft, topRight))
13         startActivityForResult(activity, builder.build(activity),
14             PLACE_PICKER_REQUEST, null)
15     } else {
16         activity.alert(Appcompat,
17             c.getString(R.string.report_location_missing_api)).show()
18     }
19 }
```

Zdrojový kód 6: Použití PlacePickeru z Google Places API

Druhým zdrojem informací o pozici uživatele je již dříve zmíněný PlacePicker.

Využívám zde možnosti nabízené Google Places API. V rámci komponent pro layout, mám i komponentu pro zadání pozice, která při zobrazení obsahuje informaci o aktuální pozici a pomocí tlačítka lze vyvolat PlacePicker, který má jako počáteční místo nastavenou tuto pozici. Použití PlacePickeru je vidět ve zdrojovém kódu 6. Poté, co uživatel zvolí jinou pozici, je odeslána zpět do aplikace, automaticky překódována a vložena do cílového pole.

Nezbytnou funkcí mé služby je převod ze zeměpisné šířky a délky na MGRS souřadnice. Zvažoval jsem, zda si tento výpočet nenaprogramovat sám, ale je poměrně složitý, a proto jsem se rozhodl využít již existující řešení od Richard Claytona z jeho repozitáře na Githubu [32]. Tento kód využívá knihovnu projektu Solar Winds od NASA a moje aplikace tudíž musí splňovat podmínky jejich původní licence. Tyto podmínky nejsou nijak zvlášť přísné, ale pro jejich splnění je třeba, aby aplikace zmiňovala tuto licenci a její kód byl veřejně dostupný.

4.2.5 Překódování výsledné zprávy

Jedna z jednodušších služeb je kódovací služba. Je využívána v náhledu připraveného hlášení k jeho úpravám. S její pomocí lze v hlášení zakódovat číslice podle klíčového slova pro RAMROD, nechat si hlášení vyhláskovat nebo využít obě možnosti. Také obsahuje funkci pro vypsání hlášení ve zformátovaném tvaru tak, aby se uživateli snáze četlo. Tato funkce vezme data z hlášení a vypisuje všechny jeho položky, které byly zadané. U položek, které mají konkrétní název, jej vypisuje ještě před daty oddělený pomlčkou.

Kódování číslic pomocí RAMRODu je řešeno jednoduchou funkcí, která projde všechna zadaná data hlášení, znak po znaku a všechny číslice nahradí příslušným odpovídajícím písmenem z kódového slova. Poněkud složitější situace je u hláskování, které by mělo nahradit každé písmeno jeho odpovídajícím slovem. Opět je zde funkce, která postupuje znak po znaku a pokud se jedná o písmeno, nahradí jej. Pro anglickou/NATO hláskovací abecedu by tento přístup zcela stačil.

```
1  private fun isCzechLetterChFirstSymbol(alphabet: Int, index: Int, text: String) =
2      alphabet == CZECH_ALPHABET_VALUE && (index != text.length - 1)
3      && text[index].toLowerCase() == 'c' && text[index + 1].toLowerCase() == 'h'
4
5  private fun isCzechLetterChSecondSymbol(alphabet: Int, text: String, index: Int) =
6      alphabet == CZECH_ALPHABET_VALUE && (index != 0)
7      && text[index - 1].toLowerCase() == 'c' && text[index].toLowerCase() == 'h'
```

Zdrojový kód 7: Funkce používané při hláskování písmene ch

Pro českou hláskovací abecedu tento přístup nestačí. Problematické je zde písmeno ch, které se skládá ze dvou znaků a tedy již nelze jít pouze znak po znaku. V algoritmu jsem pro tento případ přidal podmínku, která když narazí na písmeno c, kontroluje následující písmeno. Pokud následuje písmeno h, vloží místo c slovo pro

ch a následující h již ignoruje. Podmínky pro kontrolu jsem obalil do funkcí, které lze vidět ve zdrojovém kódu 7. Pokud bych chtěl zavézt abecedy s dalšími podobnými, víceznakovými písmeny, musel bych zde ručně upravovat podmínky a kontrolu pro každý znak.

4.2.6 Uživatelská konfigurace

Pro získávání dat a informací u uživatelových nastavení aplikace (tzv. preferences) mám další, poměrně jednoduchou službu. Tato služba umožňuje pomocí systémových prostředků získávat data uložená do preferences aplikace. Jsou zde ukládána všechna data z konfigurace - nastavení jazyka aplikace, volba denního či nočního módu a poté uživatelská nastavení, mezi které patří volací znak, vysílací frekvence, klíčové slovo pro RAMROD, volba hláskovací abecedy a zvolené časové pásmo.

Možnost přepínat barevné schéma podle denního času jsem do aplikace oproti původnímu plánu přidal, protože se jedná o užitečnou vlastnost. Pro její implementaci stačilo přidat pouze správný soubor definující barvy pro noční variantu do správné složky. Jako další vylepšení jsem zvažoval automatické nastavení časového pásma, podle uživatelovy pozice.

Tuto možnost jsem zavrhl ze dvou důvodů. Jednak časové pásmo, používané v terénu, nemusí souviset s reálnou lokací, na které se uživatel nachází. Také by mohlo docházet k nedorozuměním při pohybu na rozhraní pásem a solidní problém by zde dělal i přechod mezi letním a zimním časem. Proto jsem volbu časového pásma ponechal na uživateli.

4.2.7 Sdílení zpráv pomocí intentu

Kromě jednotlivých služeb se v aplikaci vyskytuje ještě několik řešení, které stojí za to zmínit. První z nich je funkcionalita, kterou jsem původně neměl v plánu implementovat. Jedná se o možnost odeslat připravené hlášení pomocí SMS, mailu či jiných vhodných aplikací. Android nabízí opravdu jednoduché elegantní řešení, jak toto zpracovat, a proto jsem nakonec tuto možnost do aplikace doplnil. Android nabízí možnost odeslat text jako obsah intentu - způsobu, jak poslat či předat žádost jiným nainstalovaným aplikacím. Klasickým případem je třeba funkce „Sdílet“, která se v Androidu nabídne po označení textu. Je to přesně funkce, která se mi hodila pro sdílení obsahu hlášení. V kombinaci s frameworkem Anko, který má pro vytvoření tohoto intentu vlastní funkci (lze vidět v ukázce zdrojového kódu 8), se ve výsledku jedná o jeden řádek v kódu (nepočítám vytvoření tlačítka vyvolávajícího tuto funkci), který výrazně rozšiřuje možnosti aplikace.

```
reportPreviewUI.shareButton.setOnClickListener {
    share(this.title.toString() + Utilities.NEW_LINE
    + encodingService.formatReportText(report), this.title.toString())
}
```

Zdrojový kód 8: Využití intentu pro sdílení textu za pomocí frameworku Anko

4.2.8 Slovník v databázi

Z frameworku Anko jsem využil ještě jednu jeho další část a to soubor funkcí a metod pro jednodušší práci se SQLite databází v Androidu. Při práci na slovníku bylo třeba rozhodnout, jakým způsobem budou uchovávány všechny jeho položky. Protože jsem chtěl vyzkoušet i tuto možnost nabízenou Ankem, rozhodl jsem se pro databázi. Framework zde přináší zjednodušení v práci s databází. Není nutné složitě parsovat výsledky dotazů nebo řešit otevírání a zavírání spojení s databází. Všechny tyto věci za mně řeší framework, já jsem si musel pouze určit strukturu tabulky pomocí třídy s datovým modelem. Vše se mi podařilo zprovoznit podle mých představ překvapivě rychle a pohodlně.

4.3 Otestování v praxi

Součástí zadání této práce je i pokus o získání zpětné vazby z používání aplikace při výcviku AZ. Pro tuto část jsem využil cvičení AZ Hradba 2018. Jedná se o společné cvičení všech jednotek AZ, v rámci kterého by měly prokázat své schopnosti a nacvičené dovednosti. V rámci cvičení bylo třeba předávat i standardizovaná hlášení a tak teoreticky byl prostor pro otestování aplikace. Průběh a výsledky testování a také návrhy na možná rozšíření aplikace uvedu v této části zprávy.

4.3.1 Způsob testování aplikace

Pro testování aplikace jsem připravil dotazník s otázkami v aplikaci Google Forms. V úvodu dotazníku jsem uvedl odkaz na uživatelský manuál a odkaz ke stažení zkompilované aplikace. Tyto materiály jsem v průběhu týdne před cvičením Hradba 2018 předal k testování kolegům z jednotky AZ, ke které náležím. Nedával jsem jim žádné zvláštní pokyny k používání a testování a nechal jsem na jejich uvážení, zda a jak aplikaci využijí. Vzhledem k tomu jsem upravil i otázky v dotazníku.

4.3.2 Dotazník a souhrn odpovědí

Samotný dotazník jsem rozdělil na tři tematické okruhy. Nejprve jsem zkoumal využívání smartphonů při výcviku, dále využití samotné aplikace při výcviku a jako poslední okruh zhodnocení samotné aplikace a prostor pro návrhy pro doplnění. Abych co nejvíce usnadnil vyplňování dotazníku, využil jsem v maximální možné

míře otázky s předem stanovenými možnými odpověďmi, ze kterých uživatelé mohou vybírat. Jediná otevřená otázka je dotaz na možná vylepšení samotné aplikace. Cílem dotazníku bylo sesbírat reakce na aplikaci a případně zjistit, jak by bylo možné ji vylepšit, tak aby mohla lépe sloužit uživatelům.

Dotazník jsem rozeslal 68 příslušníkům roty AZ, ve které sloužím. Celkově se mi sešlo 8 vyplněných odpovědí, což odpovídá 11,7%. Celkový počet respondentů není nijak velký a i u nich dále záleželo na dobrovolnosti jestli aplikaci otestují a zda následně dotazník vyplní. Pro malý počet odpovědí nepovažuji odpovědi za zvláště významné, ale trend který ukazují se pokusím dále popsat.

První část dotazníku obsahovala dvě otázky: „Je vhodné používat smartphone při výcviku jako učební, navigační nebo jinou pomůcku?“ a „K jakým činnostem využíváte smartphone při výcviku?“. Otázky měli respondenta lehce uvést do problematiky a zároveň zjistit jeho postoj ke smartphonům. Nikdo z respondentů se nestavěl k využití smartphone zamítavě. Ostatně u druhé otázky se ukázalo poměrně pestré zastoupení všech možných odpovědí. Smartphone je jimi při výcviku využíván jako pomůcka při nácviku a navigaci i pro osobní účely, převážně ke komunikaci.

Zajímavější část dotazníku bylo použití aplikace při proběhlém cvičení. Zde je třeba brát odpovědi s rezervou, protože se nejednalo o nezávislé respondenty a evidentně se moji kolegové z AZ snažili mi pomoci nebo si usnadnit práci. Některé dotazníky se mi totiž vrátili vyplněné ještě před začátkem cvičení, na kterém jsem plánoval aplikaci otestovat. Otázky směřovali k tomu jak a kolikrát byla aplikace během cvičení použita a zda ji uživatelé plánují používat i na dalších cvičeních. Zde se projevil charakter samotného cvičení, které se zaměřovalo více na střežení a práci na kontrolně propouštěcím místě. Na spojovací přípravu a předávání hlášení nebyl samostatný výcvikový blok. Většina uživatelů aplikaci zkusila jednou, případně v jednotkách případů. Další nezanedbatelná část ji zvládla pouze otestovat soukromě, nad rámec cvičení a pouze jeden člověk ji použil vícekrát. Na druhou stranu všichni uživatelé plánují aplikaci využívat v budoucnu a pro většinu je přínosem.

Pro získání zpětné vazby k aplikaci sloužila poslední část dotazníku, kde jsem se dotazoval na používání, vzhled a ovládání aplikace. Na všechny respondenty působila aplikace přehledně a většinou neměli žádný problém se v ní zorientovat. Ovládání aplikace považují za intuitivní, i když by bylo možné některé věci zlepšit. Čtvrtina respondentů se setkala po instalaci aplikace s chybovou hláškou. Na tuto hlášku jsem též při svém testování občas narazil a zatím se mi nepodařilo zjistit její příčinu. Nicméně tuto chybu lze obejít restartem aplikace. Z funkcí nabízených aplikací byly jako nejužitečnější vybrány předně automaticky získávané a formátované údaje o pozici uživatele a o datu a čase. Další oblíbenou vlastností jsou formuláře hlášení, díky kterým si uživatel nemusí pamatovat formát hlášení. V otázce na možné zlepšení aplikace jsem dostal pouze jednu rozumnou odpověď, návrh na přidání další odpovědi do jednoho formuláře hlášení.

4.3.3 Reakce uživatelů

Možná cennější, než předchozí dotazník, pro mě byly reakce kolegů v AZ, se kterými jsem se o aplikaci mohl bavit. Od velitele roty jsem dostal souhlas s testováním aplikace a vyjádřil se nadšeně k využití aplikace při výcviku. Cennými reakcemi pro mě byla diskuze o aplikaci s jeho zástupcem, který má mnohaleté zkušenosti z AZ i z vojenské historie a komunikace o aplikace s velitelem mojí čety, který má obdobné zkušenosti. Díky jejich podnětům se mi podařilo odladit vzhled aplikace, formátování některých informací a upravit nepřesnosti ve formátech hlášení ještě před dokončením této zprávy. Některé nápady a návrhy na zlepšení jsem implementovat nestihl a uvedu je v následujícím části zprávy. Jako povzbudivé vnímám i další reakce členů jednotky, kteří na aplikaci reagovali pozitivně a nadšeně.

4.3.4 Návrhy pro další verze

Již v průběhu implementace mě napadlo několik možných rozšíření a vylepšení aplikace, které v následujících odstavcích nastíním. Mezi drobnější úpravy patří aktualizace souřadnic a času v textových polích v reálném čase. Aktuálně se data načtou při otevření aktivity a dále nejsou měněna. Jednoduchou úpravou by bylo také doplnění podrobnějších informací k jednotlivým prvkům hlášení. Bylo by možné pomocí dialogových oken nabídnout uživateli detailní informace o významu každé položky a nabídnout mu příklad, jak by dané pole mělo být vyplněno.

Trochu složitější úpravou by bylo vytvoření odkazu na obrazovce zamčeného telefonu. Uživatel by při zamčené obrazovce mohl tažením této ikony zahájit odemknutí telefonu a po odemknutí by se dostal přímo do spuštěné aplikace. Nejsem si jistý zda toto implementovat, případně jak složité by to bylo, ale v terénu by se rozhodně jednalo o užitečnou funkci.

Možným rozšířením slovníku by bylo centralizované úložiště jeho hesel, které by bylo na serveru. Zde by bylo spravované jedním zodpovědným člověkem, který by doplňoval nová hesla, a zbytek uživatelů (např. v rámci jedné jednotky) by si je vždy mohl aktualizovat ze serveru. Bylo by ovšem nutné domyslet, jak a která hesla synchronizovat a také možnosti správy v rámci serveru. Pravděpodobně by se zde s výhodou dala využít platforma Firebase od Google.

Mezi výraznější rozšíření bych zařadil možnost seznamu vlastních bodů zájmu. Ten by si uživatel vyplnil (nebo stáhnul ze serveru) před výcvikem v terénu a poté by je mohl z výhodou využívat jako referenční k určování pozice. Místo souřadnic MGRS by mohl nadřízenému stupni udávat svoji polohu jako relativní vůči určitému bodu, pomocí azimutu a vzdálenosti. Při reálném fungování by body byly označeny kódově a zvýšilo by to bezpečnost uživatele, proto by to bylo vhodné využít i při výcviku.

Další prvek, který by bylo možné zlepšit, je rozložení aktivit v aplikaci a přechod mezi nimi. Úvodní obrazovka je v aktuálním stavu nevyužitá (kromě obou menu) a bylo by možné nabídnout uživateli konfigurací zvolit její obsah (konkrétní hlášení či slovník). Podobně by bylo možné swypováním (rychlé tažení z jedné strany obrazovky na druhou) listovat mezi hlášeními, bez nutnosti otevírat menu. Konfi-

gurace by pak umožňovala nastavit pořadí či polohu jednotlivých aktivit vůči sobě. Tyto úpravy by pravděpodobně vyžadovaly velký zásah do aplikace. Možná by bylo snazší vybudovat v tomto případě aplikaci znovu, s využitím získaných zkušeností a některých částí původní aplikace.

Uživatelsky atraktivní vlastností by byla možnost si v aplikaci vytvářet další vlastní formuláře hlášení. Podobně jako předchozí úprava by pravděpodobně vyžadovala výrazný zásah do aplikace a bylo by třeba ji nejprve dobře navrhnout.

Toto jsou možné návrhy k rozšíření aplikace. V průběhu implementace jsem také několik původních nápadů zavrhnul. Na začátku práce jsem uvažoval i o použití aplikace v ostré akci, které by vyžadovalo aplikaci zabezpečit před zneužitím. Minimálně by bylo vhodné vstup do aplikace umožnit jen po zadání hesla. Také by bylo vhodné data, která si aplikace udržuje v telefonu, šifrovat. K problematice bezpečnosti se vyjadřuji již dříve, v předchozí kapitole.

Druhý, původně zvažovaný nápad, bylo nabídnout uživateli možnost ukládat si vlastní poznámky v aplikaci. Poznámky by mohly být mít formu textu nebo bodových seznamů. Při promýšlení tohoto nápadu mi došlo, že se snažím vymyslet vlastní variantu aplikace Google Keep a proto jsem od nápadu upustil.

Další část nápadů na rozšíření aplikace mi dodali zástupce velitel roty a velitel mojí čety. Přišli s návrhem udržovat informace o pozici ve formulářích stále aktualizované, případně uživatele upozornit na jejich neaktuálnost. Také jsem od nich dostal návrh umožnit uživateli si zvolit formát souřadnic zeměpisné šířky a délky ve stavovém panelu. Pro práci v terénu by se také hodil konvertor souřadnic mezi oběma souřadnicovými systémy a mezi jejich různými zápisy. Také navrhovali ukládat v aplikaci zpracovaná hlášení a u nezpracovaných ponechávat jejich data ve formuláři. Poslední návrh směřoval k zapracování úsporného režimu pro aplikaci.

5 Závěr

Při přemýšlení nad zadáním této práce jsem propojil svoje znalosti z informatiky a zkušenosti ze služby v Aktivních zálohách. Moje znalosti z informatiky mě přivedly k myšlence zjednodušit přípravu vojenských hlášeních, které běžně při výcviku AZ používám já i moji kolegové. Příprava hlášení je poměrně rutinní a pracná záležitost, která vyžaduje vyhledání a přípravu různých dat. Postup je zde vždy stejný, pouze vstupní data se mění, což je typické místo pro zapojení automatizace. Navíc část dat lze získat ze senzorů chytrých zařízení a tím odstranit nutnost jejich ručního vyhledání. Z těchto základních myšlenek jsem vycházel při návrhu řešení, ve kterém jsem spojil několik okruhů svých zájmů a prozkoumal další, pro mě nová témata.

Během návrhu jsem se pokusil prozkoumat dostupná řešení, vhodná pro použití při výcviku Aktivních záloh, ale zjistil jsem, že se jedná o natolik specifický problém, že neexistuje žádné řešení plně odpovídající kladeným požadavkům. Všechna z nalezených řešení splňovala vždy pouze část kritérii, se zaměřením na některou z oblastí, proto nebylo možné porovnat vlastní návrh s jinými podobnými.

Při vymýšlení řešení jsem zvolil jako cílovou platformu chytré telefony s operačním systémem Android a navrhl pro ně aplikaci, která zjednodušuje přípravu vojenských hlášení. Aplikaci jsem implementoval s použitím relativně nového programovacího jazyka Kotlin, který jsem se v průběhu práce mohl naučit. Obdobně jsem využil tuto práci i k seznámení s několika na něj navázanými frameworky, díky kterým jsem nemusel implementovat vše od úplných základů. Za zmínku zde stojí frameworky Anko a KOIN. Vzhledem k využití knihovny pro převod geografických souřadnic, která je pod licencí projektu NASA Solar winds, zpřístupňuji výsledný kód ve veřejném repozitáři, aby byl k užitku veřejnosti.

Vytvořenou aplikaci jsem předal kolegům z jednotky Aktivních záloh k otestování v průběhu vojenského cvičení Hradba 2018. Toto cvičení bylo zaměřené výrazně na jiné dovednosti a nebylo zde moc prostoru k procvičování spojovací přípravy u všech zúčastněných vojáků. Přesto jsem získal pozitivní zpětnou vazbu od svých kolegů i nadřízených a všichni dotázaní plánují aplikaci dále používat. Díky reakcím od kolegů se mi také podařilo upravit a zlepšit některé ovládací prvky a odstranit několik chyb.

Oproti počátečnímu návrhu, který jsem si vytvořil na základě rešerše, jsem zvládl také přidat několik užitečných funkcí. Do výsledné aplikace tak přibylo automatické nastavování barev podle toho, zda je noc či den, nebo možnost sdílet výsledné připravené hlášení standardními komunikačními kanály chytrého telefonu (e-mail, SMS, různé aplikace pro posílání zpráv). Naopak některé nápady jsem nerealizoval, protože se ukázaly jako nepraktické (zaheslování aplikace) nebo by se zbytečně snažily

nahradit jiné aplikace (vlastní uživatelské poznámky).

Podle zadání práce jsem měl uvést současný stav přípravy vojenských hlášení při výcviku AZ a zkusit nalézt jiná užívaná řešení. Dále pak podle výsledků této rešerše navrhnout vlastní řešení, implementovat je a, pokud to bude možné, otestovat je v praxi. Toto vše jsem postupně splnil, zpracoval a popsal v této písemné zprávě. Z výsledků testování vyplynulo několik možných dalších vylepšení a rozšíření aplikace, které bych rád do aplikace zapracoval spolu s mými dalšími nápady. Bohužel to z časových důvodů nebylo možné, protože vojenské cvičení, na kterém probíhalo testování aplikace, končilo dva týdny před odevzdáním této práce.

Vytvořením této aplikace se mi podařilo s využitím prostředků chytrých telefonů zjednodušit část výcviku vojáku v Aktivních zálohách. Spolu s tím jsem měl možnost si prohloubit znalosti vytváření mobilních aplikací pro Android a naučit se práci s jazykem Kotlin. S vývojem aplikace bych rád pokračoval dále nad aktuální verzi i přesto, že již nyní má všechny nezbytné funkce, protože vidím další možnosti jejího rozšíření.

Literatura

[1] *Příručka vojáka AČR*. 3. vydání. Vyškov: Velitelství výcviku Vojenská akademie Vyškov, 2009.

[2] *Statically typed programming language for modern multiplatform applications [online]*. Prague, Czech Republic: JetBrains, 2017 [cit. 2018-03-10]. Dostupné z: https://kotlinlang.org

[3] *Kotlin and Android [online]*. California, USA: Google, 2017 [cit. 2018-03-10]. Dostupné z: https://developer.android.com/kotlin/index.html

[4] *Time Zones (2012).gif*. In: Wikimedia Commons [online]. 2.5. 2007 [cit. 2018-03-21]. Dostupné z: https://commons.wikimedia.org/wiki/File:Time_Zones_(2012).gif

[5] *Utm-zones.jpg*. In: Wikimedia Commons [online], 26.1. 2007 [cit. 2018-03-21]. Dostupné z: https://commons.wikimedia.org/wiki/File:Utm-zones.jpg

[6] *Utmzonenugitterp.png*.In: Wikimedia Commons [online], 7.5. 2005 [cit. 2018-03-21]. Dostupné z: https://commons.wikimedia.org/wiki/File:Utmzonenugitterp.png

[7] *GridGPS [online]*. Legionowo Poland: GridCommand, 2017 [cit. 2018-04-02]. Dostupné z: https://play.google.com/store/apps/details?id=com.raveron.gridgps

[8] *Koord Konverter [online]*. Dominoc925, 2015 [cit. 2018-04-02]. Dostupné z: https://play.google.com/store/apps/details?id=com.dom925.convertor.koordkonverter

[9] *Land Nav Assistant [online]*. Rogers, Arizona, USA: Gammon Applications, 2017 [cit. 2018-04-02]. Dostupné z: https://play.google.com/store/apps/details?id=com.gammonapps.android.straightlinenavigator

[10] *Personal Eye System [online]*. Ljubljana, Slovinsko, Milsistemika, 2017 [cit. 2018-04-02].
Dostupné z: https://play.google.com/store/apps/details?id=si.milsistemika.pes

[11] *Military Acronym Reference Guide [online]*. Applied Information, 2018 [cit. 2018-04-02].
Dostupné z: https://play.google.com/store/apps/details?id=com.milacks.militaryacronym

[12] *Marine Corps Pocket Knowledge [online]*. Silverdale, Washington USA: Valor Applications LLC, 2016 [cit. 2018-04-02].
Dostupné z: https://play.google.com/store/apps/details?id=com.jlgabriel.pocketknowledge4

[13] *Army Leader Smart Cards [online]*. Winchester, Kentucky, USA, Polemics Applications, 2016 [cit. 2018-04-02].
Dostupné z: https://play.google.com/store/apps/details?id=com.polemics.armyleadersmartcard

[14] *Army Reports [online]*. Jolly Development, 2013 [cit. 2018-04-02].
Dostupné z: https://play.google.com/store/apps/details?id=appinventor.ai_alan_p_roberts.armyReports_0_2

[15] *9-Liner [online]*. Tallinn, Estonsko, OpApps OÜ, 2016 [cit. 2018-04-02].
Dostupné z: https://play.google.com/store/apps/details?id=com.opapps.nineliner

[16] *GPS Utility [online]*. DIMENSION S.r.l., 2015 [cit. 2018-04-03].
Dostupné z: https://itunes.apple.com/cz/app/gps-utility/id292815934?mt=8

[17] *Tactical NAV [online]*. AppDaddy Technologies, 2017 [cit. 2018-04-03].
Dostupné z: https://itunes.apple.com/us/app/tactical-nav/id412650650?mt=8

[18] *MilGPS [online]*. Cascode Labs Pty Ltd [cit. 2018-04-03].
Dostupné z: https://itunes.apple.com/us/app/milgps/id405835358?mt=8

[19] *Army Leader Smart Cards [online]*. Double Dog Studios LLC, 2017 [cit. 2018-04-03].
Dostupné z: https://itunes.apple.com/us/app/army-ranger-handbook/id378163411?mt=8

[20] *Military Terms & Acronyms [online]*. Army.ca Technologies, Inc., 2016 [cit. 2018-04-03].
Dostupné z: https://itunes.apple.com/us/app/military-terms-acronyms/id482153872?mt=8

[21] *Army Leader Smart Cards [online]*. Polemics Applications LLC [cit. 2018-04-03].
Dostupné z: https://itunes.apple.com/us/app/army-leader-smart-cards/id554523771?ls=1&mt=8

[22] *9 Line MEDEVAC Training App [online]*. [cit. 2018-04-03].
Dostupné z: http://appcrawlr.com/ios/9-line-medevac-training-app#authors-description

[23] *Android historical version distribution - vector.svg*. In: Wikimedia Commons [online]. 2018 [cit. 2018-04-04].
Dostupné z: https://en.wikipedia.org/wiki/Android_version_history#/media/File:Android_historical_version_distribution_-_vector.svg

[24] *Kotlin Language Documentation [online]*. Prague, Czech Republic: JetBrains, 2017 [cit. 2018-04-05].
Dostupné z: https://kotlinlang.org/docs/kotlin-docs.pdf

[25] *Fitness tracking app Strava gives away location of secret US army bases [online]*. The Guardian, 28.1. 2018 [cit. 2018-04-07].
Dostupné z: https://www.theguardian.com/world/2018/jan/28/fitness-tracking-app-gives-away-location-of-secret-us-army-basesa

[26] *Russian hackers tracked Ukrainian artillery units using Android implant: report [online]*. Washington, USA: Reuters, 22.12. 2016 [cit. 2018-04-07].
Dostupné z: https://www.reuters.com/article/us-cyber-ukraine/russian-hackers-tracked-ukrainian-artillery-units-using-android-implant-report-idUSKBN14B0CU

[27] *How the U.S. is using terrorists' smartphones and laptops to defeat them [online]*. USA Today, 31.1. 2018 [cit. 2018-04-07].
Dostupné z: https://www.usatoday.com/story/news/world/2018/01/31/smartphones-computers-terrorists-intelligence-agency-united-states/1079982001/

[28] *Anko [online]*. [cit. 2018-04-08].
Dostupné z: https://github.com/Kotlin/anko

[29] *SingleDateAndTimePicker [online]*. [cit. 2018-04-08].
Dostupné z: https://github.com/florent37/SingleDateAndTimePicker

[30] *Place Picker [online]*. [cit. 2018-04-08].
Dostupné z: https://developers.google.com/places/android-api/placepicker

[31] *KOIN - a concise and pragmatic dependency injection framework for Kotlin [online]*. [cit. 2018-04-14].
Dostupné z: https://github.com/Ekito/koin

[32] *Geo-Coordinate-Conversion-Java [online]*. [cit. 2018-04-15].
Dostupné z: https://github.com/Berico-Technologies/Geo-Coordinate-Conversion-Java

A Hláskovací abecedy

písmeno	NATO verze	česká verze
A	Alpha	Adam
B	Bravo	Božena
C	Charlie	Cyril
Č	-	Čeněk
D	Delta	David
Ď	-	Ďáblice
E	Echo	Emil
F	Foxtrot	František
G	Golf	Gustav
H	Hotel	Helena
CH	-	Chrudim
I	India	Ivan
J	Juliet	Josef
K	Kilo	Karel
L	Lima	Ludvík
M	Mike	Marie
N	November	Norbert (dříve Neruda)
Ň	-	Nina
O	Oscar	Oto (dříve Otakar)
P	Papa	Petr
Q	Quebec	Quido
R	Romeo	Rudolf
Ř	-	Řehoř
S	Sierra	Svatopluk
Š	-	Šimon (dříve Šárka)
T	Tango	Tomáš
Ť	-	Těšnov
U	Uniform	Urban
V	Victor	Václav
W	Whisky	Dvojité v (dříve William)
X	X-ray	Xaver
Y	Yankee	Ypsilon
Z	Zulu	Zuzana
Ž	-	Žofie

B Obsah přiloženého CD

- text diplomové práce
  - DP.pdf
- manuál aplikace
  - manual.pdf
- odpovědi na dotazník
  - dotaznik_odpovedi.xlsx
- archiv se zdrojovým kódem aplikace a obsahem repozitáře
  - Military-Reports-Assistant-master.zip