TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Mobilní aplikace pro RFID sportovní časomíru

Diplomová práce

Studijní program:	N2612 Elektrotechnika a informatika
Studijní obor:	Informační technologie
Autor práce:	**Bc. Tomáš Košek**
Vedoucí práce:	Ing. Jan Kolaja, Ph.D. Ústav nových technologií a aplikované informatiky

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Mobilní aplikace pro RFID sportovní časomíru

Jméno a příjmení: **Bc. Tomáš Košek**
Osobní číslo: M18000143
Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2020/2021**

Zásady pro vypracování:

Vytvořte mobilní aplikaci pro OS Android, která bude komunikovat s ručním UHF RFID readerem, webovým serverem časomíry a dalšími aplikacemi v zařízení. Během práce postupujte podle následujících kroků:

1. Seznamte se s RFID UHF technologií, s konkrétním readerem a dle manuálu nastudujte způsob použití knihoven výrobce.
2. Vytvořte aplikaci, která bude komunikovat přes bluetooth rozhraní s UHF RFID readerem pomocí knihoven výrobce. V rámci aplikace bude možné nastavit parametry RFID snímání, dle dispozic zařízení.
3. Aplikace dále bude komunikovat s webovým serverem pomocí web services. Server na základě identifikace readeru pošle aplikaci informace potřebné k jejímu nastavení (závod, časy startů, startovní listina, režim kontroly).
4. Aplikace nechť si veškerá data ukládá lokálně (i když je možné je okamžitě odeslat na server) pro případné opětovné zpracování.
5. Aplikaci a RFID reader otestujte v reálném provozu a odlaďte.
6. Kód aplikace řádně komentujte a na základě testování sestavte uživatelský manuál.

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

Seznam odborné literatury:

[1] KHATTAB, Ahmed, Esmaeil AMINI, Magdy BAYOUMI a Zahra JEDDI. *RFID Security*. Springer, 2017. Analog Circuits and Signal Processing. ISBN 978-3-319-47544-8.
[2] LACKO, Ľuboslav. *Mistrovství Android*. Přeložil Martin HERODEK. Brno: Computer Press, 2017. ISBN 978-80-251-4875-4.
[3] ZHOU, Mark, ed. *Advances in sport science and computer science*. Southampton: WIT Press, [2014]. WIT Transactions on information and communications technologies, volume 57. ISBN 978-1-84564-917-3.

Vedoucí práce:	Ing. Jan Kolaja, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	19. října 2020
Předpokládaný termín odevzdání:	17. května 2021

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2020

Prohlášení

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

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

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

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

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

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

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

7. ledna 2021 Bc. Tomáš Košek

Poděkování

Touto cestou bych rád poděkoval všem, kteří se v této nelehké době jakoukoliv měrou podíleli na vzniku mé diplomové práce. Svému vedoucímu Ing. Janu Kolajovi, Ph.D., který se při online konzultacích obratně zvládal věnovat všem svým dětem a zároveň poskytovat cenné rady k mé práci. Své rodině, přátelům a samozřejmě i té, která si toho se mnou prožila nejvíc.

Abstrakt

Diplomová práce shrnuje technologii RFID a jednotlivé způsoby měření závodů s jejím použitím. Popisuje tvorbu mobilní aplikace pro operační systém Android psané v programovacím jazyce Java. Jedná se o rozšíření již existujícího systému časomíry Sportchallenge o možnost měřit závodníky na kontrolních stanovištích. Jejím hlavním úkolem je získat načtené tagy z mobilní UHF RFID čtečky, se kterou komunikuje za pomocí Bluetooth technologie, a ty následně zpracovat a odeslat na webový server. Aplikace dále přehledně zobrazuje uživateli informace o závodě, startovní listinu či různé varovné notifikace související s chybným skenováním či odesíláním hodnot štítků. Práce v poslední části dokumentuje testování vytvořené aplikace a čtečky TSL 1128, jehož výsledkem je závěr, že jsou vhodné pro identifikaci sportovců na závodech pouze za určitých podmínek. Příloha také obsahuje uživatelský manuál k aplikaci.

Klíčová slova:

RFID, sportovní časomíra, Android aplikace, TSL 1128, ruční UHF RFID čtečka

Abstract

Master thesis summarizes RFID technology and different ways of race measuring with its use. It describes the creation of mobile application for the Android operating system written in the Java programming language. It’s an extension to the already existing Sportchallenge chip timing system in order to measure competitors at checkpoints. Main goal is to obtain scanned tags from handheld UHF RFID reader, with which it communicates using Bluetooth technology, and then process and send them to the web server. The application also clearly displays information about the race, starting list or various warning notifications related to incorrect scanning or sending tags values. In the last part, thesis documents testing of the created application and the TSL 1128 reader, the result of which is the conclusion that they are suitable for athlete’s identification in competitions only under certain circumstances. The appendix also contains a user manual for the application.

Key words:

RFID, chip timing, Android application, TSL 1128, handheld UHF RFID reader

Obsah

Seznam obrázků

Seznam zdrojových kódů

Seznam grafů

Seznam tabulek

Seznam zkratek

BLE	Bluetooth Low Energy – Bluetooth navržené pro nízkou spotřebu energie
BNC	Bayonet Neill Concelman connector – rychloupínací konektory pro antény
CRUD	Create, Read, Update, Delete – základní operace pro práci s daty
DNF	Did Not Finish – označení pro závodníka, který byl diskvalifikován
EDGE	Enhanced Data rates for GSM Evolution – mobilní internet druhé generace
GPS	Global Positioning System – globální družicový polohový systém
HF	High Frequency – vysoká frekvence – 13,56 MHz
HTTPS	Hypertext Transfer Protocol Secure – zabezpečený komunikační protokol
ID	Identifikace osoby, identifikátor zařízení
IDE	Integrated Development Environment – software pro vývoj programu
IP54	Ingress Protection Code 54 – ukazatel odolnosti elektrických zařízení
JSON	JavaScript Object Notation – datový formát pro přenos dat
LF	Low Frequency – nízká frekvence – 125 nebo 134 kHz
MAC	Media Access Control – fyzická unikátní adresa síťového zařízení
NFC	Near Field Communication – RFID rozšířené o vzájemnou komunikaci
PoE	Power over Ethernet – napájení zařízení po datovém síťovém kabelu
RAM	Random Access Memory – operační paměť zařízení
RFID	Radio Frequency Identification – identifikace pomocí rádiové komunikace
RSSI	Received Signal Strength Indication – ukazatel síly signálu
SDK	Software Development Kit – sada vývojářských nástrojů pro vývoj softwaru
SIM	Subscriber Identity Module – identifikační karta k mobilní síti
SQL	Structured Query Language – strukturovaný dotazovací jazyk
TSL	Technology Solutions – společnost zabývající se mobilními RFID čtečkami
UHF	Ultra High Frequency – ultra vysoká frekvence – 868 MHz pro Evropu
URL	Uniform Resource Locator – jednoznačná adresa zdroje
USB	Universal Serial Bus – univerzální sériová sběrnice pro komunikaci
XML	Extensible Markup Language – rozšiřitelný značkovací jazyk

Úvod

Technologie RFID je součástí měření běžeckých závodů již od roku 1993 a stále se v této oblasti jedná o volbu číslo jedna. Oproti pořádání závodů za pomocí tužky, papíru a stopek je potřeba pouze zlomek dobrovolníků a odměnou jsou přesnější cílové časy a mnohem snazší správa účastníků. RFID systémy pro měření závodů se stále zdokonalují a rozvíjejí, a i proto bylo vybráno pro závěrečnou práci toho téma.

Cílem diplomové práce je naprogramovat mobilní aplikaci pro operační systém Android komunikující s RFID čtečkou od společnosti Technology Solutions a s již zaběhnutým systémem od subjektu Sportchallenge. Aplikace by měla sloužit k identifikaci sportovců na kontrolním stanovišti závodu. Jejím hlavním úkolem je získat naskenované štítky z mobilní čtečky a ty odeslat na server Sportchallenge. Dalším bodem práce je tuto aplikaci společně s mobilní čtečkou 1128 UHF RFID otestovat při skutečném závodě a vyladit nedostatky. V poslední řadě k softwaru sestavit uživatelský manuál.

Výstupem práce jsou kromě naprogramované aplikace také doporučení pro práci se čtečkou, tedy popis jakým způsobem zvládá skenování většího množství závodníků, případně na jaké vzdálenosti a zdali je možné tento model pro skenování použít. Na základě hodnot udávaných výrobcem, je předpokládaným výsledkem dostatečný výkon čtecího zařízení (pro pokrytí probíhajících závodníků kontrolním stanovištěm) a obstojná výdrž baterie, která by měla vydržet po celou dobu většiny závodů.

Diplomová práce je rozdělena do pěti hlavních kapitol. Zprvu popisuje technologii RFID, její princip, jaké existují čtečky a štítky a příklady použití mimo závodní sféru. V druhé kapitole demonstruje řešení postavené přímo na míru potřebám organizace závodů, jak je finančně náročné takový systém sestavit z levnějších komponent, nebo jaké jsou alternativy k RFID. Další část představuje testovanou mobilní čtečku TSL 1128. Čtvrtá kapitola popisuje vývoj mobilní aplikace a v ní použité technologie a poté komunikaci se serverem Sportchallenge. V poslední části práce bylo provedeno testování aplikace a čtečky v praxi.

1 Radio Frequency Identification – RFID

Bezdrátová technologie RFID ke své funkčnosti vyžaduje speciální čtečku, která generuje elektromagnetické pole, a identifikační štítky (také nazývané tagy), které v blízkosti tohoto pole na své antény indukují napětí. Toto napětí následně slouží jako napájení pro mikročip umístěný ve štítku. Obvod čtečce odpoví vysláním svého identifikačního čísla a také obsahem své datové paměti.

RFID bývá mylně označováno jako technologie, která nahradí čárové kódy. Ve skutečnosti se spíše jedná o doplnění nedostatků a nevýhod, které používání čárových kódů přináší. Ty nemusí zvládat poskytnout svoji hodnotu po celou dobu výrobního procesu, který může obnášet například lakování nebo mytí výrobku. RFID nevyžaduje přímou viditelnost, a navíc ke štítkům mohou být připisovány doplňující informace [7]. Nicméně čárové kódy mají mnohonásobně levnější čtecí zařízení, k vytištění štítku není potřeba speciální tiskárna a v mnoha oblastech budou stále plně dostačujícím způsobem identifikace.

1.1 Dělení RFID podle frekvence

Z hlediska frekvencí můžeme RFID komunikaci rozdělit na čtyři typy:

- Nízká frekvence (LF – low frequency) – 125 nebo 134 kHz,
- Vysoká frekvence (HF – high frequency) – 13,56 MHz,
- Ultra vysoká frekvence (UHF – ultra high frequency) – pro Evropu 868 MHz,
- Super vysoká frekvence (SHF – super high frequency) někdy také označovány jako mikrovlny (MW – microwave) – 2,4 GHz.

Čím vyšší frekvence je použita, tím větší je dosah komunikace (až stovky metrů), ale zároveň dochází k horší propustnosti signálu skrze překážky nebo k většímu rušení způsobenému kovy v okolí.

1.1.1 Nízká frekvence

Nízké frekvence mají čtecí vzdálenost v jednotkách centimetrů, avšak nepodléhají rušení kovy. Používají se například pro identifikaci osob (docházkové systémy, jídelny), či evidenci zvířat (díky dobré propustnosti skrze tekutiny). Dalším příkladem jsou automobilový průmysl, kde Škoda Auto začala testovat RFID technologie ve své

zrekonstruované laboratoři v Kvasinách na testování voděodolnosti automobilů [8], nebo průmysl potravinářský, kde se identifikují zálohované pivní sudy [9].

1.1.2 Vysoká frekvence a NFC

Velice podobně jako nižší frekvence, z hlediska použití, na tom je frekvence vysoká. Ta se akorát vyjímá svojí speciální podkategorií typu přenosu dat nazvanou NFC. Díky bezkontaktním platbám a faktu, že skoro ke každému obchodu může zákazník disponovat svojí věrnostní kartičkou, se tato technologie stala dnes již běžnou výbavou mobilních telefonů. NFC pracuje na frekvenci stejné jako kategorie HF. Zde je dokonce výhodou, že se jedná o komunikaci pouze na vzdálenosti jednotek až desítek centimetrů. Pro zákazníky by bylo bezpochyby nežádoucí zaplatit nákup na sousední pokladně. NFC zařízení mají také tu vlastnost, že mohou sloužit jako čtecí zařízení nebo jako štítek. To znamená, že mobilní telefon podporující tuto technologii zvládne nejenom naskenovat NFC štítek, ale i třeba sloužit jako platební karta.

1.1.3 Ultra vysoká frekvence

UHF RFID nevyžaduje ke svému provozu přímou viditelnost mezi čtečkou a štítkem, přesto dokáže komunikovat na vzdálenosti jednotek metrů. To je nesmírnou výhodou v mnoha oblastech. Jednou z nich je bezesporu provádění inventur, kdy není třeba u každého produktu hledat čárový kód, ale stačí se po skladě, obchodě či kanceláři se čtečkou doslova jen projít. Opačným způsobem lze technologii využít při hledání konkrétního výrobku. V této situaci stačí pouze čtečce sdělit identifikátor hledaného předmětu (nebo provést skenování a potřebný štítek vybrat) a sledovat sílu signálu.

Tato technologie se velice rychle rozrostla skrze nejrůznější odvětví průmyslu. Ať už se jedná o přepravu zboží, jeho skladování nebo sledování produktu během výrobního procesu. Dokonce i při prodeji. Ať už přímo na pokladnách, jako to má například firma Decathlon, která využívá speciální koše, do nichž zákazník vloží svůj nákup a ihned je zřejmé, kolik položek a za jakou částku nákup obsahuje. Nebo u vstupů do obchodních prostor, kde je tato technologie používána pro kontrolu zákazníků, zdali se nepokouší odnést zboží, které nezaplatili.

Příjem zboží od dodavatele – tedy kontrola podle dodacího listu, zdali se na paletě nachází vše, jak má – je operace, která především v případě vysokého počtu položek může zabrat hodiny. Ideálním scénářem by bylo, kdyby již při příjezdu dopravce do areálu firmy byl celý nákladní vůz naskenován a vyhodnocen, zdali je zboží na paletách

umístěných uvnitř kompletní. Bohužel takovýto scénář nikdy nefunguje se stoprocentní spolehlivostí [7]. Nejenom, že je obtížné naskenovat všechny položky skrze kovové konstrukce přívěsů kamiónů či dokonce kontejnerů, ale jelikož antény takové čtecí brány by musely mít vysoký výkon, mohlo by dojít i k mylnému naskenování štítku mimo nákladní vůz. Ne vždy je tedy možné se na tuto technologii spolehnout. Je nezbytné brát v potaz, že UHF RFID technologie nemá žádné ověřovací mechanismy, které by dohlížely na to, zda došlo k úspěšnému doručení informace.

1.1.4 Super vysoká frekvence

RFID štítky se dělí na pasivní a aktivní. Pasivní štítky vynikají svojí prakticky neomezenou životností a velmi nízkou cenou. Aktivní obsahují baterii, díky které mají mnohonásobně větší dosah komunikace, avšak také mnohonásobně vyšší cenovku. V super vysokém frekvenčním pásmu se již předpokládá nasazení aktivních štítků. Používají se například u motoristických závodů, kde je očekáváno skenování tagů při vysokých rychlostech.

1.2 UHF RFID čtečky

Každé pásmo vyžaduje konkrétní konstrukci čtečky a štítku – nelze je mezi sebou kombinovat. UHF RFID čtečky se vyrábějí především ve dvou základních provedeních – mobilní a stacionární. Mobilní jsou díky svému tvaru a přenositelnosti používané pro provádění inventur, příjem zboží, vyhledávání ztracených produktů, nebo jako pomocník pro asistenty prodeje k zjištění doplňujících informacích o zboží. Zákazníky může zajímat aktuální skladový stav nebo jeho další dostupné varianty či velikosti. Zatímco stacionární čtečky jsou stvořené pro pevnou montáž, například na výrobní lince nebo na letištích pro sledování zavazadel.

1.2.1 Mobilní a stacionární čtečky

Stacionární čtečky vynikají velkou škálou konektorů, které na nich lze naleznout. I celé strany zařízení bývají vyhrazené pro připojení antén. Jejich velké množství (běžně až čtyři) lze ještě doplnit o rozbočovače. USB porty slouží buďto pro komunikaci s počítačem nebo pro připojení flash disku, na který lze nahrávat informace o načtených štítcích (někdy je přítomen i slot na MicroSD karty). Komunikovat je dále umožněno přes Ethernet (s možností napájení pomocí PoE) a rozhraní RS-232.

Obrázek 1: Stacionární čtečka Impinj R420 Speedway Revolution [10]

Oproti tomu čtečky mobilní se vyznačují svým vzhledem nejčastěji ve tvaru pistole s dobrou ergonomií, nízkou hmotností a integrovanou baterií a anténou. Dále mají již zabudované dotykové zařízení (nejčastěji s operačním systémem Android) nebo výrobci nabízejí pouzdra pro přidělání mobilního telefonu. Na to navazuje použití komunikačních technologií, které jsou především bezdrátové. Příklad mobilní čtečky je vidět na obrázku 2.

Obrázek 2: 1128 Bluetooth UHF RFID Reader [3]

2 RFID pro měření závodů

Poprvé bylo RFID použito pro sportovní účely při měření motoristického závodu v roce 1980 [12]. O první běžecký závod se postarala skupina holandských studentů a jejich firma Championchip v roce 1993. Za vděk tomuto systému se výrazně zkrátila doba potřebná ke zpracování výsledků a již nebylo potřeba tolika dobrovolníků. Podstatou změnu to znamenalo pro startování závodů, kdy se již závodníci nemuseli tlačit na startovní čáře, protože každému z nich se začal čas počítat až s průběhem startovní brány [13].

Při závodech se používají především stacionární čtečky. Mobilní čtečky, jak už jejich název napovídá, nejsou myšleny pro pevnou montáž. To pro měření přesného času v závodě znamená, že není jasné, kde se čtečka nachází například vzhledem k umístění cílové čáry. Pro tento případ jsou vhodné stacionární čtečky. Buďto s připojenými anténami nasměrovanými na cílovou čáru nebo s napojenou rohoží obsahující zabudované antény, která cílovou čáru kopíruje. Avšak ani tato řešení neposkytnou natolik přesné výsledky, aby se profesionálně pořádané závody obešly bez cílové kamery. Navíc v současnosti neexistuje žádný RFID způsob měření závodů, který by byl schválený mezinárodní atletickou federací. Tedy nelze například s jeho použitím stanovit světový rekord [26].

2.1 Stavba vlastního měřícího systému založeného na RFID

Žádný systém postavený na RFID technologii se neobejde bez čtečky. Kvalitní stacionární čtečky pro pasivní štítky se pohybují v cenovém rozmezí 20 000 až 50 000 korun. Není vhodné, aby se cílová čára skládala z jedné antény, takže je důležité, aby čtečka měla ideálně alespoň čtyři anténní konektory.

Existuje několik způsobů, jak zachytit RFID signál. Velice populární jsou anténní rohože. Jedná se o anténní cívky zabalené do ochranného obalu, který se položí na místo cílové čáry, aby přes něj mohli přebíhat běžci. U tohoto řešení se nabízí, aby se RFID tagy nacházely na obuvi závodníků, avšak rozhodně se nejedná o podmínku. Nevýhodou tohoto systému je jeho velká nepraktičnost například při závodě cyklistů nebo motokár. I přes to, že existují nízko profilové rohože, se pro svoji universálnost a bezpečnost také často používají postranní antény. Ty se nejčastěji vyskytují ve dvojicích – každá na jedné

straně cílové čáry. V případě potřeby zachytit rychlejší typy závodů se tyto antény doplňují ještě o další umístěné na horní části konstrukce cílové brány.

Další nezbytnou položkou do nákupního košíku jsou RFID štítky a startovní čísla. Velice populární štítky SMARTRAC DogBone Monza R6 se dají pořídit za cenu okolo šesti korun za kus (cena se silně obvíjí od objednaného množství). Celé startovní číslo, tedy štítek nalepený na nepromokavou destičku s vlastím potiskem, stojí okolo čtyřiceti korun. Pak už záleží na pořadateli závodu, zdali se rozhodne mít univerzální design nebo bude pro každý závod nakupovat či tisknout nová startovní čísla, která zároveň mohou obsahovat i jména závodníků.

Cena celého systému také závisí na uspořádání závodu. Pokud start a cíl nejsou umístěny na stejném místě, je potřeba rozpočet prakticky zdvojnásobit. Je tu samozřejmě možnost měřit čas závodníkům pouze v cíli, avšak při větším množství účastníků není vždy možné, aby startem proběhli všichni najednou. V takovém případě by tedy nebyly zajištěny stejné podmínky pro každého sportovce.

Mít záložní systém je dobrou radou prakticky pro jakoukoliv práci s technikou a při měření závodů tomu není jinak. Jelikož RFID není na milisekundy přesnou záležitostí, cílová kamera je, pro případ nerozhodného výsledku a zkoumání cílové fotografie, nutnost. Neuškodí tedy si zkontrolovat, že záznam obsahuje přesný čas, díky kterému je zároveň možné zrekonstruovat cílové časy závodníků v případě poruchy nebo výpadku primárního měřícího systému.

Bez započítané ceny za notebook a software, který je možné si napsat ve většině případů sám, by sestava základního vybavení mohla vypadat následovně. Čtečka Impinj R420 za 35 000 Kč, dvě MTI MT-263003/N antény každá za 3000 Kč, stativ a uchycení pro obě antény každé za 1500 Kč a kabeláž pro jejich připojení za 4400 Kč. Celková cena tohoto vybavení se blíží částce 50 000 Kč [11].

2.2 Profesionální řešení na míru

Pro představení již existujícího systému stvořeného přímo pro měření závodů je možné zmínit například řešení od firmy RACE RESULT. Nejenom, že mají přehledné stránky a velké množství užitečných videí, jak pracovat s jejich systémem, zároveň nabízejí vícero řešení pro používání aktivních štítků, které mnoho firem nenabízí.

Pasivní řešení od této společnosti se skládá z robustního kufru, který zahrnuje RFID čtecí zařízení, výrobcem naprogramovaný blíže nespecifikovaný počítač pro správu

závodu (není potřeba vlastní notebook či jiné další zařízení), baterii s výdrží minimálně osm hodin a GPS modul pro synchronizaci přesného času. Dále je zde k dispozici bohatá nabídka konektorů. Konkrétně obsahuje 8 krát BNC pro připojení antén, 2 krát RJ-45 (zahrnuje integrovaný síťový prvek switch) nebo 4G/LTE modul se slotem na sim kartu pro online komunikaci a USB port pro automatické zálohování dat. V sadě s 4,8 metrů dlouhou anténní rohoží se cena vyšplhá mírně přes 100 000 Kč.

Aktivní řešení se vyznačuje vysokou přesností (až 4 ms), levnější nákupní cenou antén, avšak mnohonásobně dražšími štítky. Firma RACE RESULT nabízí tři typy aktivních tagů, které se liší přesností, jakou rychlost zvládnou detekovat (až 250 km/h) a životností. Nejprodávanější varianta dosahuje přesnosti 10 ms a stojí 1300 Kč jeden kus. Další velkou výhodou je 100% pravděpodobnost na detekci štítku. Tagy totiž využívají frekvenci 2,4 GHz a komunikují oběma směry. Vysílají tedy informaci o své přítomnosti, dokud nedostanou potvrzení o doručení. Nejlevnější sestava se u aktivního systému pohybuje kolem ceny 80 000 Kč (bez nákladných štítků) [14].

2.3 RFID alternativy pro měření závodů

RFID technologie se stala pro pořádání závodů volbou číslo jedna. Avšak zvažovat alternativu je možné hnedka z několika důvodů:

- Pořizovací cena i toho nejzákladnějšího vybavení se pohybuje v řádu desítek tisíc korun. Takový nákup může být, speciálně pro malé závody, nerealizovatelný.
- Hmotnost rohoží, které obsahují antény pro RFID a pokládají se například přes cílovou čáru, se pohybuje okolo šesti kilogramů na jeden metr. V případě závodu, kde k cílové čáře nevede silnice nebo není umožněn vjezd vozidlům, může být rozhodujícím faktorem hmotnost celého vybavení.
- Alternativa může sloužit i jako méně přesný a levný záložní systém, který by měl být součástí každého amatérského či profesionálního měřeného závodu [11].

2.3.1 Čárový kód

Velkou výhodou používání čárových kódů je, že každý závodník si svůj přidělený může doma vytisknout. Pořadatel tak nemusí zaštiťovat a programovat RFID tagy a starat se o jejich následnou distribuci konkrétním závodníkům. Stačí dorazit na start a je možné vyrazit. Dále není potřeba žádné speciální vybavení. Číst čárové kódy dnes zvládne každý chytrý telefon.

V cíli poté stojí dva lidé. Jeden, který rozdává žetony reprezentující pořadí v závodě, a druhý, který zaznamenává časy pro jednotlivá umístění (to kvůli tomu, aby se na cílové čáře netvořily fronty). Závodník poté s žetonem přistoupí k třetí osobě. Ta naskenuje jeho čárový kód a do systému přiřadí pozici podle obdrženého tokenu.

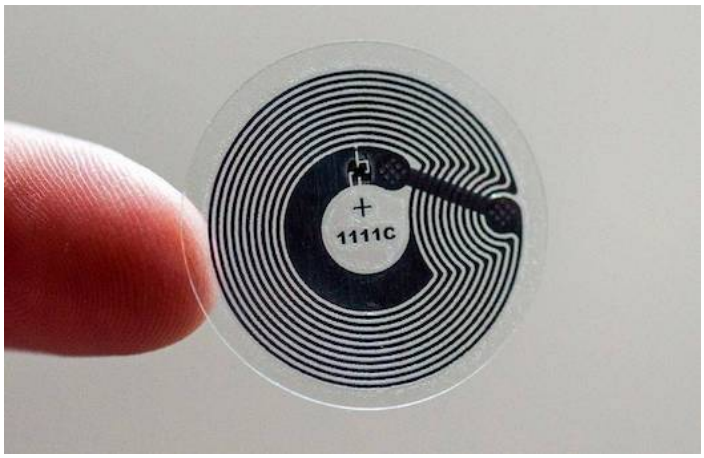
Tuto metodiku používá například anglická společnost parkrun Limited [2], která již zaštítila téměř 170 000 závodů po celém světě. Jedná se o amatérské závody bez startovního poplatku. Jsou nejčastěji pořádány jednou týdně v sobotu dopoledne. V České republice se ale bohužel žádný nekoná a nejbližší závod k Libereckému kraji je v polském městě Jelenia Góra (kvůli aktuální Covid situaci je mnoho závodů zrušeno).

Obrázek 3: Ukázka čárového kódu pro závody [1]

Tak levné řešení má samozřejmě svá úskalí. Za zhoršených světelných podmínek je obtížnější kód načíst (především s mobilním telefonem). Dále není vhodné, aby čárový kód vytisknutý závodníkem neprošel žádnou další úpravou. Zmačkaný nebo promočený papír, ať už od potu či deště, nemusí být možné načíst. Většinu těchto nevýhod odstraňuje použití NFC štítků.

2.3.2 Near field communication – NFC

Štítky NFC se při závodech používají podobným způsobem jako čárové kódy. Jejich měřící dosah je v jednotkách centimetrů, a tedy pokrytí celé cílové čáry by bylo nereálné. Avšak pokud se již malé vzdálenosti dosáhne, čtení je bezproblémové. Štítek může být mokrý, špinavý, poškrábaný nebo skenovaný v úplné tmě, a i přesto poskytne svoji hodnotu.



Obrázek 4: NFC štítek [1]

Přítomnost NFC v chytrých mobilních telefonech se v posledních letech rapidně zvýšila. Jak již bylo zmíněno, mohou za to bezkontaktní platby provozované právě touto technologií. Z aktuálně 894 prodávaných dotykových telefonů na portále CZC.cz jich 629, tedy 70%, podporuje NFC. I v tomto případě proto není nutný nákup dalšího vybavení pro provoz závodu. Pro jeho zúčastnění je potřeba NFC štítek, jehož cena je sice pouze v řádu jednotek korun, avšak jedná se o starost navíc oproti obyčejným čárovým kódům.

2.3.3 Bluetooth

Bluetooth je technologie, kterou rovněž nalezneme u každého nového chytrého mobilního telefonu. V tomto případě by se nutnost mít takové zařízení přesunula na jednotlivé závodníky. Díky unikátnosti MAC adres chytrých telefonů je možné tyto adresy přiřadit každému závodníkovi jako startovní číslo. Avšak i u tohoto řešení se vyskytuje mnoho nevýhod. Přestože Bluetooth má velmi vysoký dosah, nebylo by možné určit přesnou polohu telefonu vzhledem k cílové čáře. Navíc také dochází k situacím, kdy se probíhající závodník nestihne načíst vůbec [5]. Tato technologie by se tedy mohla používat například na stanovištích kontrolních bodů (zde přesný čas nehraje příliš velkou roli). Nebo, jako u NFC či čárových kódu, pouze pro identifikaci závodníka za cílovou čárou pro zápis do systému.

Existují i profesionálnější řešení, kdy se namísto RFID antén do rohoží umístí BLE (Bluetooth Low Energy) přijímače. Nejedná se o masivně vyráběný produkt, a tudíž zprovoznění takového zapojení již vyžaduje pokročilé znalosti. Každá implementace se proto může více či méně lišit. Odměnou potom je přesný cílový čas s maximální naměřenou tolerancí ± 500 ms za přijatelnou cenu [6]. Přesnost je z velké části ovlivněna

umístěním BLE štítku. Pokud závodník bude mít štítek umístěn na levé noze, ale touto částí těla překročí cílovou čáru až jako poslední (tak jak je to znázorněno na obrázku 5), nepřesnost měření se zásadně zvýší. Tento fakt ovšem není možné dávat za vinu technologii BLE. Cílová kamera s dostatečným množstvím snímků za vteřinu je tedy nutným doplňkem takového zapojení při profesionálním použití.

Obrázek 5: Závodník s BLE štítkem na levé noze [6]

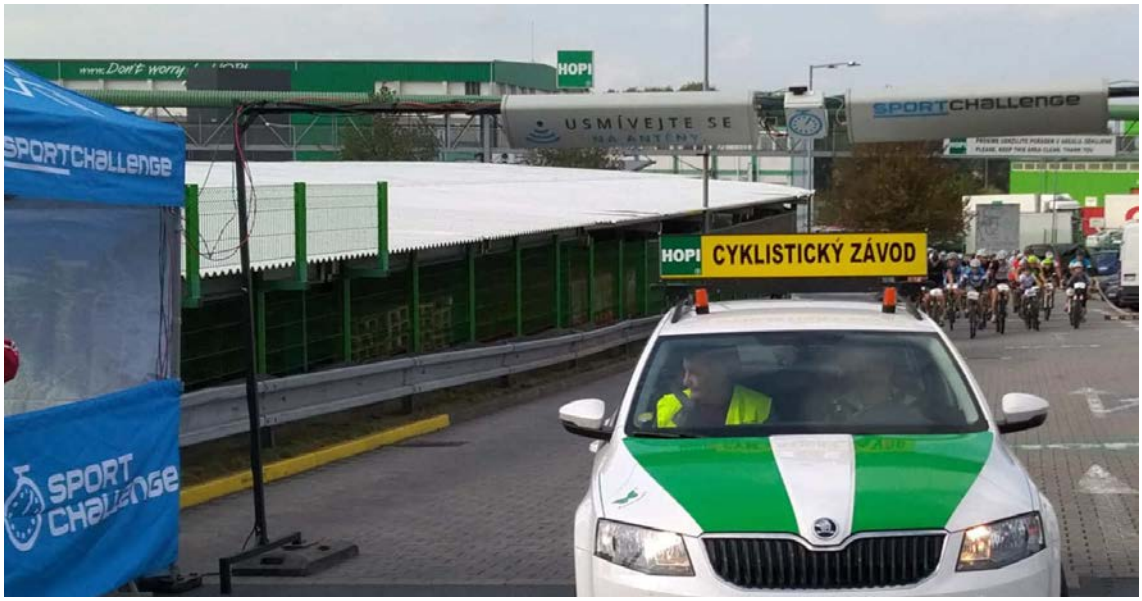
2.4 Sportchallenge

Výstup této práce bude sloužit potřebám časomíry Sportchallenge, což je jeden z mnoha subjektů působících v oblasti časomíry sportovních akcí v českém prostředí. Původně projekt tří studentů Technické univerzity v Liberci, který začal roku 2008, je nyní profesionální čipovou časomírou. Sportchallenge pořadateli závodů zajistí kompletní službu zahrnující registraci účastníků, výběr startovného, výrobu RFID startovních čísel, zajištění cíle a kontrol na trase, tisk a zveřejnění výsledkových listin.

2.4.1 Aktuální hardware a software

Sportchallenge již má vlastní řešení pro měření času na cílové čáře. Jedná se o konstrukci s RFID anténami umístěnými nad probíhajícími závodníky. Její podobu je možné vidět na obrázku 6. Srdcem této sestavy je stacionární čtečka Impinj R420 Speedway Revolution zobrazena na obrázku 1 s vlastním softwarem pro komunikaci se serverem www.sportchallange.cz. Cílem této diplomové práce je naprogramování mobilní aplikace pro komunikaci s mobilní čtečkou od společnosti Technology Solutions popsanou v kapitole 3. Ty se budou používat pro rozšíření stávajícího systému

Sportchallenge na kontrolních stanovištích, kde přesný čas není důležitý, ale podstatné je, zdali závodník na tomto stanovišti fyzicky byl. Čtečka bude komunikovat přes Bluetooth s mobilní aplikací, která naskenované tagy zpracuje a případně odešle na server www.sportchallenge.cz.



Obrázek 6: Ukázka cílové brány Sportchallenge (autor: www.sportchallenge.cz)

3 Mobilní čtečka TSL 1128 Bluetooth® UHF RFID

Zařízení 1128 RFID od společnosti Technology Solutions (TSL) je jak mobilní čtečka, tak zároveň přepisovač vysokofrekvenčních (UHF) štítků. S cenovkou 795 liber (k 10. srpnu 2020) se jedná o jedno z levnějších řešení této společnosti. Nabízí senzor s dosahem čtení až sedm metrů, ochranu IP54, možnost rozšíření o SD kartu, komunikaci pomocí USB nebo Bluetooth 2.1, NFC pro snadné spárování a baterii o velikosti 2400 mAh.

Tato čtečka je certifikována na komunikaci s RFID štítky první třídy druhé generace (ISO 18000-6C). Druhá generace přinesla vyšší výkon a délku kódu až 256 bitů. První třída znamená, že čtečka umí zapisovat na štítky prakticky bez omezení (výrobci štítků uvádějí více jak 100 000 programovacích cyklů). Nultá třída neumožňuje přepisování, zatímco štítky s certifikací třídy třetí a vyšší již mohou mít integrované senzory pohybu, teploty, tlaku a mnohé další. Dále se již nemusí jednat pouze o pasivní prvky, nýbrž mohou obsahovat baterie, které mnohonásobně zvýší dosah komunikace a umožní například komunikovat s okolními štítky [4].

3.1 Sady vývojářských nástrojů (SDK)

Pro práci se čtečkou nelze napsat jednu univerzální aplikaci, která by splňovala požadavky každého zákazníka, každé firmy. Společnost Technology Solution proto spravuje a aktualizuje několik sad vývojářských nástrojů pro různé platformy a programovací jazyky. K dispozici jsou aktualizované balíčky pro iOS, Xamarin nebo Android. Dále s poslední aktualizací v roce 2019 i pro .NET, .NET CF a Windows Phone.

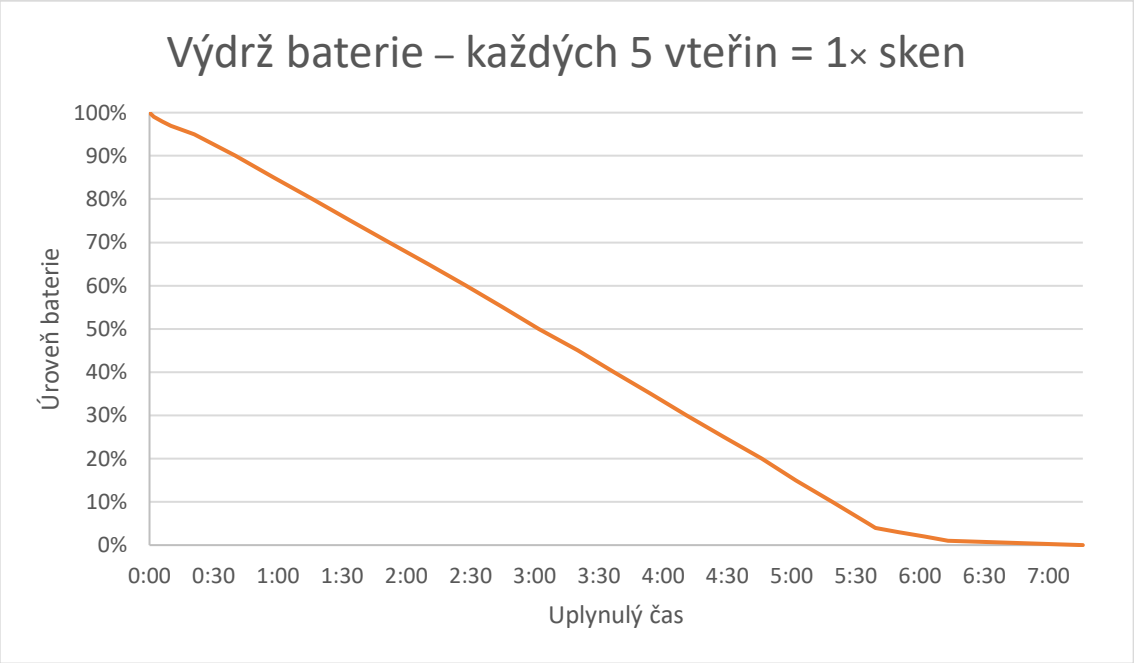
Tyto sady obsahují knihovny pro práci se čtečkami a k nim velice strohou dokumentaci, manuál s označením „Jak začít“, a v poslední řadě několik rozsáhlých kódů ukázkových aplikací. Čeho je programátor schopen s těmito nástroji dosáhnout je možné si vyzkoušet v některých z ukázkových aplikací. Ty lze stáhnout z distribučních platforem Google Play, App Store nebo Microsoft Store.

3.2 Test výdrže baterie

Čtečka je vybavena 3,7V baterií o kapacitě 2400 mAh. Výrobce udává, že při opravdu nejvytíženějším scénáři baterie vydrží minimálně dvě hodiny a zpátky na 100% se nabije za dvě a půl hodiny.

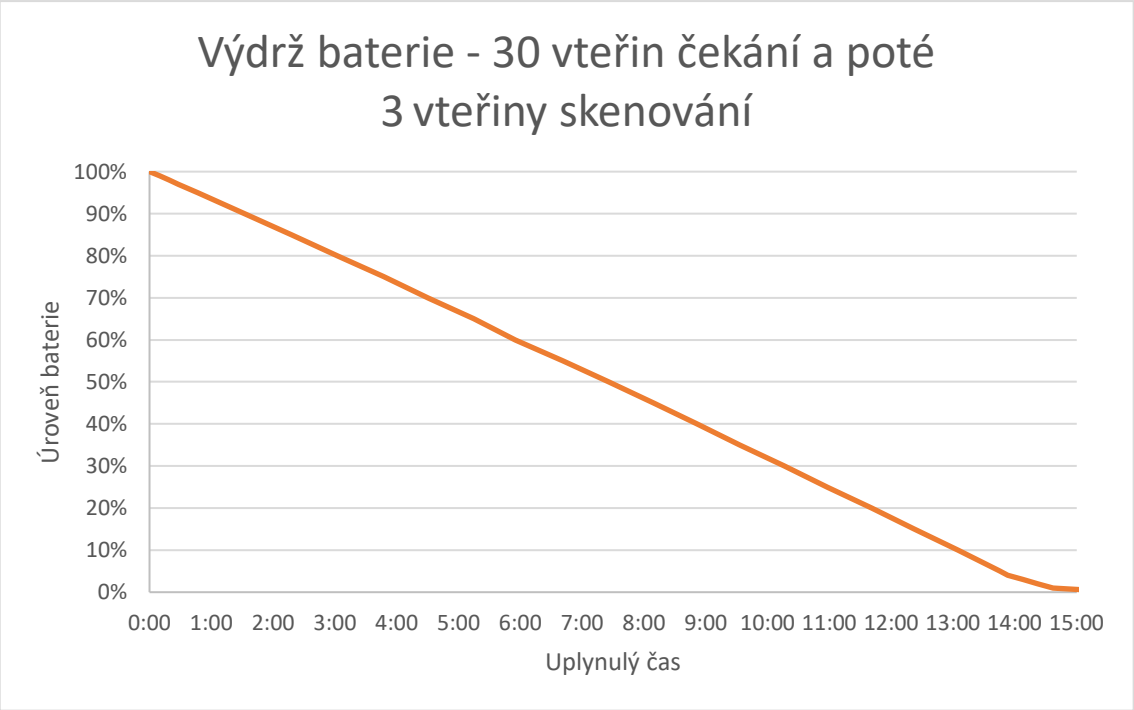
Jelikož atletické závody mohou trvat i několik hodin, bylo provedeno více zátěžových testů s cílem zjištění, zdali je tato čtečka pro takové použití vhodná. Pokud by výsledky dopadly nepřívětivě, bylo by nutné počítat s náhradními bateriemi či jiným způsobem dobíjení čtečky.

Pro potřeby testování byla napsána krátká Android aplikace, která po připojení ke čtečce začne každých pět vteřin posílat požadavek na provedení skenování o maximálním výkonu. Z grafu 1 je patrné, že čtečka při velké zátěži v testu vydrží necelých šest hodin. To odpovídá přibližně čtyřem tisícům skenování.



Graf 1: Výdrž baterie čtečky – každých 5 vteřin = 1× sken

Při nízkém stavu baterie, přesněji při 3 % a méně, čtečka přestane skenovat. Přesto zůstává stále připojená a komunikuje. Umožňuje tak stále vyžádání informací například o MAC adrese nebo o stavu baterie, avšak na požadavky nastavení výkonu snímání či samotné skenování reaguje chybovou hláškou: „Battery level too low“ (úroveň baterie příliš nízká). Důsledek této vlastnosti čtečky je vidět v grafu 1 i grafu 2, kdy ke konci má křivka téměř vodorovnou podobu, což symbolizuje velmi mírné vybíjení baterie.



Graf 2: Výdrž baterie čtečky – 30 vteřin čekání a poté 3 vteřiny skenování

Pro druhý scénář byla zvolena situace více se podobající simulaci skutečného závodu. Pokud závodník bude probíhat kontrolním bodem, obsluha RFID čtečky nezmáčkne spoušť pouze jednou, nýbrž ji bude držet, dokud nebude mít jistotu, že se naskenování podařilo. Jako čekací doba mezi závodníky bylo zvoleno třicet vteřin, což je typický rozestup mezi závodníky při startování u závodu s posunutým startem i jednotlivých sportovců.

Ze dvou výše uvedených grafů je patrné, že se pro velkou většinu závodů není potřeba zatěžovat externí baterií. Pokud se nebude jednat o nesmírně časově náročný závod nebo závod s velice vysokou účastí závodníků, bude interní baterie zařízení nabitá na 100 % stačit.

3.3 Porovnání s konkurencí

Pro srovnání byly vybrány dvě čtečky: Denso Wave SP1 handheld reader a druhá od firmy Alien Technology, model ALR-H460. Druhá zmíněná zastupuje skupinu čteček s již zabudovaným zařízením se systémem Android.

TSL a Denso používají stejný RFID čip od firmy Impinj [20]. Obě čtečky komunikují pouze přes Bluetooth nebo USB, mají stejnou třídu ochrany i podobnou výdrž baterie. Denso čtečka se vyznačuje rychlostí čtení až 700× za vteřinu jinak si jsou na papíře velmi podobné. Obě také nabízejí vývojářské sady pro Android, iOS i Windows.

Dalším sledovaným faktorem je ergonomie posuzovaných zařízení, konkrétněji jak dobře, či jakými způsoby, je možné ke čtečce připevnit mobilní telefon. TSL nabízí příslušenství, díky kterému je možné upevnit telefon ke čtecímu zařízení, avšak jedná se pouze o velmi malé množství modelů a žádný z nejnovějších se mezi nimi nenachází. Nabízí také univerzální řešení, kdy je možné libovolný kryt mobilního telefonu buďto přišroubovat (a zároveň nenávratně zničit) nebo přilepit oboustrannou páskou ke čtečce. Denso má oproti tomu v nabídce držák na svůj mobilní terminál anebo universální s uchycovacím systémem Quad Lock. To znamená, že je možné si po nákupu speciálního pouzdra na mobil ke čtečce pohodlně připevnit jakýkoliv telefon od firem Apple, Google, Samsung řady „Note“ a „S“ a pár dalších.

Výhodou řešení od firmy Alien Technology je, že není třeba řešit žádný způsob upevnění telefonu, protože řešení je dodáváno již s jakýmsi terminálem, telefonem, který nahrazuje klasický mobilní telefon. Tento terminál je propojen se čtečkou a celá sestava se nabíjí jedním konektorem. Na oficiálních stránkách ani na jiných internetových obchodech se nenachází informace o prodeji telefonu či čtečky jako samostatných dílů. Předpokladem tedy je, že pokud se rozbije telefon, je RFID čtečka již nepoužitelná a naopak. Zákazník je dále vázán k vývoji své aplikace na platformu Android a to vcelku zastaralou. Přestože bylo zařízení umístěno na trh přibližně v půlce roku 2018, pracuje na Androidu verze 6, které je z roku 2015. Nejaktuálnější verzí je však Android 11. To by mohlo pro některé firmy působit jako bezpečnostní riziko. Zároveň s novějšími verzemi Androidu se nevyvíjí jenom jeho bezpečnost a nové funkce pro uživatele, ale vylepšují se například i vývojářské nástroje. Ty pak mohou programátorovi usnadnit při programování aplikace mnoho práce.

4 Android aplikace pro rozšíření stávajícího systému Sportchallenge

Jak již bylo zmíněno, cílem této diplomové práce bylo vytvoření aplikace, která by v případě úspěšného testování, mohla sloužit pro skenování závodníků probíhajících či projíždějících kontrolním bodem. Pro napsání aplikace byl vybrán programovací jazyk Java a to z důvodu stále dominujícího zastoupení telefonů na trhu se systémem Android. Jelikož součástí zadání nebyl požadavek, aby aplikace fungovala i na operačním systému iOS, nebyla zvažována ani vývojářská platforma Xamarin pro vývoj multiplatformní aplikace. Jako vývojové prostředí bylo vybráno Android Studio, protože se jedná o oficiální vývojové prostředí (IDE) pro vytváření programu na platformu Android [15]. Jako takové je neustálé vyvíjené, s velkým množstvím aktualizací. Právě kvůli jejich velké četnosti zde ani není uvedena verze programu, ve které byla aplikace napsána. Ta je mířená na zařízení se systémem Android verze 8.1 a novější.

Jedná se o aplikaci pracující s RFID čtečkami od společnosti Technology Solutions a se serverem na adrese www.sportchallenge.cz. Aplikace bude sloužit k modernímu snímání závodníků na kontrolním stanovišti za pomocí technologie RFID. Jejím úkolem je zachytit a vyhodnotit naskenovaná data, potřebné informace odeslat na server a v neposlední řadě stáhnout a uložit startovní listinu závodu do příslušných objektů, se kterými následně pracuje. Zároveň si všechna data ukládat lokálně a co nejlépe je zobrazit uživateli. Aplikace byla vyvíjena a testována na zařízeních LG Nexus 5X (Android 8.1) a Google Pixel 4a (Android 11).

4.1 Komunikace a práce se čtečkou

Ať už je čtečka připojena přes USB nebo bezdrátově přes Bluetooth, komunikace je řízena protokolem ASCII verze 2.5 vyvinutým přímo společností Technology Solutions. Aplikace tedy není vázaná na práci s jedním konkrétním typem čtečky, nýbrž na jednoho výrobce. Komunikace je převážně vedena ve stylu požadavek – odpověď. Kromě situace, kdy je stisknuto tlačítko spouště na čtečce, které způsobí skenování a pošle data do aplikace, čtečka není iniciátorem komunikace.

Poskytnutý tutoriál od výrobce má kvalitně zpracovaný návrh připojování a odpojování čtečky, správu Bluetooth zařízení daného telefonu a logování informací z probíhající komunikace. Postrádá ovšem například ukázku, jak pracovat s informacemi,

které čtečka naskenuje a pošle. Přeci jen hlavní práce každé aplikace pracující se čtečkou je získat a zpracovat štítky, které čtečka načetla, a to se v tutoriálu nenachází.

Dokumentace k ASCII protokolu je velmi strohá. Většina metod a tříd neobsahuje kvalitní popis. Ten je převážně tvořen jednou větou, ve šťastných případech i dvěma. To příliš nenapomáhá snadné orientaci ve velkém množství tříd, která tato sada nabízí, či v jejich propojení. Většinou nezbývá nic jiného než odhadovat podle názvu, o co by se mohla příslušná třída starat, co by mohla daná metoda vykonávat.

Ukázkových aplikací je k dispozici velké množství. Kód je místy komentovaný ale i přesto není snadné se v něm zorientovat. Aplikace jsou totiž rozsáhlé, nikoliv kousky kódu popisující určitou činnost. A to navazuje na problém, že se s případnými dotazy není možné obrátit na oficiální dokumentaci, protože tam se hledané odpovědi nenachází.

Například na následujícím obrázku 7 se nachází výňatek z dokumentace k metodě `setSinglePressRepeatDelay()`, která nastavuje zpoždění mezi jednotlivými skeny, pokud uživatel drží stisknutou spoušť. O co se přesně tato metoda stará a jaký konkrétní parametr přijímá (vteřiny, milisekundy) je nutné zjistit testováním, protože se zde nenachází žádný další komentář.

setSinglePressRepeatDelay

```
public final void setSinglePressRepeatDelay(int singlePressRepeatDelay)
```

Obrázek 7: Ukázka nedostatečně okomentované metody v dokumentaci

Veškerá komunikace se čtečkou a správa spárovaných Bluetooth zařízení je prováděna přes poskytnuté SDK společnosti Technology Solutions. Aplikace se po spuštění zaregistruje, aby jakákoliv komunikace se čtečkou byla nejdříve logována a až poté předána dále ke zpracování. To aplikace provádí v přepsané metodě `processReceivedLine()`, která zpracovává textovou komunikaci řádek po řádku. Ukázka získaných dat při naskenování závodníka se startovním číslem 535 je vidět na obrázku 8.

```
I/LoggerResponder: >SW: single
I/LoggerResponder: >SW: off
I/LoggerResponder: >EP: CB20190407000000000000535
I/LoggerResponder: >RI: -43
```

Obrázek 8: Ukázka komunikace při naskenování štítku

Každý řádek komunikace nejdříve začíná dvoupísmenným kódem oznamující druh informace, kterému následuje hodnota samotná. Jejich významy jsou uvedeny níže:

- **SW** – informuje o stisknutí spouště na čtečce (single – jednoduchý stisk, double – dvojstisk) či o jejím uvolnění (off), v případě, že by uživatel držel spoušť déle, nejprve se zobrazí hodnoty štítků (některé opakovaně) a až poté by následovala informace o sundání prstu ze spouště,
- **EP** – hodnota naskenovaného štítku,
- **RI** – síla přijatého signálu v jednotkách dBm (čím je štítek blíže ke čtečce, tím je číslo bližší nule).

I přestože si čtečka nastavení ukládá, nikdy není jisté, k jaké jiné aplikaci byla naposledy připojena, a je tedy nutné si ji ihned po připojení nastavit. To je v aplikaci prováděno pokaždé, když je čtečka připojena, a zahrnuje informace jako: nastavení parametrů, které chceme od čtečky přijímat (výše zmíněné hodnoty SW, EP, RI), nebo jakým způsobem by čtečka měla provádět snímání. Dále aplikace vypne možnost detekce dvojstisku spouště, pro jednoduchý stisk nastaví funkci čtení dat a umožní kontinuální čtení tagů. Dále si zjistí minimální a maximální nastavitelný výkon a podle těchto hodnot nastaví v hlavním okně aplikace rozsah pro vizuální element `SeekBar` (posuvník pro získání hodnoty výkonu z nastaveného rozsahu). Jakmile uživatel přestane měnit hodnotu tohoto posuvníku, aplikace v metodě `onStopTrackingTouch()` do čtečky odešle informace o nově nastaveném výkonu snímání. V poslední řadě se aplikace každé dvě minuty dotazuje na stav baterie čtečky. Ačkoliv velké množství funkcí aplikace je závislých na připojené čtečce, lze ji spustit i bez její přítomnosti. Poté je v aplikaci například možné procházet startovní listinu nebo informace o závodě.

4.2 Třída pro vytvoření okna aplikace – Activity

Jedná se pravděpodobně o nejdůležitější stavební kámen aplikací napsaných pro Android. Aktivita je třída, která poskytuje okno s uživatelským rozhraním. Jedna třída odpovídá jednomu oknu aplikace, které ovšem může být vyvoláno z různých důvodů a z různých míst v kódu. Při programování aplikací pro osobní počítače se mnoho programátorů neobtěžuje s uspáváním softwaru při jeho minimalizaci a nechá ho dále konzumovat systémové prostředky. Mobilní telefony jsou zařízení, kde se velmi dbá na efektivitu práce s baterií, a proto má každá aktivita svůj životní cyklus upravený pro tyto potřeby. Ten se řídí pomocí metod `onCreate()`, `onPause()`, `onResume()`

a `onDestroy()`. První zmíněná se spustí při prvním načítání aktivity. A to v případě uživatelské či programátorské iniciace, nebo pokud aplikace byla z důvodu nedostatku paměti RAM ukončena (spustí se metoda `onDestroy()`) a je potřeba ji spustit znovu. Metoda `onPause()` se volá v případě, že jiná aktivita vstoupila do popředí anebo naopak metoda `onResume()`, když se vrací kontrola aktivitě původní. Vstupní branou do této aplikace je třída `MainActivity`.

4.2.1 Hlavní okno aplikace – MainActivity

Uživatel aplikace musí mít všechny informace k dispozici rychle a přehledně. Většina závodníků probíhajících mezičasem chce vědět svůj čas nebo svoje pořadí v závodě při průběhu. Někteří již nemají síly na to pokračovat a chtějí být diskvalifikováni. Nastávají i situace, kdy si to závodník znovu rozmyslí a v závodě by rád opět pokračoval. Zároveň uživatel aplikace musí být upozorňován, pokud čtečka naskenovala štítek, který nezná, pokud závodník probíhá mezičasem v neočekávaném čase, nebo pokud se nepodaří poslat data na server. Při všech těchto situacích aplikace musí být připravená na skenování, i když se uživatel zrovna nenachází v hlavním oknu aplikace.

Ačkoliv správné chování by bylo, kdyby aplikace ukončovala spojení se čtečkou kdykoliv je minimalizována, je zde brát ohled na uživatele, pro kterého není žádoucí, aby na svém mobilním telefonu po celou dobu závodu sledoval pouze okno aplikace. Automatické odpojování a připojování čtečky bylo bráno v potaz a zkoumáno, avšak výsledky rychlosti automatického připojení nevyhovují potřebám závodu. Nelze zaručit, že bude dostatečný výhled na blížící se závodníky, nebo že uživatel bude věnovat 100% pozornost závodu. Skenování tedy funguje dokonce i v případě, pokud se uživatel nachází v rozhraní Android nebo v jiné aplikaci.

Hlavní okno aplikace obsluze čtečky zobrazuje:

- všechny aktuálně naskenované závodníky – jejich číslo, jméno, čas a umístění v závodě (na příslušné trase i v dané kategorii),
- počet závodníků, který má obsluha na mezičase ještě očekávat,
- počet oznámení, která je potřeba vyřešit nebo jim alespoň věnovat pozornost,
- zdali je připojena čtečka, a pokud ano tak i její název a procentuální stav baterie,
- možnost nastavit výkon čtečky při skenování.

Třída `MainActivity` je výchozí aktivitou pro spuštění aplikace. Stará se o spojení se čtečkou, stejně tak jako o aktualizaci výše zmíněných parametrů v případě potřeby. Zároveň slouží jako brána k dalším informacím. Při kliknutí na číslo reprezentující „zbývající počet závodníků“ nebo „počet oznámení“ je uživatel odkázán na jejich příslušný seznam. Obrázek čtečky přesměruje uživatele na aktivitu vypisující základní informace o čtečce. V rozbalovacím menu se dále nachází možnosti připojení, změny nebo odpojení čtečky, zobrazení informací o závodě a kompletní soupisky, vygenerování výsledkové listiny (pro mezičas, na kterém uživatel skenuje závodníky) nebo v poslední řadě důležitá možnost pro nastavení ID čtečky. To se následně používá pro získání konkrétní startovní listiny ze serveru nebo pro identifikace dat směřujících na server.

Největší prostor hlavního okna aplikace zabírá vizuální prvek `ListView` (slouží k vykreslení libovolného seznamu dat) pro zobrazení naskenovaných štítků. Ten se zároveň automaticky posouvá podle množství závodníků, takže obsluha aplikace má na očích vždy ty nejpozději naskenované závodníky. V případě, že se bude jednat o rozpoznaný štítek ze startovní soupisky (je podbarven zeleně), je možné si kliknutím zobrazit veškeré další dostupné informace o závodníkovi. V poslední řadě je možné pomocí vyskakovacího okna, vyvolaného tlačítkem „vyhledat závodníka“, manuálně zjistit informace o konkrétním atletovi zadáním jeho startovacího čísla.

PopupWindow

Za pomocí oficiální třídy `PopupWindow` bylo vytvořeno vyskakovací okno, do kterého je umístěn jednoduchý formulář pro vyhledání závodníka. Při zadávání čísla závodníka nevyskočí uživateli klávesnice, nýbrž číselník pro snazší zadávání. Při pozicování tohoto okna byla objevena nečekaná vlastnost, kdy nastavení pozice `Gravity.TOP` neznamenalo umístění k horní hraně rodičovského (nadřazeného) elementu, ani aplikace, nýbrž k horní hraně celého okna. Je tedy nutné si zjistit výšku stavové lišty systému Android a přičíst ji jako odsazení (offset).

4.2.2 Informace o čtečce – DeviceInfoActivity

Po připojení ke čtečce je uživateli umožněno podívat se na její základní informace. Například verze firmwaru pro zjištění, jestli se na internetu nenachází novější. Dále MAC adresu Bluetooth, minimální a maximální vysílací výkon zařízení nebo nastavený čas. Bohužel se nezdařilo nastavit aktuální čas čtecího zařízení, avšak ten by byl v případě úspěchu akorát neaktualizovanou kopií času z telefonu, odkud již aplikace čas využívá.

V dokumentaci je také zmínka, že čtečka nabízí informace o životnosti baterie, avšak tyto informace se nepodařilo získat.

Čtečka má také možnost několika nastavení. Kromě způsobu snímání, které až na výkon čtečky uživatele nemusí trápit, je možné například vypnout vibrace, změnit hlasitost pípání či výšku tónu. Dále je možné vypnout nebo zapnout Bluetooth či změnit název zařízení. Většinu těchto nastavení nelze provádět při způsobu komunikace právě přes Bluetooth, a proto je aplikace ani nemá implementované.

4.2.3 Okno s oznámeními – NotificationActivity

Jedná se o okno aplikace, kde uživatel nalezne všechna oznámení, které aplikace vytvořila. Může se jednat o následující chybové situace:

`AlreadyScannedNotification` – závodník již byl naskenován a není v pořádku, že byl v tento čas naskenován znovu. To znamená, že uplynula doba (tolerance) na průběh kolem čtečky a opuštění oblasti nebo v závodě není více kol, a přesto se štítek ocitl v oblasti stanoviště mezičasu.

`TooSoonNotification` – podobná situace jako ve výše zmíněném případě s rozdílem, že zde mohly nastat následující stavy. Závodník uběhl kolo tak rychle, až to není reálně možné, nebo se objevil na kontrolním bodě v čase, který předchází startovnímu zapsanému na soupisce.

`InvalidPrefixNotification` – čtečka načetla štítek, který nemá žádné vlastnosti naznačující, že by se jednalo o informaci související se závodem. S největší pravděpodobností se jedná o RFID štítek umístěný například na neodstřižené cedulce kusu oblečení. Pro jistotu i přesto aplikace vytvoří notifikaci a nechá uživatele rozhodnout, jestli informaci pošle na server.

`InvalidNumberNotification` – obdobný scénář jako výše, odlišný pouze jistotou, že se jedná o účastníka závodu (štítek obsahuje kód závodu), avšak jeho startovní číslo se nenachází na startovní soupisce. Je velice běžné, že závod umožňuje registrace i těsně před startem. Může se tedy stát, že se jedná o dodatečně zapsaného závodníka a je potřeba aktualizovat soupisku.

`ConnectionFailedNotification` – závody se často běhají mimo zalidněné oblasti, které bývají pokryté mobilním signálem. Je tedy třeba počítat s tím, že aplikace nebude mít stabilní připojení k internetu nebo bude úplně bez připojení. Pokud tedy při odesílání

dat na server nastane problém, vytvoří se nová notifikace a uživatel se může kdykoliv pokusit o opětovné odeslání.

Pokaždé, když je vytvořena notifikace, je s ní uložena i hodnota štítku a čas, kdy byl naskenován. Nezáleží tedy, kdy data budou odeslána, server bude mít i tak správné údaje. Tlačítko „aktualizovat“ projde všechna uložená oznámení a zkontroluje, zdali stále platí chybový stav. Pokud se jedná o notifikaci s problémem připojení, znovu se pak pokusí odeslat data na server. Zeleně podbarvená oznámení znamenají, že data jsou validní. Červená barva značí, že je něco v nepořádku. U každé notifikace se tedy nachází informace o hodnotě štítku, důvod vzniku a tlačítko odeslat a smazat. V rozbalovacím menu se poté nachází hromadné akce odeslat a odstranit všechny notifikace, které jsou následovány potvrzovacími okny. Všechny notifikace jsou ihned po vytvoření zároveň uloženy do interního uložiště telefonu a při načítání aplikace automaticky nahrány.

4.2.4 Informace o závodě – RaceInfoActivity

Pokud se uživatel chce podívat na informace o závodě nebo se podívat na startovní listinu některé trasy či konkrétní kategorie, z hlavního okna v rozbalovacím menu klikne na položku „soupiska“. Zde jsou uvedené údaje jako název závodu a jeho ID nebo ID čtečky, pro ověření, že se jedná o správná data; atribut „Poslední aktualizace“, který informuje, kdy byla soupiska naposledy aktualizována; informace o datumu závodu anebo prefixu, který by se měl objevovat v hodnotách naskenovaných tagů závodníků.

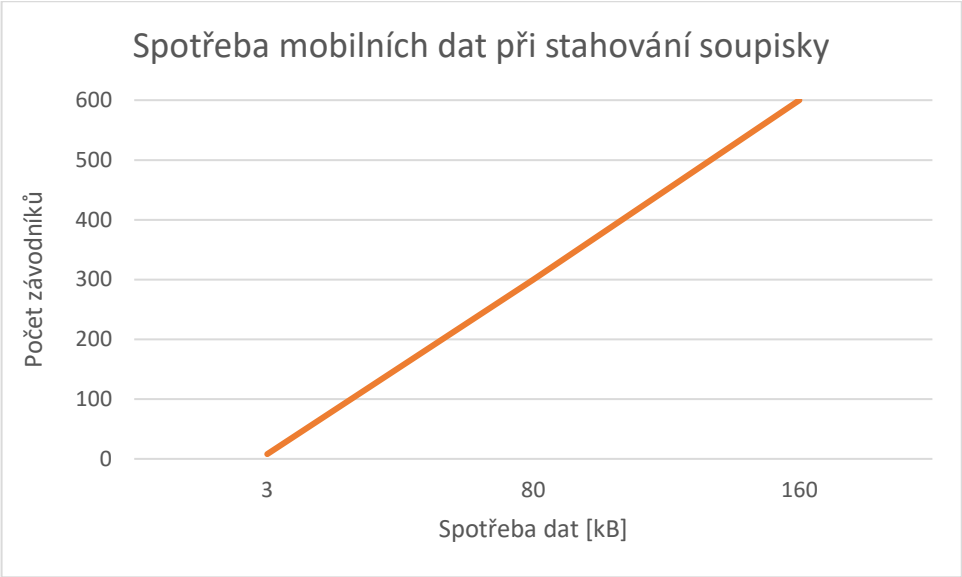
Pokud nechce uživatel zobrazit kompletní soupisku, slouží k tomu dvě rozbalovací nabídky. První je výběr trasy (jedním kontrolním stanovištěm může procházet několik tras, které jsou součástí jednoho závodu), který odkryje informace jako druh závodu (časovka, maraton); čas, kdy má být závod odstartován nebo kolik kol se na této trase soutěží. Zároveň umožní výběr ve druhé rozbalovací nabídce, kde je poté možné si zvolit konkrétní kategorii na dané trase (například ženy, muži, a tak dále). K zobrazení startovní listiny už jen stačí výběr potvrdit příslušným tlačítkem.

Informace o závodě či startovní listina se před nebo i během závodu mohou měnit. Velice často je umožněna registrace až na místě, případně se nestihne závod odstartovat v plánovaném čase. Je tedy nezbytné, aby soupisku bylo možné aktualizovat. U telefonu se předpokládá, že bude připojen za pomocí mobilních dat, a to i v místech se špatným pokrytím signálem. Aplikace si tedy ukládá datum a čas poslední aktualizace dat a tuto informaci předává serveru, který vygeneruje JSON soubor obsahující pouze data,

u kterých nastala od poslední aktualizace změna. Po každé aktualizaci se také zkontrolují všechny notifikace, zdali již nejsou validní, a tudíž mohou být odeslány na server.

Spotřeba mobilních dat

Startovní listina obsahující například 300 závodníků má velikost okolo 80 kB. Pokud bude přidán jeden závodník a uživatel aplikace si vyžádá aktualizaci dat (předá serveru informaci o datu posledního stahování soupisky), bude muset stahovat přibližně 3 kB dat. Taková úspora dat může být pro uživatele aplikace klíčová, zvláště pokud se nachází v terénu, kde mu k připojení na internet slouží pouze technologie EDGE, která nabízí přenosové rychlosti v jednotkách kilobajtů. Na následujícím grafu 3 je zobrazeno, jak počet závodníků ovlivňuje velikost stahovaného souboru.



Graf 3: Spotřeba mobilních dat v závislosti na počtu závodníků na soupisce

4.2.5 Zobrazení soupisky – LineUpActivity

Další třídou je univerzální okno pro zobrazení všech nebo pouze určitého množství závodníků, podle potřeby. Výpis obsahuje startovní číslo, celé jméno a rok narození. Tento výpis se používá pro zobrazení soupisky (jakkoliv filtrované podle typu závodu) nebo z hlavního okna pro získání seznamu závodníků, kteří ještě neproběhli kontrolním bodem. Z tohoto důvodu má aplikace na výpise umístěn rok narození, aby uživatel mohl na základě věku předpokládat, kdy se závodník objeví na mezičase.

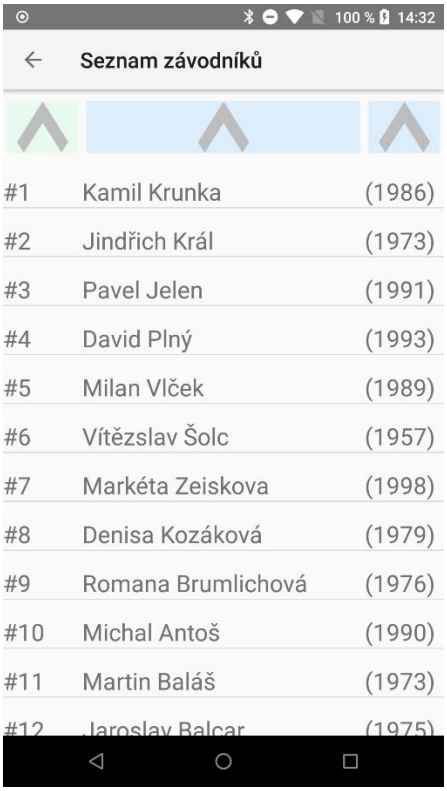
Třída obsahuje čtyři statické konstanty pro předání informace, jakou množinu závodníků je potřeba vypsat. Pokud tedy jiná aktivita vytváří instanci třídy `Intent` k zavolání třídy `LineUpActivity`, umístí do metody `putExtra()` příslušnou

statickou konstantu a požadovanou hodnotu (v následujícím zdrojovém kódu 1 je ukázka nastavení trasy a kategorie). Možnosti jsou: všechny závodníky na startovní soupisce, všechny z jedné trasy, všechny z konkrétní kategorie nebo právě již zmíněný seznam stále očekávaných závodníků.

```
if(selectedCategory.isPresent())
{
    intent.putExtra(LineUpActivity.SHOW_RACERS_IN_CATEGORY_STRING,
selectedCategory.get());
    intent.putExtra(LineUpActivity.SHOW_RACERS_IN_PACKET_ID,
selectedPacket.getIdPacket());
}
```

Zdrojový kód 1: Ukázka použití statických konstant ze třídy LineUpActvity

Aktivita dává uživateli možnost si výpis závodníků seřadit pomocí tří tlačítek umístěných nad výpisem. Zelené podbarvení znamená, že se jedná o aktuálně vybranou možnost. V tlačítkách jsou dále umístěné šipky, které symbolizují, jestli se jedná o sestupné (šipka směřující dolů) nebo vzestupné (šipka směřující nahoru) řazení. Kliknutí na libovolný z filtrů tedy způsobí zelené podbarvení, otočení aktuálního stavu šipky a samozřejmě seřazení dat.



Obrázek 9: Zobrazená startovní soupiska ve třídě LineUpActivity

4.2.6 Ostatní aktivity

V aplikaci se dále také nacházejí následující aktivity.

Informace o závodníkovi – RacerInfoActivity

Jakmile se někde v aplikaci objeví jméno závodníka, je možné na něj kliknout a nechat si zobrazit veškeré informace. Mezi nejdůležitější patří: jméno a číslo závodníka, umístění na trase a její název (například 5 km, 10 km), umístění v kategorii a její název (například muži, ženy), čas na kontrolním bodě, startovní čas nebo název týmu. V tomto okně se také provádí diskvalifikace případně zařazení závodníka zpátky do závodu. Oba případy jsou doprovázeny oknem na potvrzení operace.

Výsledková listina – ResultListActivity

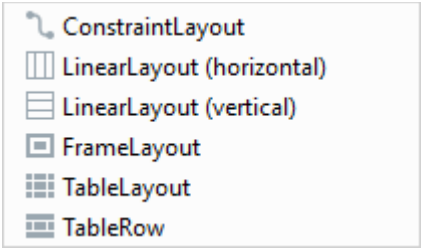
Pokud by uživatel nebo pořadatel závodu chtěli zjistit umístění sportovců na kontrolním stanovišti nebo by se čtečka náhodou používala na cílové čáře, aplikace nabízí zobrazení výsledkové listiny. Stejně tak jako při zobrazení soupisky, i zde je zavedené sdružování závodníků do skupin podle toho, na jaké trase se účastnili, a poté je možné nepovinné dělení podle toho, v jaké kategorii jsou zařazeni. Výpis seřazený podle umístění obsahuje číslo závodníka, celé jméno, čas a samotnou pozici v závodě.

4.3 Použité technologie v aplikaci

V následujících podkapitolách jsou popsány použité technologie při vývoji aplikace, ukázány praktiky a nástroje pro usnadnění programování nebo rozebrány důležité soubory projektu.

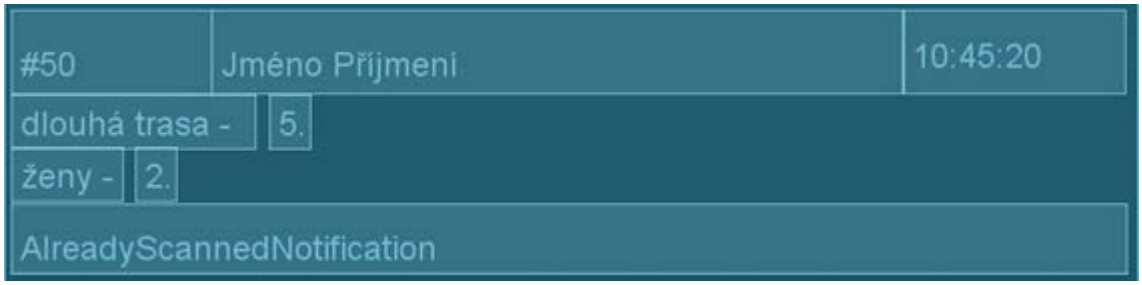
4.3.1 Layouty

Android Studio doporučuje pro rozmisťování prvků na obrazovku použití mnoha různých layoutů, tedy jakýchsi šablon, předpisů. Ty se liší přístupem k pozicování jednotlivých elementů.



Obrázek 10: Dostupné výchozí layouty v programu Android Studio [16]

Předpis `Constraint Layout` se skvěle hodí pro návrh responzivních uživatelských rozhraní. Byl představen v roce 2016 a umožňuje velmi flexibilně poskládat jednotlivé prvky na obrazovku. Definuje se pomocí předávání odkazů na jednotlivé prvky. To znamená, že každému prvku je podle potřeby určováno jeho umístění vzhledem k jinému elementu či okrajům obrazovky. Jeho použití je vhodné v případech, kdy se na obrazovce vyskytují nepravidelně umístěné prvky. Zároveň se velmi dobře postará o jejich responzivní chování. Každému prvku je totiž možné zadat procentuální hodnotu, kolik má v okně zabírat místa. Jedná se o náhradu za `Relative Layout`, který se již nenachází v nabídce. Android Studio dokonce nabízí funkci převést různé jiné layouty právě na `Constraint Layout`. Aplikace tento layout používá například pro zobrazení záznamu o naskenovaném závodníkovi, tak jak je znázorněno na obrázku 11.



Obrázek 11: Ukázka použití Constraint Layout

Předloha `Linear Layout` se vyskytuje ve dvou variantách – horizontální a vertikální. Právě druhou zmíněnou aplikace taktéž hojně využívá. Princip je jednoduchý. Podle toho, v jakém pořadí zapíšeme v kódu jednotlivé funkční prvky, v takovém budou seřazeny pod sebou (případně vedle sebe). Layout je vhodný pro většinu stavebních struktur jako třeba tvorbu jednoduchých formulářů. Kvůli tvaru mobilních telefonů je často žádoucí umisťovat prvky pod sebe. Jinak řečeno, vyhovuje v situacích, kdy prvky na obrazovce jsou umístěny systematicky pod sebou nebo vedle sebe.

Šablona `Frame Layout` je vhodný nástroj pro docílení překrývání jednotlivých elementů. V nejčastější kombinaci se používá pro zobrazení textového pole `TextView` přes obrázek `ImageView`. Ostatní layouty neumisťují jednotlivé prvky přes sebe, pokud jim chování programátor nevnutí. `Frame Layout` má tuto činnost zajištěnou automaticky. Pořadí prvků umístěných v kódu udává, který element bude posazen nejvýše, a tedy bude viditelný. Aplikace rozvržení využívá k vytvoření oznamovací

ikony, která udává počet notifikací vyžadujících pozornost. Ikona je zobrazena na následujícím obrázku 12.

Obrázek 12: Ukázka vytvoření ikony pomocí Frame Layout

`Table Layout`, jak již název napovídá, slouží k uspořádání prvků do tabulky. Každý řádek je definován pomocí layoutu `Table Row`, který se, pokud by nebyl potomkem `Table Layout`, bude chovat jako horizontální `Linear Layout`, od kterého také dědí. To umožňuje chování s výsledkem, že se nemusí jednat o pravidelnou tabulku, nýbrž každý řádek může obsahovat libovolné množství sloupců o různé šířce. Výhodou je, že pro `Table Row` není třeba určovat jejich výšku ani šířku. Šířka se automaticky určí, aby řádek vyplňoval celý `Table Layout`, a výška se přizpůsobí obsahu řádku.

Je naprosto běžné do sebe různé layouty vnořovat a kombinovat je. Právě díky této možnosti se za pomocí každého nákresu dá vytvořit prakticky jakékoliv rozložení prvků na obrazovce. Správný výběr layoutu může ve výsledku ušetřit spoustu práce, umožnit snazší implementaci změn, zlepšit výkon aplikace nebo snížit počet řádků kódu, a tedy i zlepšit jeho přehlednost.

Ačkoliv aplikace nemá implementované třídy typu `Fragment`, je možné ji používat v horizontální poloze, na což byla i testována. Současné ovládání čtecího zařízení a mobilního telefonu je bez přídavných držáků nepříjemnou záležitostí. Proto aplikace nepředpokládá, že by byla spouštěna na tabletech nebo na mobilních telefonech v horizontální poloze. Navíc by se v případě montáže ke čtecímu zařízení telefon taktéž nacházel v poloze vertikální. Dále byla testována na změnu velikosti písma v systému Android. I při největším možném písmu aplikace uživateli stále důstojně zobrazovala všechny důležité informace.

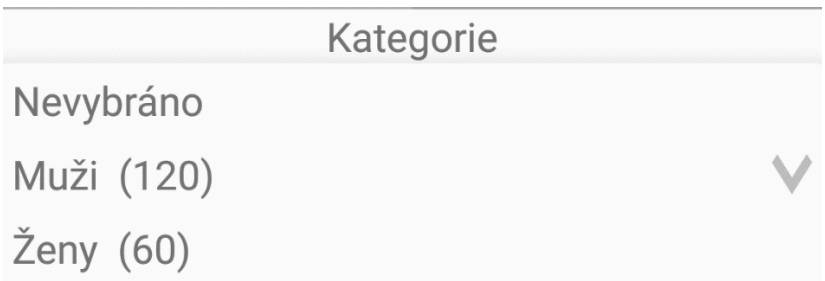
4.3.2 Definování vzhledu výpisu dat – ArrayAdapter

Pokud je potřeba zobrazit uživateli seznam dynamických dat, jejichž velikost není dopředu známa, je vhodné vytvořit vlastní `ArrayAdapter` a layout, který mu bude

přidělen. Návrh rozložení je vidět například na obrázku 11. Adaptér slouží jako zprostředkovatel dat mezi vrstvami *model* a *view*. Stačí mu nadefinovat, jaká data z objektu má napojit na jednotlivé prvky v layoutu (tzv. „nabindovat“). Podle velikosti listu nebo pole objektu, který je adaptéru předán, poté vytvoří příslušné množství těchto layoutů a přiřadí je do vizuálních elementů `ListView` nebo `Spinneru` (rozbalovací nabídka) dle volby programátora.

`MainArrayAdapter` je třída, která se stará o načtené tagy a jejich doplňující informace, a rozhoduje o jejich rozmístění na hlavním okně aplikace. Existují tři scénáře zobrazení tagu: vše je v pořádku; je znám závodník, ale nastala chyba; úplně neznámý tag. Tento adaptér provádí změny v rozložení prvků podle dostupných informací. Například pokud není možné identifikovat čtečkou načtený tag, je jasné, že také nelze uživateli zobrazit jméno a příjmení závodníka. Jestliže aplikace vyhodnotila, že závodník se ocitl na kontrolním bodě v neočekávaný čas, je použito rozmístění informací zobrazené na obrázku 11.

`RaceInfoArrayAdapter` je jediný adapter, který není aplikovaný na prvek `ListView`, nýbrž na `Spinner`, a to rovnou v několika případech. Liší se tím, že má navíc implementovanou metodu `getDropDownView()`, ve které je definováno, jak mají vypadat jednotlivé řádky, pokud se uživatel rozhodne si zobrazit rozbalenou nabídku. Je zde možné například speciálně upravit aktuálně vybraný řádek (aby byl rozpoznatelný od ostatních) nebo odlišit liché a sudé řádky pro lepší přehlednost textu. `Spinner` není v základním provedení připravený na to, aby mohl obsahovat možnost „nevybráno“, která je v rozbalovacích nabídkách populární. Tento adaptér se tedy stará i o to, aby se na vrcholu rozbalovací nabídky vždy tato možnost nacházela. To je vidět na následujícím obrázku 13 s vybranou kategorií „muži“, která obsahuje 120 závodníků.



Obrázek 13: Ukázka rozbalovací nabídky (Spinner)

`ResultListArrayAdapter` je adaptér použitý pro získání přehledné výsledkové listiny. Jakmile obdrží data o závodnících, kteří mají být zobrazeni,

automaticky si ke každému zjistí jeho umístění a celý výpis je následně seřazen od první pozice až po tu poslední. Případně se na konci objeví závodníci, kteří ještě závod nedokončili anebo jsou diskvalifikováni. Umístění závodníka nemusí být pouze klasické číslo. Jelikož mobilní čtečkou nelze zaručit přesné časy, aplikace vyhodnocuje stejné nebo velmi podobné časy závodníků jako jedno umístění v závodě. Pokud tedy čtečka naskenuje tři závodníky velmi brzy po sobě, bude jejich umístění na mezičase shodné například ve tvaru „1–3“. I tady si aplikace poradí a závodníci s takovými pozicemi se objeví na prvních třech řádcích výsledkové listiny. Dále se adaptér stará o vlastnost, kdy každý sudý řádek pro lepší přehlednost podbarví odlišnou barvou.

`LineUpArrayAdapter` dbá na zobrazení konkrétního seznamu závodníků. Ve výchozím stavu jsou seřazeni podle startovního čísla. Tento adaptér dále obsahuje tři metody pro uspořádání závodníků, a to `sortBySurname()`, `sortByNumber()` a `sortByYear()`. Neboli pro třídění podle příjmení, startovního čísla anebo roku narození. Každá z těchto metod obsahuje vstupní parametr datového typu `boolean`, kterým se metoda informuje, zdali má závodníky řadit vzestupně či nikoliv (sestupně). A na jejich konci se nachází volání metody `notifyDataSetChanged()` pro znovu vykreslení celé vrstvy *view*.

Aplikace dále adaptéry používá pro zobrazování notifikací nebo závodní soupisky. Jedná se o efektivní způsob, jak vytvořit vrstvu *presenter*, která přijímá data z *modelu* a v prezentovatelné podobě je předá *view*.

4.3.3 Soubor AndroidManifest.xml

Tento soubor musí být přítomen v každé Android aplikaci na přesně určeném místě, kterým je kořenová složka projektu. Obsahuje informace o aplikaci, jako jsou její název, ikona, výchozí schéma nebo nastavení zálohování. Dále je zde každá aktivita zaregistrována a může mít na sebe navázaný tzv. `intent-filter`. Ten například definuje, že se jedná o vstupní aktivitu do aplikace nebo že aktivita může komunikovat s ostatními aplikacemi a jakým způsobem.

Dále v tomto souboru nalezneme informace o oprávněních (přístup k souborům, internetu) a hardwarové požadavky (například, že telefon musí mít kameru, gyroskop), které software požaduje pro zajištění svých funkcí. Tato aplikace se pro svoje fungování dožaduje následujících oprávnění:

- `android.permission.INTERNET` – pro stažení startovní listiny a odesílání informací o naskenovaných závodnících na server,
- `android.permission.BLUETOOTH` – aby aplikace mohla vyzvat uživatele k zapnutí Bluetooth ve svém telefonu,
- `android.permission.BLUETOOTH_ADMIN` – aby si aplikace mohla zobrazit dostupná Bluetooth zařízení a připojit se k nim,
- `android.permission.ACCESS_FINE_LOCATION` – pro případné připojení BLE (Bluetooth Low Energy) zařízení k telefonu s operačním systém Android verze nižší než 8.0 [17].

4.3.4 Třída Intent

`Intent` se při vyvíjení aplikací nejčastěji používá pro nastartování nové aktivity v rámci aplikace. Té se navíc mohou předat dodatečné informace a po zavolání metody `startActivity(intent)` se uživateli zobrazí nové okno aplikace. To ovšem není jediné využití. Slouží také k zobrazení webové stránky, libovolného kontaktu, odeslání SMS nebo uskutečnění hovoru. Aplikace třídu `Intent` používá ke zjištění, zdali je Bluetooth zapnuté a pokud tomu tak není, uživateli vyskočí dialogové okno s možnostmi „ano“ či „ne“ zdali chce Bluetooth zapnout. Samozřejmě většina těchto operací se neprovede, pokud nebudou definovaná potřebná oprávnění v souboru `AndroidManifest.xml` a uživatel k nim nedá při prvním spuštění aplikace svolení.

4.3.5 Návrhový vzor Singleton

Návrhový vzor zajišťuje, že daná třída bude mít pouze jednu unikátní statickou instanci a nikdy jich nebude vytvořeno více. Jak už slovo statický napovídá, tato instance bude poté dostupná napříč celou aplikací. Těchto vlastností je docíleno za pomocí privátního konstruktoru (od třídy tedy není možné vytvářet instance mimo třídu), privátního atributu pro uložení instance, který je datového typu dané třídy a je iniciován na hodnotu `null` a dále metody `get()`. Ta v případě potřeby vytvoří jedinečnou instanci privátním konstruktorem anebo ji pouze navrátí. Používá se například k uchování databázového spojení nebo jiného způsobu přístupu k datům, udržení informace o přihlášeném uživateli, zajištění unikátnosti tiskové fronty nebo přístupu k jiným fyzickým zařízením, u kterých by mohl vícenásobný přístup způsobit problém.

Aplikace využívá tohoto návrhového vzoru k uchování připojení k lokální databázi `SQLite` použité k zálohování dat, dále pro přístup k informacím o závodě a startovní listině nebo k uložení aktuálně nastaveného ID čtečky.

4.3.6 Kontejner Optional

V kódu programu je spousta situací, kdy není na 100% zaručeno, že daný atribut bude obsahovat požadovanou informaci. Ať už se jedná o neočekávanou chybu nebo jenom nejistou situaci – třída `Optional` poskytuje řešení, jak tyto případy zabezpečit před pádem aplikace. Jakýkoliv datový typ nebo objekt lze zabalit do tvaru jako je vidět například na následující ukázce zdrojového kódu 2.

```
Optional<Date> kontrola;
```

Zdrojový kód 2: Ukázka použití třídy Optional

Za předpokladu, že závodník ještě neproběhl kontrolním stanovištěm a atribut `kontrola` má tedy hodnotu `null`, programátor musí mít při vývoji aplikace pořád na paměti, že pokud by pracoval s takovým atributem, je nutná jeho kontrola na hodnotu `null`. Třída `Optional` naopak donutí programátora ve všech případech čtení hodnoty k zamyšlení, zdali není vhodné zavolat metodu `isPresent()`, která vrátí stav pravda nebo nepravda podle toho, jestli atribut je nebo není `null`.

Aplikace tuto třídu využívá kromě výše zmíněného případu k ukládání celé soupisky. Jelikož aplikace funguje i bez načteného seznamu závodníků, musí každá aktivita kontrolovat, zdali jsou data k dispozici. Dále při vypisování startovní listiny uživatel nemusí zadat konkrétní trasu či kategorii. Díky tomu, že jsou zabaleny třídou `Optional`, bude programátor při snaze o získání dat na tento fakt upozorněn. Je možné označit i návratový datový typ metody. Tedy namísto neočekávaného chybového návratu hodnoty `null` nebo „-1“ je programátor přímo vyzván ke kontrole, zdali metoda neskončila v chybovém stavu.

4.4 Možnosti ukládání dat

Třída pro ukládání dat `SharedPreferences` je jednou z možností přístupu k pevnému uložišti telefonu, která je nabídnuta programátorovi. Jedná se o jednoduše implementovatelné rozhraní, které nabízí ukládání dat v podobě klíč – hodnota. Klíč musí mít podobu řetězce a hodnota musí být jedním z primitivních datových typů. Jak již název

napovídá, jedná se o způsob ukládání například předvoleb nežli velkých datových souborů. Android si tato data vnitřně ukládá v podobě XML souboru.

Pokud aplikace vytváří například JSON soubor nebo jiné, u kterých není žádoucí, aby k nim ostatní aplikace měly přístup, je vhodné je umístit do vnitřního uložiště telefonu do prostoru vyhrazeného pro aplikační data. Android brání ostatním aplikacím v přístupu k těmto datům a od verze Android 10 jsou dokonce šifrovaná. Při odstranění aplikace jsou následně smazána.

Pro ukládání dat je možné použít i externí uložiště. Před snahou na něj zapisovat je nutné si ověřit jeho dostupnost. Na uložišti by se neměly objevit žádná osobní či jiná citlivá data. Dále je k dispozici sdílené uložiště. Při uložení na toto uložiště budou data, jak je z názvu patrné, dostupná ze všech aplikací v telefonu.

Systém Android dále nabízí zabudovanou podporu pro open-source databázi `SQLite`. Ta pracuje nad souborem uloženým v privátní složce aplikace, ke kterému ostatní aplikace nemají přístup. Samozřejmostí je možnost vykonání základních CRUD operací (create – vytvořit, read – číst, update – aktualizovat, delete – odstranit) stejně tak jako používání dotazovacího jazyka SQL. Implementace třídy `SQLiteOpenHelper` požaduje přepsání dvou metod: `onCreate()` a `onUpdate()`. V první zmíněné se nachází příkazy pro vytvoření tabulek v databázi a vazby mezi nimi. Druhá slouží pro sepsání instrukcí, jak přetvořit databázi a data v ní, pokud se programátor rozhodne pro její aktualizaci a nechce, aby uživatel přišel o všechna data.

4.4.1 Třídy Gson, JsonSerializer<T> a JsonDeserializer<T>

Ukládání nebo předávání objektu jako celku se může zdát jako složitý proces, ale existuje spousta nástrojů, jak si práci usnadnit. Jedním z nich je `Gson`. Jedná se o open-source Java knihovnu vyvíjenou Googlem pro práci s daty typu JSON. Tento nástroj aplikace používá ve dvou případech.

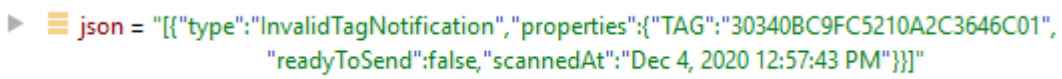
První je pro předávání objektů mezi jednotlivými aktivitami. Přesněji pokud je vytvořena instance třídy `Intent` pro nastartování jiné aktivity. Ta totiž neumožňuje nové aktivitě předat celý objekt, nýbrž pouze primitivní datové typy a jejich pole nebo třídy typu `Parcelable` či `Serializable`. K získání určitého objektu v podobě textového řetězce lze použít metodu `Gson().toJson()`, který `Intent` umí předat. V nové aktivitě se již pouze zavolá metoda `Gson().fromJson()`, které je předána kromě řetězce k převedení na JSON také informace, o jaký datový typ pole objektu se jedná.

V druhém případě aplikace používá `Gson` pro ukládání objektů na lokální uložiště. Jelikož i všechny typy notifikací v aplikaci je potřeba zálohovat a nejedná se o tak rozsáhlá data, byla zvolena kombinace `SharedPreferences` a `Gson`. Notifikace jsou uložené v listu datového typu abstraktní třídy `INotification`. Jedná se o předpis pro pět tříd, které představují různé druhy upozornění, na které aplikace reaguje odlišně. V tomto případě není možné pouze zavolat již zmíněné dvě metody. `Gson` potřebuje při zpětném načítání dat vědět, o jakou konkrétní notifikaci se aktuálně jedná. K tomu slouží třída s názvem `NotificationJsonAdapter`, která implementuje dvojici rozhraní `JsonSerializer<T>` a `JsonDeserializer<T>`. Jedná se o lehce vyhledatelnou implementaci, kterou si každý programátor doladí na míru svým potřebám.

Tato třída obsahuje dvě metody – `serialize()` a `deserialize()`. První se stará o to, aby do textového řetězce byla kromě samotných dat připsána i informace, o jakou konkrétní třídu notifikace se jedná. Druhá zase o to, aby tato informace byla navíc zohledněna při zpětném vytváření objektu z textového řetězce. Nyní už jenom zbývá pro každý typ notifikace zaregistrovat příslušnou třídu jako na následující ukázce zdrojového kódu 3. Výsledek tohoto procesu je vidět na obrázku 14.

```
gson.registerTypeAdapter(TooSoonNotification.class, new
NotificationJsonAdapter<TooSoonNotification>());
```

Zdrojový kód 3: Ukázka registrace adaptéru pro Gson



Obrázek 14: Výsledek práce třídy NotificationJsonAdapter

Ani pro jeden z těchto případů není vhodné tuto metodu používat pro objemná data. Pátrání po oficiálních hodnotách, které by určovaly maximální velikost takto přenášených dat, skončilo nezdarem. Různé nezávislé testy přinesly informaci, že maximální množství dat možné předat pomocí `intent.putExtra()` je mezi 500 kB a 1 MB [21], [22]. A pro uložení dat pomocí `SharedPreferences`, která jsou uložena jako XML soubor, je limitem maximální velikost objektu `String` [23], [24].

4.4.2 Porovnání rychlostí načítání a zápisu

Jako první ukázka je popsán test, při kterém byla data zabalená pomocí `Gson` a uložená použitím `SharedPreferences`. Jednalo se tedy o načítání z pevného uložiště telefonu a vytváření datových objektů do paměti RAM. Na starším telefonu

střední třídy – LG Nexus 5x (vydán 2015) trvalo nahrání 636 závodníků v průměru 2,5 vteřiny (načtení dat trvalo dvě vteřiny a vytvořit objekty 0,5 vteřiny). Opačný směr, tedy ukládání dat, bylo dlouhé v průměru pouhých 0,3 vteřiny, kde většinu času zabrala práce knihovny `Gson` (měřeno při ukládání dat pomocí synchronní metody `commit()`, nikoliv asynchronní varianty `apply()`).

Výhodou této varianty je jednoduchá implementace. Může být znepříjemněna v případě ukládání objektů, které jsou rozšířením některé abstraktní třídy a jsou obsaženy v kolekci datového typu právě této třídy. Pro jednorázové načítání a ukládání se jedná o velmi rychlý způsob. Problém nastává v případě, pokud kontrolním bodem proběhne větší počet závodníků naráz a časy potřebné k uložení dat se poté téměř sčítají.

Jako druhá varianta bylo provedeno testování rychlosti při práci s databází `SQLite`. Datová struktura je v podobě dvou tabulek: trasa a závodník, kde entita závodník má vazbu na tabulku trasa typu 1:N. Neboli jeden závodník se účastní na jedné trase a opačně jedna trasa může mít na soupisce libovolné množství závodníků. Při stejné startovní soupisce nemá databáze `SQLite` sebemenší problém s načítáním celé soupisky z databáze do objektů. To trvalo přibližně 200 ms. Ukládání dat je zde možné provádět po jednom záznamu (není potřeba přepisovat celý soubor). Tudíž rychlost této operace je prakticky neměřitelná (přibližně 15 ms). Problém nastává při stahování a ukládání celé startovní listiny. Pokud se číslo 0,015 vynásobí počtem závodníků (636), výsledný čas se dostane na hodnotu okolo devíti vteřin. V následující tabulce 1 jsou zobrazeny časy, jak dlouho trvalo aplikaci uložit data při různých režimech přístupu k databázi.

Tabulka 1: Měření rychlosti ukládání startovní listiny při různých přístupech k databázi

Pokus číslo	1	2	3	4	5	6	Průměr
Nová instance databáze [s]	12,7	12	12,8	14,4	14,5	14,8	13,5
Nové otevírání databáze [s]	11,5	9,6	10,7	9,2	7,9	11,7	10,1
Načtení objektu pro práci s databází [s]	7,5	8,7	9,5	7,7	9,2	9,8	8,7
Při zápisu se testuje, zdali ID již neexistuje [s]	10,4	11,4	10,7	10,5	10,6	9,7	10,6

První řádek tabulky představuje volání konstruktoru databáze při každém dotazu. Jak z dat vyplývá, jedná se o velice neefektivní přístup. Druhý řádek je případ, kdy se databáze před každým dotazem otevírá a na jeho konci zase zavírá, avšak její instance je již uložená pomocí návrhového vzoru Singleton. Třetí přístup je situace, kdy instance i objekty databáze pro čtení a zápis jsou uloženy pomocí vzoru Singleton. Databáze se

otevře při prvním voláním a již se tedy nezavírá. V tomto scénáři dosahuje aplikace také prokazatelně nejlepších výsledků. V poslední řadě bylo testováno, o kolik se prodlouží doba ukládání soupisky, pokud se bude u každého ukládaného záznamu nejprve kontrolovat, zdali se již v databázi nenachází.

Tato varianta vyžaduje náročnější implementaci a nutnost znalost jazyka SQL. Programátorovi ale nabídne podstatně rychlejší načítání dat a provádění změn. Není však vhodná pro aplikace, kde dochází k častým změnám velkého množství dat. V následující tabulce 2 je shrnuto, kolik času trvá ukládání a načítání dat při použití `SharedPreferences` anebo `SQLite`.

Tabulka 2: Porovnání rychlostí mezi SharedPreferences a SQLite

	Načítání z pevného uložiště do objektů	Ukládání 636 závodníků	Ukládání jednoho závodníka
SQLite [s]	0,2	8,7	0,015
SharedPreferences [s]	2,5	0,3	0,3

4.5 Jazyk aplikace

Android Studio si velice hlídá, zdali programátor neumisťuje textové řetězce do svého kódu napřímo. U každé takové situace zanechá varování, které vysvětluje, že je optimálnější takové řetězce umisťovat do souboru `res/values/strings.xml`. Tato aplikace doporučení striktně dodržuje. Hlavním důvodem, proč se takto s řetězci nakládá, je následné ulehčení překládání aplikace do nejrůznějších cizích jazyků. Android Studio má pro správu překladů šikovný nástroj, díky kterému je přehledně zobrazen textový řetězec a jeho adekvátní podoby v cizích jazycích. Aplikace má jako výchozí jazyk nastavenou angličtinu. Nicméně pokud je systém Android v telefonu nastaven na češtinu, bude se i aplikace zobrazovat právě v tomto jazyce. Příčinou je vytvoření dalšího `strings.xml` souboru, pro český jazyk ve složce `values-cs`. Android Studio dohlíží také na to, aby se každý textový řetězec nacházel ve všech dostupných překladech. Výjimku tvoří řetězce označené jako `translatable="false"`. Ty aplikace používá například pro definování sledovaných názvů v JSON souboru nebo při výpisu jednotek síly signálu.

Dalším přirozeným požadavkem na práci s textem je spojování několika různých dat z atributů do jednoho textového řetězce. K tomu slouží tzv. „placeholders“ neboli zástupné symboly. Do deklarace řetězce se umisťují ve tvaru „%x$y“, kde „x“ značí

pořadí symbolu a „y“ jeho datový typ (například „s“ pro textový řetězec, „d“ pro číslo). Jak vypadá textový řetězec v souboru `strings.xml`, práce se zástupnými symboly a získání řetězce v libovolné aktivitě je vidět na následující ukázce zdrojového kódu 4.

```
<string name="dBm" translatable="false">Reader: %1$s     
-       %2$s dBm</string>

getString(R.string.dBm, "115", "28")
```

Zdrojový kód 4: Ukázka použití tzv. placeholders v souboru strings.xml a získání řetězce

4.5.1 Třída Enum

`Enum`, neboli výčtový typ, je speciálním typem třídy, který programátorovi umožní nadefinovat vlastní datový typ. Používá se v situacích, kdy je od atributu očekáváno, že bude nabývat pouze omezeného množství hodnot, ale neexistuje primitivní datový typ, který by toto chování popisoval. Nejtypičtějším příklad jsou dny v týdnu nebo měsíce v roce, dále třeba stav objednávky, určení směru a mnohé další. Každá z definovaných hodnot je vnitřně uložená jako celočíselná konstanta. Hodnoty ani jejich číselné reprezentace nelze při běhu programu dále přidávat, měnit ani upravovat. Vše musí být definováno v jednom souboru před spuštěním aplikace. `Enum` slouží jako výrazné zpřehlednění kódu, usnadnění programování a zmenšení pravděpodobnosti výskytu chyb.

Aplikace využívá výčtových datových typů ve dvou případech. První je pro uložení pohlaví závodníka a druhý pro zaznamenání typu závodu. Při definování výčtového typu je potřeba brát v potaz, že aplikace je napsána ve více jazycích a není možné pouze navrátit hodnotu v textové podobě. Základem je u každé výčtové hodnoty uložit číselnou hodnotu (resource ID), která reprezentuje textový řetězec uložený v souboru `strings.xml`. Dále jsou k dispozici dvě možnosti. První spočívá v předávání stavu aplikace (`Contextu`) při každém dotazování na textovou hodnotu. Druhá, která je i použita v aplikaci, je vytvoření statické třídy pro uchování právě již zmíněného `Contextu`. Tuto třídu je třeba nastavit jako název aplikace v souboru `AndroidManifest.xml` a při každém startu aplikace se do ní automaticky uloží základní `Context`, který stačí pro přístup k souboru `strings.xml`. Nyní stačí přepsat základní chování metody `toString()`, aby v ní `Enum` vracel text příslušící nastavenému jazyku v telefonu a programátor nemusel předávat `Context` při každém dotazování na hodnotu výčtového typu.

4.6 Komunikace s webovým serverem

Komunikace se serverem www.sportchallenge.cz probíhá způsobem požadavek – odpověď. Aplikace posílá požadavky, kde v URL specifikuje, o jaké konkrétní informace má zájem či naopak jaké informace serveru poskytuje. Existuje několik scénářů probíhající komunikace: aplikace žádá o celou startovní listinu nebo o její část, která byla změněna až po určitém datu a času; aplikace informuje server o naskenovaném závodníkovi, o jeho diskvalifikaci anebo o jeho zpětném zařazení do závodu.

Při požadavku na získání startovní soupisky se v URL vyskytuje identifikační číslo čtečky, pro kterou chce aplikace sehnat soupisku a nepovinný atribut typu datum a čas, jestli nechce obdržet pouze závodníky, jejichž údaje byly změněny až právě po předané hodnotě. Aplikace obdrží data ve formátu JSON. Na jejich začátku se nachází obecné informace o závodě jako kupříkladu název, datum a prefix (předpona) hodnoty štítků závodníků. Následuje výčet tras a k nim vždy seznam registrovaných závodníků. Každá trasa obsahuje pro aplikaci důležité informace jako startovní čas (pokud se jedná o závod s hromadným startem), typ závodu, počet kol a případně nejnižší očekávaný čas uběhnutí kola. U závodníka je důležité jeho startovní číslo (pro sestavení hodnoty RFID štítku), startovní čas (v případě závodu typu časovka, tedy s postupným startem) a kategorie, do které patří.

Pokud chce aplikace uvědomit server o tom, že na mezičase proběhl závodník, učiní tak, že do URL umístí informaci o identifikačním čísle čtečky, hodnotě štítku závodníka a o jeho čase. V případě diskvalifikace je potom průběžný čas nahrazen informací „DNF“.

Jak již bylo zmíněno v kapitole 4.2.3, aplikace při získávání i odesílání dat kontroluje, zdali při spojení se serverem nedošlo při komunikaci k chybě. Při neúspěšné snaze získat startovní listinu aplikace vytvoří vyskakovací okno k uvědomění uživatele, že se nepodařilo navázat spojení se serverem. Pokud nastane chyba při odesílání dat, vytvoří se notifikace `ConnectionFailedNotification` a s ní se uloží všechny potřebné informace k budoucímu odeslání. Tento typ notifikací se uživatel může kdykoliv pokusit odeslat znovu.

5 Testování aplikace v praxi

Naneštěstí vzhledem k protiepidemickým opatřením kvůli viru Covid-19 byla zrušena drtivá většina sportovních akcí, včetně závodů. Část této práce se měla zabývat testováním aplikace a také schopností čtecího zařízení při skutečném závodě. Do poslední chvíle nebylo jasné, zdali to bude možné či nikoliv. Jako nejzazší možný závod připadal v úvahu liberecký Vánoční Botokros pořádaný přímo na Štědrý den. Bohužel i ten byl na poslední chvíli zrušen a pořádán nakonec nebyl.

Pro testování tedy vznikly značně ztížené podmínky. Nakonec byla zvolena varianta na otevřené louce s provizorními nástroji nesoucími jednotlivé tagy, které představovali samotné sportovce. Nejníže umístěný štítek byl ve výšce 65 cm a naopak nejvyšší ve 125 cm. Tímto tedy byla simulována různá výška závodníků. Testování probíhalo za dodržování veškerých vládních nařízení (mimo improvizované běžce, kteří nedodržovali dvou metrové rozestupy). Jejich rozestavení je možné vidět na následující fotografii 15.

Obrázek 15: Ukázka improvizovaných běžců

5.1 Testování úspěšnosti načtení závodníků

Jelikož nebylo možné třináct „běžců“ uvést do pohybu, bylo do pohybu uvedeno čtecí zařízení, a to v přibližné výšce 120 cm. Simulovanými průběhy kolem štítků z levé a pravé strany bylo nejprve zjištěno, že při těsném kontaktu RFID štítku a kovové tyče opravdu dochází k silnému rušení a štítky se staly při vyšších vzdálenostech prakticky nečitelnými. Došlo tedy k menší úpravě dvou problémových „závodníků“, kde byl štítek oddálen o pár centimetrů od nosné tyče. Výsledky se poté značně zlepšily. Po úpravě se při obou scénářích (vlevo a vpravo) nedařilo načítat vždy pouze ten nejvzdálenější štítek (od čtečky vzdálen přibližně čtyři metry). Výrobce na svých stránkách pro tento konkrétní model čtečky uvádí čtecí vzdálenosti 5,5–7 metrů. Průběhy byly pokaždé testovány celkem třikrát, počty načtených štítků versus jejich celkový počet byl zadán do tabulky 3 uvedené níže.

Tabulka 3: Testování úspěšnosti načítání tagů

Pokus č.	1	2	3
zleva, před úpravou	10/13	11/13	10/13
zprava, před úpravou	11/13	11/13	10/13
zleva, po úpravě	12/13	12/13	12/13
zprava, po úpravě	12/13	12/13	12/13

Z tabulky je patrné zlepšení po úpravě dvou štítků, které byly v těsném kontaktu s kovovou tyčí. Bylo provedeno i měření se čtecím zařízením ve výšce 65 cm, avšak nepřineslo žádné změny v počtu naskenovaných štítků.

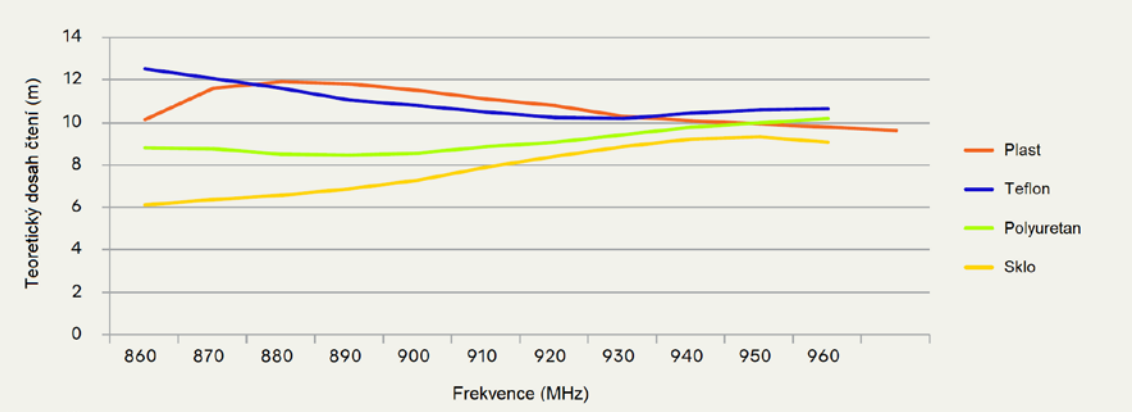
5.2 Testování čtecího dosahu

Dalším testem bylo měření dosahu skenování čtečky. Tentokrát při přímém namíření čtecího zařízení na pouze jeden tag. I tento test probíhal na otevřeném nerušeném prostranství, a to pro dva typy RFID tagu: Dogbone Monza R6 a Dogbone, Monza 4D. Výsledky jsou zobrazeny v následující tabulce.

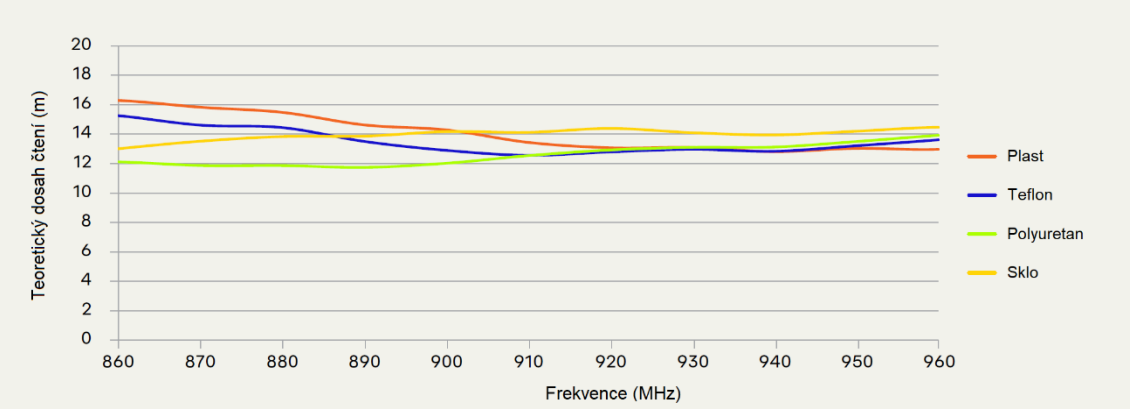
Tabulka 4: Porovnání dvou typů RFID tagů

Vzdálenost	5 m	6 m	6,5 m	7 m	7,5 m
Dogbone, Monza R6	10/10	10/10	10/10	9/10	0/10
Dogbone, Monza 4D	10/10	9/10	0/10	0/10	0/10

Pro každou vzdálenost a každý tag bylo provedeno deset skenování. V tabulce se vždy nachází počet úspěšných naskenování závodníka k celkovému počtu pokusů. Údaje z tabulky jednoznačně sympatizují se štítkem typu Monza R6. Ten dosahoval o jeden metr lepších výsledků. Tento rozdíl v dosahu skenování podporují i oficiální produktové dokumentace těchto štítků [18], [19].

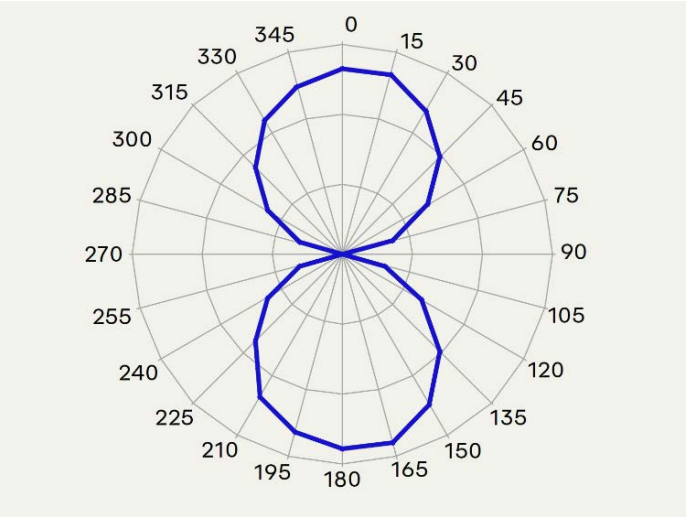


Graf 4: Teoretický dosah čtení tagu Monza 4D [19]



Graf 5: Teoretický dosah čtení tagu Monza R6 [18]

Po testování bylo zjištěno, že štítek Monza 4D byl na startovním čísle nalepen otočený o 180°, avšak dokumentace [19] informuje, že tento způsob rotace tagu nemá na jeho účinnost žádný vliv. Tento fakt je vidět na následujícím obrázku 16.



Obrázek 16: Náchylnost na otočení tagu Monza 4D [18]

5.3 Závěry z testování

Při tomto velmi provizorním testování bylo zjištěno, že použitá čtečka nedisponuje dostatečným výkonem a není tedy vhodná pro závody s předpokladem probíhajících běžců skrz kontrolní stanoviště ve větších skupinkách (závody s hromadným startem). Z výsledků je patrné, že by u takového závodu mohl vzniknout problém s načtením všech probíhajících závodníků. Řešením by mohlo být případné poučení závodníků, že kontrolním stanovištěm smí probíhat seřazeni za sebou či maximálně dva vedle sebe (nebo by pořadatel musel oblast například zřetelně označit či zúžit). Mnoho systémů pro vytvoření kontrolního stanoviště (například od společnosti SPORTident) se již zakládá na potřebě individuálního přístupu ke čtecímu zařízení, kde závodník naskenuje svůj čip, takže by se ani nejednalo o nový limitující faktor. Při skenování RFID čtečkou na vyšší vzdálenosti není již zaručen úspěch. Na druhou stranu při přímém míření na jeden skenovaný tag se testy vyrovnaly hodnotám, které uvádí výrobce.

Během těchto testů byla také odzkoušena ergonomie této sestavy, kdy nebyl použit žádný upevňovací systém pro umístění mobilního telefonu na čtečku. Jednou rukou byl ovládán mobilní telefon, druhou čtečka. Bohužel s dnešními telefony s úhlopříčkami displejů přes 6″ není možné pohodlně pracovat jen jednou rukou. Obsluha aplikace by tedy měla mít připravené řešení, jak vydržet na kontrolním stanovišti několik hodin, aniž by musela neustále třímat v obou rukách nějaké zařízení.

Dále se během testů potvrdilo velké rušení při RFID komunikaci v případě umístění štítku v těsné blízkosti některých kovů. Pro správnou funkčnost načítání je nutné si ověřit typ materiálu (například helmy nebo rámu kola), ke kterému je štítek připevněn. Pokud by se potvrdilo, že materiál je nevyhovující, stačí od něj štítek oddělit například kusem kartonu, aby nedocházelo k přímému kontaktu.

Závěr

Cílem této diplomové práce bylo vytvořit mobilní aplikaci pro operační systém Android, která slouží pro rozšíření stávající časomíry Sportchallenge o možnost jednoduchého skenování závodníků na kontrolním stanovišti.

Práce zprvu seznamuje s technologií RFID. Popisuje princip fungování, jaké existují čtečky a štítky a kde všude je možné se s RFID setkat. Poté představuje použitou mobilní čtečku TSL 1128 a sadu vývojářských nástrojů napsaných v jazyce Java, která je k dispozici ke stažení na oficiálních stránkách výrobce. Slouží pro práci a komunikaci se čtečkou a správu Bluetooth zařízení spárovaných s telefonem.

Aplikace si ze serveru www.sportchallenge.cz obstarává startovní listinu, se kterou následně pracuje. Dále nastavuje čtečce požadované parametry pro způsob skenování závodníků. Aplikace získané hodnoty štítků vyhodnocuje, ukládá do lokálního uložiště a v případě dostupného internetového spojení odesílá informace na server. Pokud hodnota štítku není rozpoznána, závodník se ocitne na kontrolním stanovišti v neočekávaný čas nebo se data nepodaří odeslat na server, je vytvořena příslušná notifikace. V takovém případě se uloží všechny potřebné informace a uživatel se může rozhodnout, jakým způsobem problém později vyřeší.

Aplikace umožňuje procházení informací o závodě, startovní listiny nebo konkrétních závodníků. Jelikož se data mohou i v průběhu závodu stále měnit, soupisku je také možné aktualizovat. Dále je k dispozici funkce vygenerování výsledkové listiny, kde jsou závodníci seřazeni podle naskenovaných časů.

Aplikace byla v rámci práce nejenom naprogramována, ale i v průběhu vyvíjena a zdokonalována. Během vývoje došlo i několikrát k úpravám dle potřeb subjektu Sportchallenge. Kvůli pandemii bohužel nemohlo dojít k odzkoušení při reálném závodě, avšak s funkčností aplikace byl zadavatel spokojen. Namísto skutečného závodu bylo provedeno simulované testování aplikace a čtečky s improvizovanými běžci s následujícími výsledky. Aplikace je stabilní a během testování nebyly zjištěny žádné problémy. U mobilního čtecího zařízení byl shledán nedostačující čtecí výkon a je nutné obsluhou zajistit, aby se při průběhu kontrolním stanovištěm sportovci nevzdalovali od čtečky například vytvořením koridoru.

Pro další rozvoj aplikace je doporučeno přidat autentizaci uživatele na straně serveru. Ačkoliv veškeré startovní listiny závodů jsou veřejně přístupné, a tedy nehrozí

únik citlivých dat, potencionálnímu útočníkovi stačí znát parametry předávané v URL adrese a může kompromitovat výsledky závodu. Dále je možné dvěma způsoby zpřesnit evidovaný čas průběhu závodníka kontrolním stanovištěm. Prvním je získání přesného času pomocí GPS, čehož využívají i profesionální RFID měřící systémy a druhým je možnost upravit aplikaci tak, aby ukládala čas získaný skenováním s nejvyšší hodnotou ukazatele síly signálu (RSSI). K možnému pokračujícímu vývoji aplikace také stojí za zmínku její důkladné otestování v praxi na reálném závodě.

Literatura

[1] 4 Alternatives to Chip Timing. Race Directors HQ [online]. Race Directors HQ, 2019, 1 October 2019 [cit. 2020-11-10]. Dostupné z: https://www.racedirectorshq.com/alternatives-chip-timing

[2] Parkrun [online]. London: parkrun Limited, 2020 [cit. 2020-11-10]. Dostupné z: https://www.parkrun.org.uk

[3] 1128 Bluetooth® UHF RFID Reader. Technology Solutions [online]. Leicestershire: Technology Solutions (UK), © 2020 [cit. 2020-11-12]. Dostupné z: https://www.tsl.com/products/1128-bluetooth-handheld-uhf-rfid-reader

[4] RFID tagy GEN1 a GEN2 - jaké jsou rozdíly? ESP holding [online]. Ústí nad Labem: ESP holding, Copyright 2011-2014 [cit. 2020-11-12]. Dostupné z: https://esp.cz/cs/blog/rfid-tagy-gen1-gen2-rozdily

[5] BERNTSSON, Fredrik. Bluetooth vs RFID in Time Tracking [online]. Karlskrona, Sweden, 2017 [cit. 2020-11-17]. Dostupné z: https://www.diva-portal.org/smash/get/diva2:1117668/FULLTEXT02. Degree project. Blekinge Institute of Technology. Vedoucí práce Emil Alégroth.

[6] Chun-I Sun, Jung-Tang Huang, Shih-Chi Weng a Meng-Fan Chien. City Marathon Active Timing System Using Bluetooth Low Energy Technology [online]. Taipei, Taiwan, 2019 [cit. 2020-11-18]. Dostupné z: https://www.researchgate.net/publication/331290586_City_Marathon_Active_Timing_System_Using_Bluetooth_Low_Energy_Technology. National Taipei University of Technology

[7] RFID – principy, typy, možnosti použití. AUTOMA časopis pro automatizační techniku [online]. Ústí nad Labem: Michal Herštus, 2011 [cit. 2020-11-19]. Dostupné z: https://automa.cz/cz/casopis-clanky/rfid-principy-typy-moznosti-pouziti-2011_07_44083_5207

[8] ABY DO AUTA NETEKLO: JAK PROBÍHÁ TESTOVÁNÍ VODOTĚSNOSTI VOZŮ ŠKODA. ŠKODA Storyboard [online]. ŠKODA AUTO a.s.: Mladá Boleslav, 2020, 6. 10. 2020 [cit. 2020-11-23]. Dostupné z: https://www.skoda-storyboard.com/cs/inovace-cs/technologie-cs/aby-do-auta-neteklo-jak-probiha-testovani-vodotesnosti-vozu-skoda/

[9] Comparing different types of RFID tags. CREATIVE LABEL SOLUTIONS. MODERN TECHNOLOGY. INNOVATIVE THINKING. [online]. Franklin, USA: Resource Label Group, 2018, 12.12.2018 [cit. 2020-11-23]. Dostupné z: https://www.resourcelabel.com/comparing-different-types-of-rfid-tags

[10] Impinj R420 Speedway Revolution, RFID čtečka pro evropské pásmo UHF Gen2, 4 porty. No more searching [online]. Praha: Codeware [cit. 2020-11-23]. Dostupné z: https://www.codeware.cz/items/rfid-a-nfc_15446626/impinj-r420-speedway-revolution-rfid-ctecka-pro-evropske-pasmo-uhf-gen2-4-porty_a_IPJ-R420.html

[11] KOLENA, Matt. Build an RFID Race Timing System. Find Everything You Need For Your Race [online]. RaceDirectorsHQ, 2020 [cit. 2020-11-25]. Dostupné z: https://www.racedirectorshq.com/build-rfid-race-timing-system

[12] Transponder timing. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2020 [cit. 2020-11-26]. Dostupné z: https://en.wikipedia.org/wiki/Transponder_timing

[13] SERRA, Abraham. HISTORY OF RACE TIMING. Building the future of timing [online]. Valencia: © Timingsense, 2015, 24.11.2015 [cit. 2020-11-26]. Dostupné z: https://timingsense.com/en/blog/history-of-race-timing

[14] RACE RESULT SYSTEM Data Sheet. Timing and Scoring of Sports Events [online]. Pfinztal: race result, 2020 [cit. 2020-11-30]. Dostupné z: https://www.raceresult.com/en/documents/RR_System_Datasheet.pdf

[15] Android Studio release notes. Build anything on Android [online]. Kalifornia, USA: Google, 2020 [cit. 2020-12-02]. Dostupné z: https://developer.android.com/studio/releases

[16] GOOGLE. *Android Studio*, 4.4.1 [software]. 5. listopadu 2020 [cit. 2020-08-12]. Dostupné z: https://developer.android.com/studio. Požadavky na systém: Microsoft® Windows® 7/8/10 (64-bit); velikost 896 MB.

[17] App Manifest Overview. Build anything on Android [online]. Kalifornie, USA: Google, 2020 [cit. 2020-12-10]. Dostupné z: https://developer.android.com/guide/topics/manifest/manifest-intro

[18] Datasheet Dogbone Monza R6. RFID and digital ID for everyone [online]. Miamisburg, USA: Avery Dennison, 2020 [cit. 2020-12-28]. Dostupné z:

https://rfid.averydennison.com/content/dam/rfid/en/products/rfid-products/data-sheets/datasheet-Dogbone-Monza-R6.pdf

[19] Datasheet Dogbone Monza 4D. RFID and digital ID for everyone [online]. Miamisburg, USA: Avery Dennison, 2020 [cit. 2020-12-28]. Dostupné z: https://rfid.averydennison.com/content/dam/rfid/en/products/rfid-products/data-sheets/datasheet-Dogbone-Monza-4D.pdf

[20] Impinj-Powered RAIN RFID Handheld Readers. Experience Boundless IoT [online]. Seattle, USA: Impinj, 2012 - 2020 [cit. 2020-12-30]. Dostupné z: https://www.impinj.com/products/partner-handheld-readers

[21] Maximum length of Intent putExtra method? For developers, by developers [online]. waqaslam, 2012 [cit. 2020-12-31]. Dostupné z: https://stackoverflow.com/questions/12496700/maximum-length-of-intent-putextra-method-force-close

[22] NeoTech Software. The Intent extras size limit [online]. NeoTech Software, 2016 [cit. 2020-12-31]. Dostupné z: https://www.neotechsoftware.com/blog/android-intent-size-limit

[23] Creating a SnapshotService persisting Objects using Shared Preferences in Flutter. Computer Science Student in Masters, Frontend Enthusiast [online]. Hannah Schieber, 2020 [cit. 2020-12-31]. Dostupné z: https://hannah-schieber.medium.com/creating-a-snapshotservice-persisting-objects-using-shared-preferences-in-flutter-6dc131b5804e

[24] Shared Preferences - max length of a single value. For developers, by developers [online]. MKJParekh, 2011 [cit. 2020-12-31]. Dostupné z: https://stackoverflow.com/questions/8516812/shared-preferences-max-length-of-a-single-value

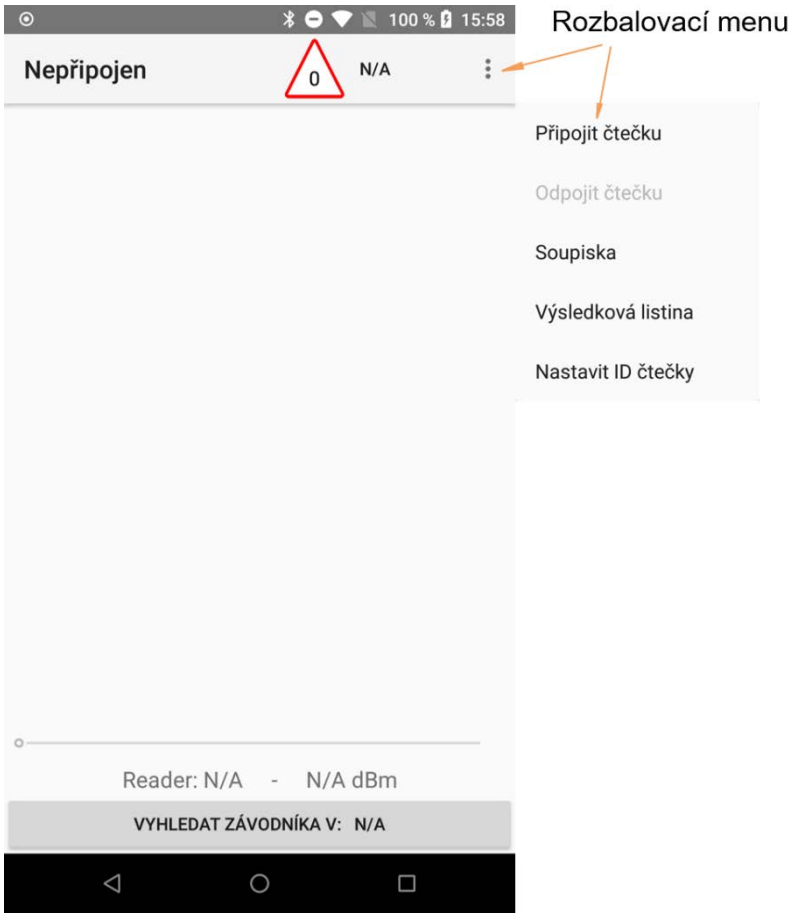
[25] PLÍVA, Z., J. DRÁBKOVÁ, J. KOPRNICKÝ a L. PETRŽÍLKA. Metodika zpracování bakalářských a diplomových prací. 2. upravené vydání. Liberec: Technická univerzita v Liberci, FM, 2014. ISBN 978-80-7494-049-1. Dostupné z: doi:10.15240/tul/002/2014-11-002

[26] H. Wllik, A. Mller and J. Herriger. “Permanent RFID Timing System in a Track and Field Athletic Stadium for Training and Analysing Purposes,” Procedia Engineering, vol. 72, pp. 202-207, 2014.

A Uživatelský manuál

A.1 První spuštění aplikace

Hlavní okno aplikace po prvním spuštění, která nemá připojenou žádnou čtečku, ani staženou žádnou soupisku má svoji podobu zobrazenou na následujícím obrázku 17.

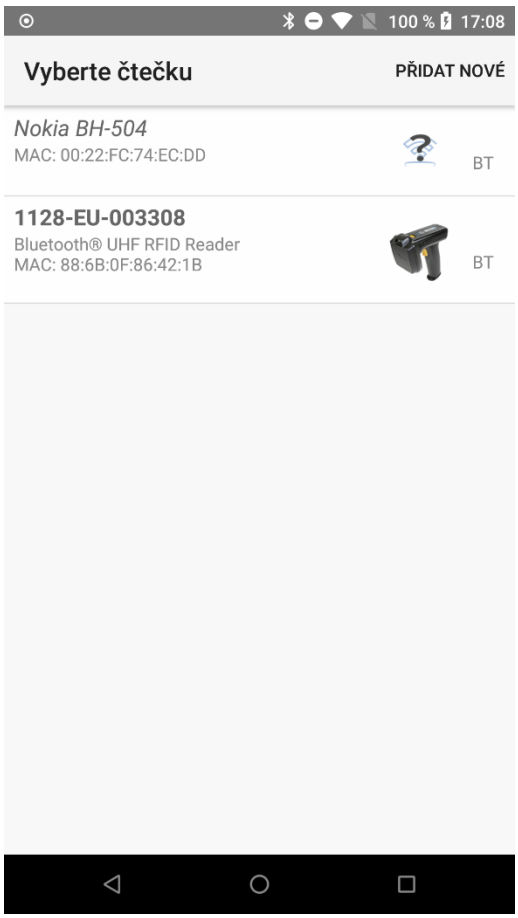


Obrázek 17: Hlavní okno aplikace, bez připojené čtečky, bez stažené soupisky

Prvním důležitým krokem je nastavit aplikaci ID čtečky, které je uživateli aplikace poskytnuto organizátorem závodu. V rozbalovacím menu stisknutím možnosti „**Nastavit ID čtečky**“ vyskočí malé vyskakovací okno a číselná klávesnice pro zadání hodnoty. Po jejím zadání a potvrzení se aktuálně nastavené číslo čtečky zobrazí ve spodní části obrazovky v textu „Reader:“.

A.2 Jak připojit čtečku k aplikaci

V rozbalovacím menu na hlavním okně aplikace se nachází možnost „**Připojit čtečku**“. Po kliknutí se zobrazí následující okno.



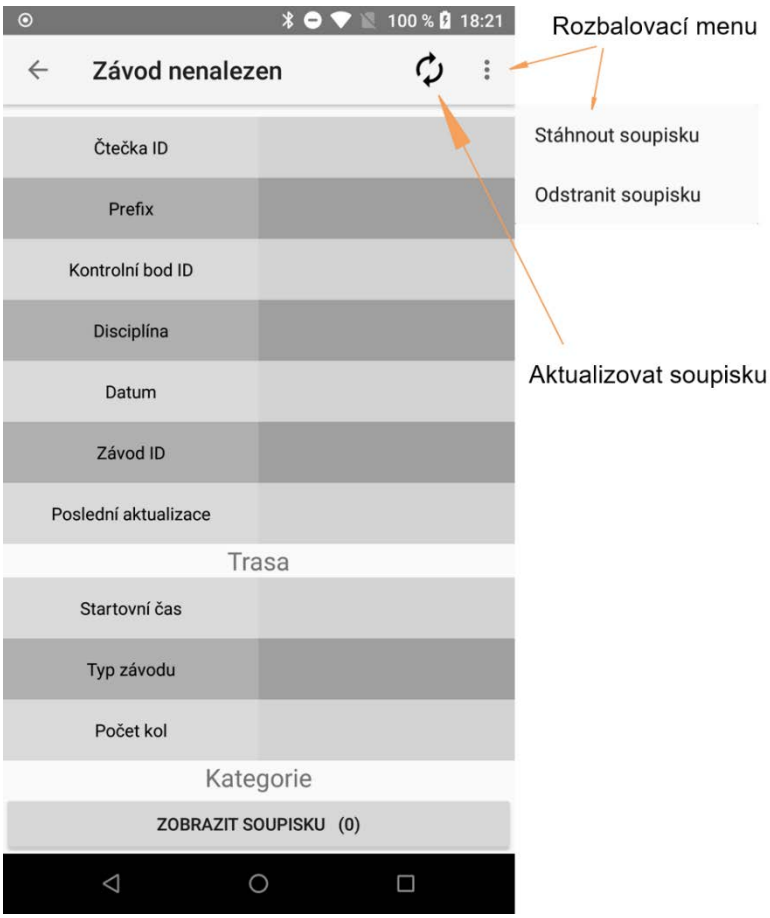
Obrázek 18: Okno pro práci s Bluetooth zařízeními

V případě, že čtečka ještě nebyla spárována s telefonem, je nutné kliknout na tlačítko „**PŘIDAT NOVÉ**“. Nyní se zobrazí rozhraní systému Android (liší se podle výrobce telefonu), kde je nutno čtečku přidat jako jakékoliv jiné Bluetooth zařízení (není nutné, aby se u čtečky objevil nápis „připojeno“, stačí, pokud bude v seznamu spárovaných zařízení). Nyní je nutné se vrátit do aplikace (tlačítkem nebo šipkou zpět), která automaticky tento list zařízení aktualizuje a čtečka by se měla zobrazit.

Pokud se již čtečka nachází v seznamu, stačí ji klepnutím vybrat a v hlavním okně aplikace by se měla zobrazit informace „připojování“ a po úspěšném připojení název připojené čtečky.

A.3 Jak do aplikace nahrát startovní soupisku

V rozbalovacím menu na hlavním okně aplikace se nachází možnost „**Soupiska**“. Po kliknutí se zobrazí následující okno.



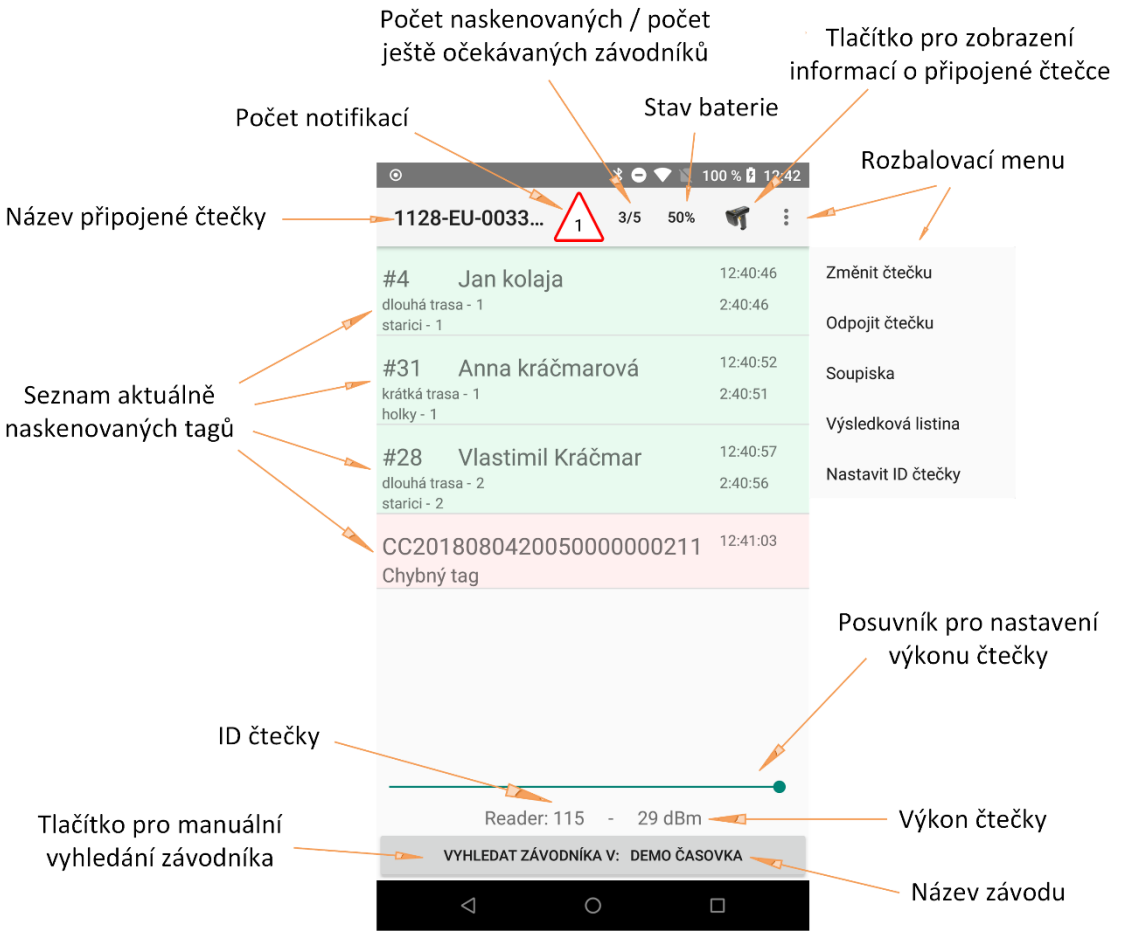
Obrázek 19: Okno pro práci se soupiskou

Z obrázku 19 je patrné, že aplikace nemá uloženou žádnou startovní soupisku. V rozbalovacím menu se nachází možnost „**Stáhnout soupisku**“. Tento proces může zabrat i několik vteřin, pokud závod obsahuje velké množství účastníků nebo je k dispozici pomalé připojení k internetu. Až se stahování dokončí, informace o závodě budou zobrazeny automaticky, není vyžadována žádná interakce od uživatele.

Pokud se soupiska v zařízení již nachází, akce „Stáhnout soupisku“ smaže všechny doposud uložené informace o závodě v aplikaci a nahradí je nově staženými. V tomto případě je vhodné provádět aktualizaci dat pomocí ikony „**Aktualizovat soupisku**“.

A.4 Hlavní okno aplikace

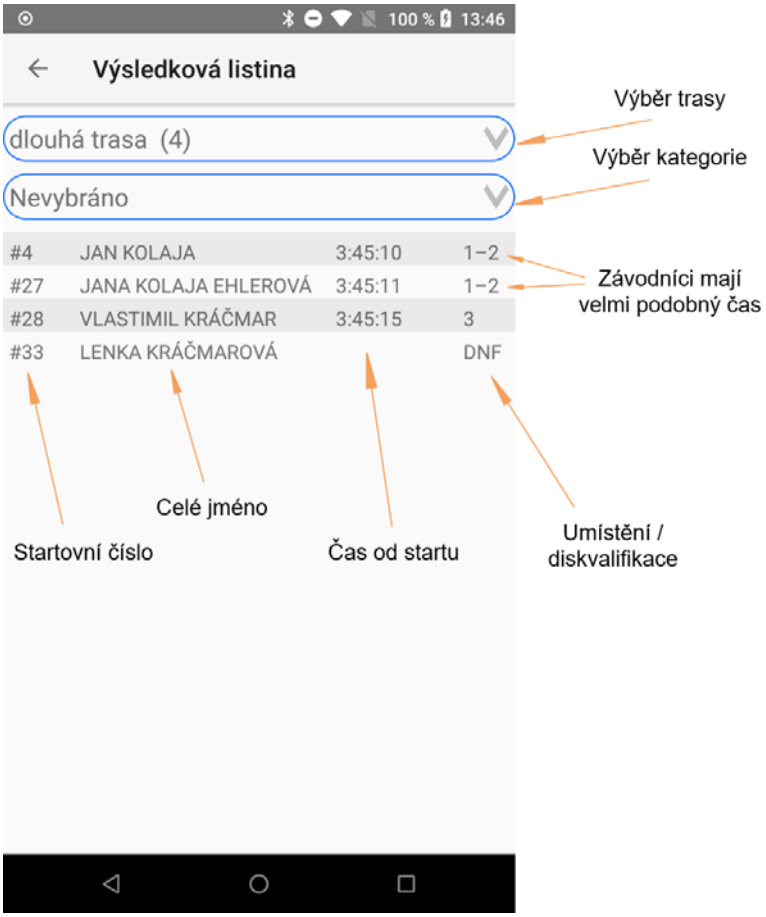
Toto okno obsahující velké množství informací je navrženo tak, aby měl uživatel aplikace všechny důležité informace na jednom místě.



Obrázek 20: Hlavní okno aplikace při skenování závodníků

- Ikona „**Počet notifikací**“ – po stisknutí odkazuje na okno, kde se nachází jejich seznam a možnost práce s jednotlivými oznámeními (viz kapitola A.5).
- Číslo „**Počet naskenovaných závodníků**“ – po stisknutí zobrazí okno, kde budou vypsáni všichni závodníci, kteří ještě neproběhli kontrolním stanovištěm. Ty je možné si v tomto okně seřadit podle čísla, příjmení nebo ročníku anebo si zobrazit jejich detailní informace.
- Tlačítko „**Vyhledat závodníka**“ – po jeho stisknutí se zobrazí jednoduchý formulář a číselník pro zadání startovního čísla a následně po jeho potvrzení i detailní informace o konkrétním závodníkovi.
- Rovněž je možné kliknout na libovolného zeleně podbarveného naskenovaného závodníka pro zobrazení detailních informací o něm.

V rozbalovacím menu se dále nachází možnost „**výsledková listina**“. Ta otevře nové okno aplikace, kde je možné si zobrazit jednotlivé závodníky, jejich čas na kontrolním stanovišti a také jejich umístění. Tyto závodníky je nutné třídit podle trasy, které se účastní a volitelně i podle kategorie, do které spadají.



Obrázek 21: Ukázka výsledkové listiny

A.5 Jak spravovat notifikace

Během závodu může nastat několik situací, které si vyžadují pozornost uživatele. Ten se následně musí rozhodnout, jak daný problém vyřešit.

Obrázek 22: Okno pro správu oznámení

Existuje celkem pět typů oznámení, které aplikace může vytvořit:

- Chybný tag – naskenovaný štítek pravděpodobně nesouvisí se závodem.
- Spojení selhalo – nepodařilo se odeslat informaci na server.
- Závodník již byl naskenován – tento tag se již na kontrolním bodě objevil a závod nemá nastaveno více kol.
- Chybné číslo závodníka – prefix štítku odpovídá danému závodu, avšak startovní číslo nebylo na soupisce nalezeno (pravděpodobně je nutná aktualizace soupisky).
- Závodník naskenován příliš brzy – závodník proběhl kontrolním stanovištěm dříve, než je to fyzicky vůbec možné anebo dříve, než vůbec měl vyběhnout.

Po stisknutí ikony „**aktualizovat notifikace**“ aplikace projde všechny notifikace a zjistí, jestli jsou chybové hlášky stále aktuální a zároveň se pokusí znovu odeslat všechny oznámení, u kterých při předešlých pokusech spojení selhalo.

B Schéma aplikace

Obrázek 23: Schéma aplikace (čtečka: [3], běžec: www.vezuvaute.cz, RFID: www.smart-tec.com)

C Obsah přiloženého CD

- Text diplomové práce
  - diplomova_prace_2021_Tomas_Kosek.pdf
  - diplomova_prace_2021_Tomas_Kosek_arch.pdf
  - diplomova_prace_2021_Tomas_Kosek.docx
  - kopie_zadani_diplomova_prace_2021_Tomas_Kosek.pdf
- Aplikace – instalační soubor
  - aplikace.apk
- Aplikace – projekt v Android Studiu
  - aplikace_projekt.zip