FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Mobilní aplikace Seznamovák
Student:	Michaela Kučerová
Vedoucí:	Ing. Tomáš Nováček
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem práce je vytvořit mobilní aplikaci Seznamovák, která bude sloužit pro potřeby účastníků a organizátorů úvodního soustředění pro nastupující studenty FIT ČVUT.

Proveďte následující kroky:
1) Analyzujte stav aktuální aplikace Seznamovák.
2) Na základě komunikace s účastníky a organizátory akce analyzujte jejich nároky na aplikaci.
3) Naimplementujte nové řešení aplikace Seznamovák v jazyce Kotlin pro operační systém Android.
4) Implementace by měla podporovat i jednoduché vytvoření aplikace Magistrovák se stejnými (či podobnými funkcemi).
5) Proveďte uživatelské testy aplikace.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 24. října 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Mobilní aplikace Seznamovák

Michaela Kučerová

Katedra softwarového inženýrství
Vedoucí práce: Ing. Tomáš Nováček

28. prosince 2020

Poděkování

V první řadě bych chtěla poděkovat vedoucímu mé bakalářské práce, Ing. Tomáši Nováčkovi, za cenné rady, aktivní přístup a pomoc v průběhu tvorby celé práce. Dále bych ráda poděkovala Bc. Marku Kodrovi za rady týkající se implementace Android aplikací, všem účastníkům a organizátorům, kteří vyplnili dotazník týkající se analýzy aplikace, jakožto i těm, kteří se účastnili uživatelského testování. V neposlední řadě mé poděkování patří mým přátelům a rodině za podporu nejen během psaní této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 28. prosince 2020 ……

Odkaz na tuto práci

Kučerová, Michaela. *Mobilní aplikace Seznamovák*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Tato práce se zabývá vývojem mobilní aplikace Seznamovák pro operační systém Android v programovacím jazyce Kotlin. Součástí práce je analýza původní verze aplikace společně se stanovením požadavků kladených na nové řešení, na které navazuje návrh a implementace tohoto řešení. Nakonec se práce zabývá uživatelským testováním aplikace. Výsledkem práce je funkční mobilní aplikace, jež bude sloužit účastníkům a organizátorům seznamovacího kurzu Fakulty informačních technologií ČVUT v Praze.

Klíčová slova mobilní aplikace, přípravný kurz, Seznamovák, Android, Kotlin, notifikace, Firebase

Abstract

The focus of this thesis is the development of the mobile application Seznamovák for Android operating system written in Kotlin programming language. The thesis includes an analysis of the original version of the application together with the definition of the requirements for the new solution, which is followed by the design and implementation of this solution. Finally, the thesis deals with the user testing of the application. The result of this thesis is a functional mobile application that will serve the participants and organizers of the introductory course of the Faculty of Information Technology of the Czech Technical University in Prague.

Keywords mobile application, introductory course, Seznamovák, Android, Kotlin, notification, Firebase

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Nástup na vysokou školu bývá těžký, proto skupina studentů Fakulty informačních technologií ČVUT v Praze pořádá každoročně seznamovací kurz (Seznamovák) pro nově příchozí studenty. Tato několikadenní akce se odehrává v rekreační areálu, kde je pro účastníky připraven bohatý program. Vzhledem k velkému zájmu jsou pořádány dva turnusy, přičemž každý má kapacitu 90 osob. Zorganizovat takto veliké množství lidí však není snadné, proto organizátoři hledali cestu, která by jim tuto práci usnadnila.

Účastníci mívají například problém s včasným chozením na srazy či nevědí, jaký je denní program. Pro zjištění počtu získaných bodů z her musejí docházet do hlavní budovy, kde jsou výsledky napsány, a při mimořádných změnách v programu je obtížné informovat všechny.

Vzhledem k tomu, že mladí lidé často a rádi používají mobilní telefony, a také faktu, že se jedná o budoucí informatiky, rozhodli se organizátoři vyřešit tyto problémy vytvořením mobilní aplikace. V současné době existují mobilní aplikace Seznamovák pro OS (operační systém) Android i iOS s několika základními funkcemi, jako je například zobrazení denního harmonogramu či počtu získaných bodů z jednotlivých her. Přínosem této práce je implementace nové stabilnější verze mobilní aplikace Seznamovák pro OS Android s více funkcemi než měla aplikace původní.

Výsledné řešení bude přínosné pro organizátory, jelikož jim usnadní informování účastníků o změnách, ale především bude důležité pro účastníky, kteří v aplikaci naleznou důležité informace týkající se kurzu a jeho organizace.

Práce je rozdělena na cíle práce, analýzu, návrh, implementaci a testování výsledné aplikace. Kapitola 1 obsahuje popis cílů této práce. Kapitola 2 obsahuje analýzu, v rámci které je zhodnocena původní verze aplikace a na základě zpětné vazby účastníků a organizátorů seznamovacího kurzu jsou zde stanoveny požadavky kladené na aplikaci. Kapitola 3 se zabývá návrhem nového řešení včetně uživatelského rozhraní, stanovení způsobu ukládání dat a volby vhodné architektury. Kapitola 4 popisuje implementaci jednotlivých zajíma-

vých částí kódu společně se zvolenými technologiemi, na niž navazuje kapitola 5 popisující nejen uživatelské testování, jeho přípravu, průběh a výsledky. V závěru je obsaženo zhodnocení práce, zda byly splněny všechny vytyčené cíle práce a také jsou zde rozebrány výhledy do budoucna.

Kapitola 1

Cíl práce

Cílem této bakalářské práce je vytvořit mobilní aplikaci Seznamovák pro operační systém Android v programovacím jazyce Kotlin, která bude sloužit pro potřeby účastníků a organizátorů úvodního soustředění pro nastupující studenty FIT ČVUT v Praze. Mezi dílčí cíle patří analýza stávající verze mobilní aplikace Seznamovák, stanovení nároků kladených na aplikaci na základě komunikace s organizátory a účastníky kurzu, implementace nového řešení a podrobení nově vzniklé aplikace uživatelským testům. Implementace by měla podporovat i jednoduché vytvoření aplikace Magistrovák, určené pro přípravný kurz studentů nastupujících do magisterského studia, se stejnými či podobnými funkcemi.

Kapitola 2

Analýza

Tato kapitola se zabývá především analýzou požadavků kladených na aplikaci, které vycházejí ze zpětné vazby uživatelů původní verze. Nejprve je představen seznamovací kurz, pro nějž aplikace vzniká. Následuje popis původní verze aplikace, na který navazuje sekce týkající se dotazníku, jenž byl rozeslán mezi uživatele této původní verze aplikace. Dále jsou zde zachyceny a popsány jednotlivé požadavky, které vychází z výsledků dotazníku. Poslední sekce kapitoly se zabývá případy užití a jejich mapováním na požadavky.

2.1 Seznamovací kurz Seznamovák

Tato sekce se zabývá popisem seznamovacího kurzu, pro který aplikace Seznamovák vzniká. Jejím cílem je obeznámení čtenáře s důvody, proč je aplikace potřeba, v jakých situacích bude používána a na jakém principu funguje hra, jež bude v rámci aplikace nově implementována.

Seznamovák je akce, která se koná každoročně na přelomu srpna a září. Slouží především k seznámení nových studentů mezi sebou a předání cenných rad od jejich starších a zkušenějších kolegů. Jsou vždy pořádány 2 turnusy, každý s kapacitou 90 osob. V rámci této akce jsou účastníci ubytováni v rekreačním areálu a každý den je pro ně připraven bohatý program, který zahrnuje především přednášky o studiu či o volnočasových aktivitách a hry. Z některých her účastníci získávají body, přičemž na konci celé akce jsou výsledky vyhodnoceny a ti nejúspěšnější získají věcné ceny na památku.

Jednou z organizovaných her je Kolíčkovaná[1]. V rámci této hry každý účastník i organizátor obdrží kolíček se jménem jiného hráče, kterého musí „zabít“. „Zabití“ probíhá připevněním kolíčku na oblečení cíle. Ten, pokud si kolíčku nevšimne do několika vteřin, je v rámci hry „mrtev“ a svému „vrahovi“ předá kolíček se jménem svého cíle. „Vrah“ tímto získá jméno svého

[1] Jiný název této hry je Mafia. V rámci implementace je použit tento druhý název. Jedná se však vždy o stejnou hru.

dalšího cíle a hra pokračuje, dokud jsou „naživu“ alespoň dva lidé. Hra je následně obodována podle počtu „zabitých“ osob. Doposud bylo třeba po každém „zabití“ vyhledat organizátora hry Kolíčkovaná a tuto informaci mu sdělit, což bylo velmi zdlouhavé. Cílem této hry je především seznámení účastníků a zapamatování jmen ostatních hráčů.

V rámci některých her se účastníci potřebují přesouvat z místa na místo v okolí areálu na základě souřadnic, které jim byly dány. Za tímto účelem mohli používat aplikaci map dle vlastních preferencí, avšak častým problémem byla jejich nefunkčnost, jelikož v okolí ubytovacího zařízení není dobré internetové připojení, proto online mapy nejsou dobrou volbou. V rámci původní verze aplikace byla mapa implementována, díky čemuž bylo vyhledávání souřadnic snazší, avšak stále pouze online, což problém s připojením neřešilo.

Jak již tomu občas bývá, může se stát, že nepřízeň počasí či jiné okolnosti donutí organizátory provést změnu programu. Tato situace není neobvyklá, avšak dokud neexistovala aplikace Seznamovák, bylo třeba účastníky o této změně informovat ústně, přičemž se stávalo, že k některým se daná informace nedonesla.

Celkově je seznamovací kurz povedenou akcí, ze které si její účastníci odvážejí spoustu zážitků, užitečných informací týkajících se nejen studia a především spoustu nových přátel. Vzhledem k množství osob, které se jej účastní, je ale také velmi náročný na organizaci. Je třeba připravit a zorganizovat jednotlivé hry, vymyslet celkový program, dohlížet na účastníky a mnoho dalšího. Jedná se o práci vskutku náročnou a jedním z cílů aplikace Seznamovák je usnadnit organizátorům šíření informací a celkově zajistit hladký průběh kurzu.

Nutno zmínit, že existuje také úvodní kurz pro studenty magisterského studia Magistrovák, v rámci něhož je aplikace také zapotřebí. Celá akce se příliš neliší, hlavními rozdíly jsou kratší doba trvání akce, nepřítomnost hry Kolíčkovaná a také nepřítomnost bodování jednotlivých her.

2.2 Původní aplikace Seznamovák

Tato sekce se zabývá analýzou původní verze aplikace, z níž nové řešení vychází. Jsou zde popsány jednotlivé funkce společně s obrazovkami, které je zachycují.

V současné době existuje aplikace Seznamovák, jež byla vytvořena organizátory stejnojmenné akce. Tato aplikace má 4 obrazovky, které představují 4 základní funkce, mezi nimiž se uživatel pohybuje pomocí Bottom Navigation Bar[2] popsaného v sekci 3.3.1.1, a obrazovku Přihlášení, která je zobrazena při prvním spuštění. Při dalším spuštění si aplikace pamatuje přihlášeného uživatele a je rovnou zobrazena obrazovka Harmonogram. Původní aplikace

[2]menu v dolní části obrazovky

je pro OS Android v době psaní této práce dostupná na stránkách Google Play [1].

Aplikace je určena všem účastníkům i organizátorům. Z hlediska používání a dostupných funkcí se aplikace pro obě skupiny uživatelů neliší. Pro organizátory však existuje jiný, obsáhlejší harmonogram. Další rozdíl je v sekci Novinky, kde aplikace pro organizátory zobrazuje data z obou turnusů.

Existuje také aplikace Magistrovák, jež má stejné poslání, avšak je určena pro přípravný kurz studentů nastupujících do magisterského studia na FIT ČVUT v Praze. Tato aplikace je taktéž dostupná na stránkách Google Play [2]. Aplikace je velmi podobná. Liší se pouze nepřítomností obrazovky Body a počtem záložek reprezentujících jednotlivé dny v rámci harmonogramu, jelikož celá akce má kratší dobu trvání.

2.2.1 Přihlášení

Přihlášení do aplikace probíhá pomocí emailové adresy. Uživatel zadá svůj email, kterým se na seznamovací kurz registroval, a po ověření správnosti mu je udělen přístup do aplikace. K přihlášení je třeba internetového připojení vzhledem k nutnosti ověření platnosti emailové adresy. Po dalším spuštění aplikace již zůstává uživatel přihlášen. V případě odhlášení a opětovného přihlášení probíhá daný proces stejně. Obrazovka Přihlášení je zachycena na obrázku 2.1a.

2.2.2 Harmonogram

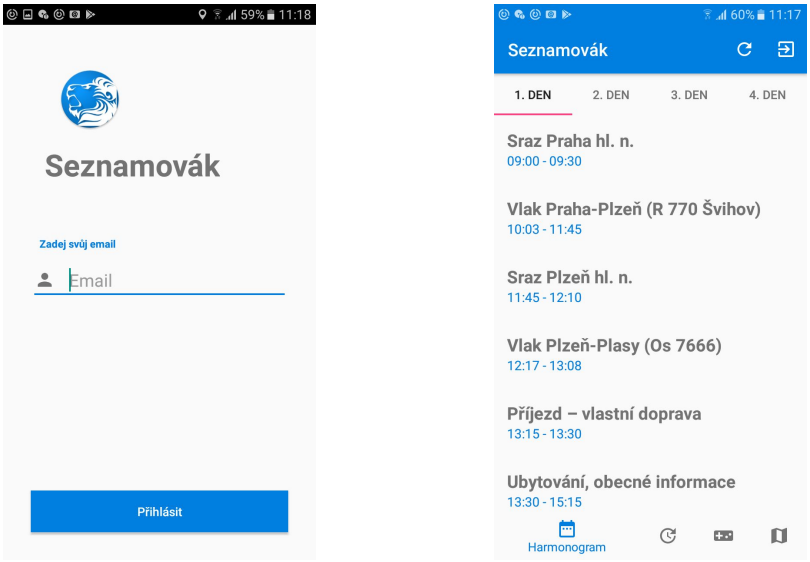
Harmonogram představuje nejčastěji používanou funkci aplikace, proto se jedná o vstupní obrazovku přihlášeného uživatele. Je rozdělen do záložek dle jednotlivých dnů. Každý den představuje seznam položek harmonogramu, přičemž každá položka obsahuje název a čas konání.

Harmonogram se vždy zobrazí na prvním dni, což není optimální z hlediska uživatele. Tento fakt je způsoben implementací, jelikož dny jsou pouze číslovány bez přiřazení ke konkrétnímu datu. Obrazovka Harmonogram je zobrazena na obrázku 2.1b.

2.2.3 Novinky

Sekce Novinky slouží především k zobrazení změn programu kurzu. Bývá využívána, pokud například dojde k posunutí programu či změnám způsobeným nepříznivým počasím. Z hlediska organizátorů je tato funkce velmi důležitá, jelikož se jedná o nejsnazší způsob informování všech účastníků o mimořádných změnách.

Novinky jsou zobrazeny ve stejném formátu jako položky harmonogramu, což napomáhá konzistentnímu vzhledu celé aplikace. Obrazovka Novinky je zachycena obrázkem 2.2a.



(a) Přihlášení (b) Harmonogram

Obrázek 2.1: Vstupní obrazovky aplikace

2.2.4 Body

Jak již bylo zmíněno v sekci 2.1, za jednotlivé hry jsou účastníkům přidělovány body. Seznam her s množstvím bodů, které účastník získal, se nachází právě v této sekci. Jednotlivé položky jsou zobrazeny v pořadí, v jakém se dané hry uskutečnily. Každá položka obsahuje název hry a počet bodů získaných uživatelem. Poslední položkou seznamu je jejich celkový počet.

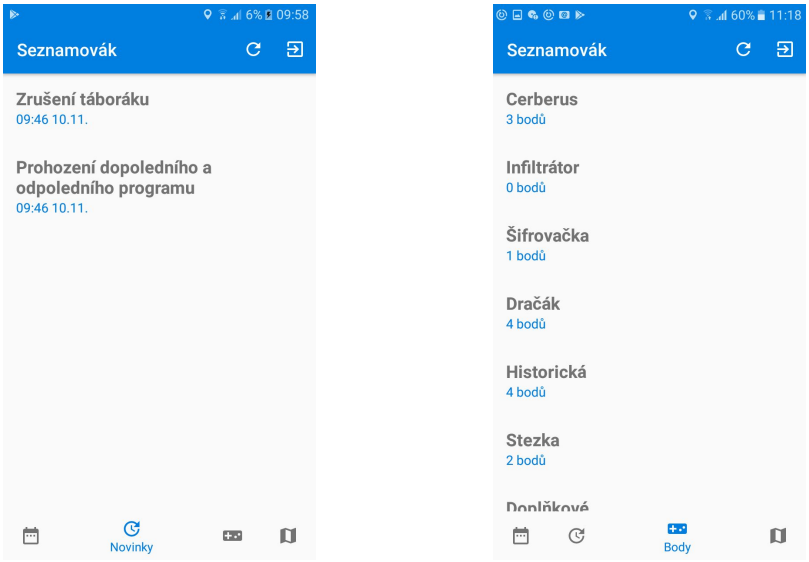
Body jsou opět zobrazeny ve stejném formátu jako novinky a harmonogram. Tato obrazovka je zachycena na obrázku 2.2b, na kterém lze také vidět, že slovo „bod“ není vždy uvedeno ve správném tvaru.

2.2.5 Mapa

Mapa je využívána během hry Chuck a Závěrečné hry. Účastníkům jsou sděleny souřadnice, s jejichž pomocí vyhledají požadované místo na mapě. Cílem je přesunout se na dané místo, kde hra pokračuje.

Tato funkce je dostupná pouze, pokud má uživatel přístup k internetu. Zdrojem dat jsou Mapy Google, které však není možné používat v offline režimu. Vyhledávání souřadnic je realizováno dialogovým oknem. Mapa je zachycena na obrázku 2.3a, dialogové okno na vyhledávání pak na obrázku 2.3b.

Pro úspěšné vyhledávání je potřeba zadat jeden konkrétní formát souřadnic, ačkoliv aplikace postrádá nápovědu, dle které by se uživatel při zadávání řídil. Správně zadané souřadnice jsou zachyceny obrázkem 2.4a, výsledek nesprávného zadání souřadnic zobrazuje obrázek 2.4b. Uživatel je upozorněn na



(a) Novinky (b) Body

Obrázek 2.2: Obrazovky Novinky a Body

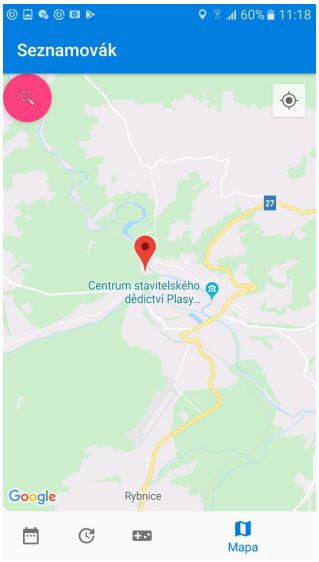
nesprávný formát, avšak nikde není psáno, jaký tento formát má být.

2.2.6 Notifikace

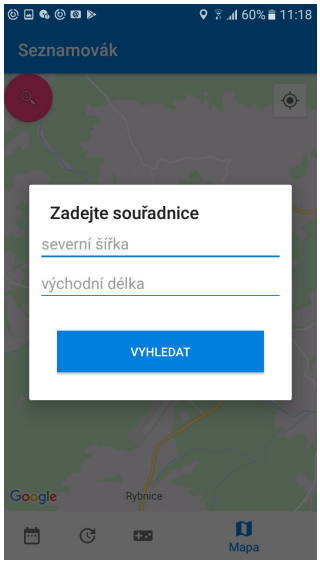
Další funkcionalitou aplikace je zasílání notifikací. Notifikace jsou zaslány pouze, pokud je uživatel připojen k internetu, jelikož jsou data získávána z webové aplikace Seznamovák pomocí služby Firebase Cloud Messaging, popsané v sekci 4.2.1. Uživatel je pomocí notifikací informován o blížící se položce harmonogramu, vytvoření novinky a přidělení bodů. Pokud je zařízení delší dobu bez internetového připojení, dorazí všechny notifikace najednou, včetně notifikací týkajících se již uplynulých položek harmonogramu.

2.2.7 App Bar

App Bar je lišta v horní části obrazovky, jež slouží například k orientaci v aplikaci pomocí zobrazení názvu aktivity, v níž se uživatel nachází, či k navigaci mezi aktivitami pomocí rozbalovacího menu. Více je App Bar popsán v sekci 3.3.1.2. Jedná se tedy o klíčový prvek, který napomáhá konzistentnímu vzhledu aplikace vzhledem k ostatním Android aplikacím [3]. Ve stávající verzi aplikace obsahuje App Bar název aplikace v levém rohu, čímž je porušen hlavní účel této komponenty, kterým je informovat o aktuální obrazovce, a tlačítka aktualizace a odhlášení v rohu pravém. V případě mapy zmíněná tlačítka chybí.

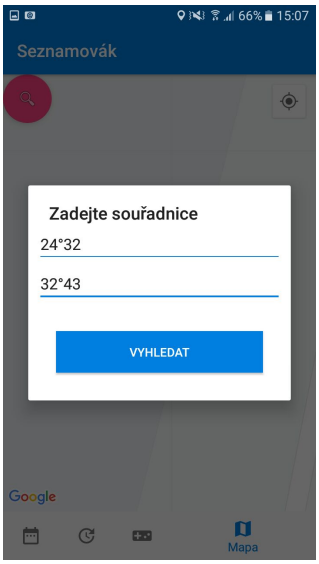


(a) Mapa

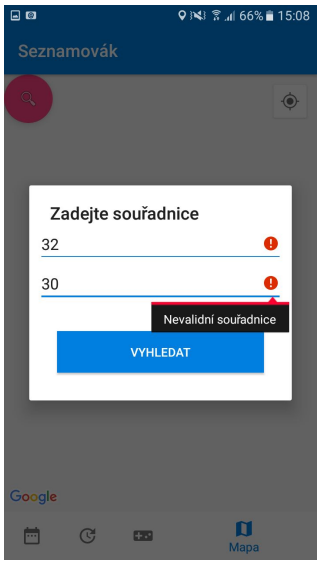


(b) Vyhledávání v mapě

Obrázek 2.3: Mapa existující aplikace



(a) Správný formát



(b) Chybný formát

Obrázek 2.4: Vyhledávání souřadnic

2.3 Dotazník

V rámci analýzy je třeba stanovit požadavky kladené na novou verzi aplikace. Požadavky mají být stanoveny na základě komunikace s účastníky a organizátory akce Seznamovák. Vzhledem k velkému množství těchto osob byl tento požadavek realizován pomocí dotazníku. Dotazník byl vytvořen pomocí Google Forms a krátce po uskutečnění seznamovacího kurzu v roce 2019 byl rozeslán mezi účastníky akce a její organizátory. Z necelých dvou stovek oslovených odpovědělo 106 zúčastněných.

Dotazník byl určen všem účastníkům bez ohledu na OS jejich mobilního zařízení, tedy i všem uživatelům aplikace určené pro OS iOS. Na základě výsledků dotazníku byly stanoveny požadavky kladené na aplikaci, jež byly následně konzultovány s vedoucím práce.

2.3.1 Otázky

Respondenti byli dotazováni na četnost užívání, spokojenost s aplikací a její možné úpravy. Seznam všech otázek:

1. Akce Seznamovák jsem se účastnil/a jako účastník/organizátor.
2. Kolikrát denně jsi používal/a mobilní verzi aplikace Seznamovák? *Pokud jsi aplikaci nepoužíval, pokračuj otázkou č. 3, jinak pokračuj otázkou č. 4.*
3. Proč jsi aplikaci nepoužíval/a?
4. Na kterém operačním systému jsi aplikaci používal/a?
5. Pomohlo ti zasílání notifikací ke včasnému chození na srazy?
6. Ohodnoť složitost ovládání aplikace (1 – zcela intuitivní, 4 – složité).
7. Ohodnoť funkce aplikace (1 – výborná, 4 – nevyhovující).
8. Jak často jsi jednotlivé funkce užíval/a? (1 – velmi často, 4 – vůbec).
9. Existuje funkcionalita, která stávající verzi aplikace chybí? *Pokud ne, pokračuj otázkou č. 11.*
10. Jakou funkci jsi v aplikaci postrádal/a?
11. Vyskytly se při používání aplikace nějaké problémy? *Pokud ne, pokračuj otázkou č. 13.*
12. S jakými konkrétními problémy ses při používání aplikace setkal/a?
13. Jsi spokojen/a se vzhledem aplikace? *Pokud ne, pokračuj otázkou č. 15.*
14. S čím konkrétně nejsi spokojen/a?
15. Ocenil/a bys zaintegrování hry Kolíčkovaná do aplikace?

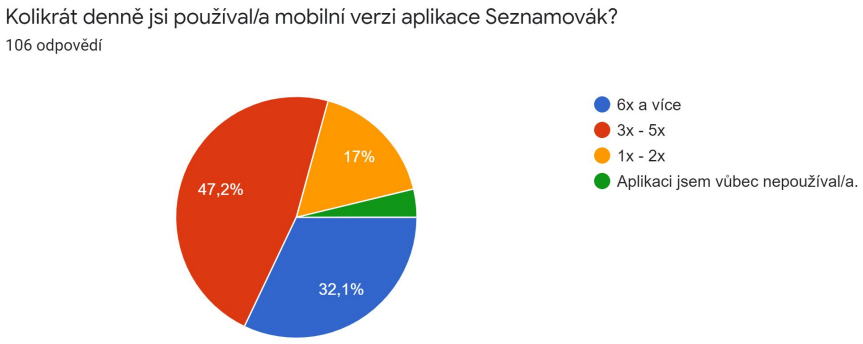
2.3.2 Odpovědi

Dotazník vyplnilo 96 účastníků a 10 organizátorů. Přes 47 % z nich používalo aplikaci 3krát–5krát denně, což ukazuje na přínos této aplikace. Celkový přehled je znázorněn v grafu na obrázku 2.5. Mezi respondenty s OS Android, kterých bylo 73,5 %, použili aplikaci všichni dotázaní.

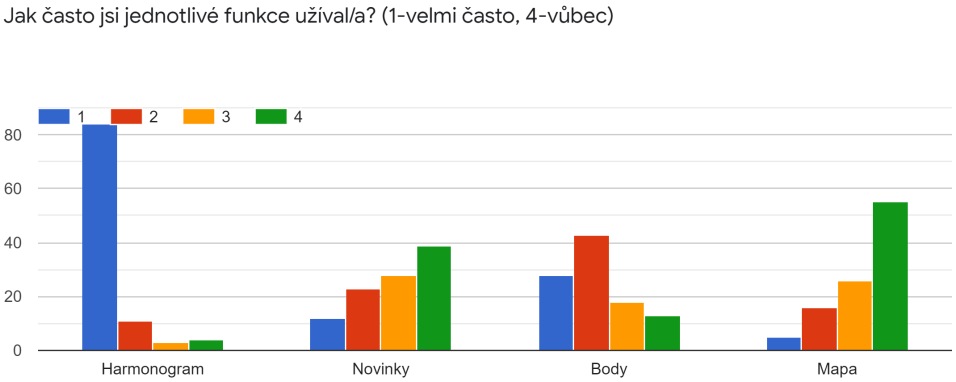
Další otázky se týkaly jednotlivých funkcí aplikace, jejich přínosu a kvality. Četnost užívání jednotlivých funkcí je znázorněna v grafu na obrázku 2.6, hodnocení jednotlivých funkcí pak v grafu na obrázku 2.7.

Bylo rozhodnuto o ponechání všech funkcí i v nové verzi aplikace. 63 respondentů uvedlo, že existuje funkce, kterou v aplikaci postrádají. Jednalo se především o Dark Theme, tedy možnost změny motivu aplikace na tmavý, a otevření harmonogramu na právě probíhajícím dni. Mezi další odpovědi patřily například:

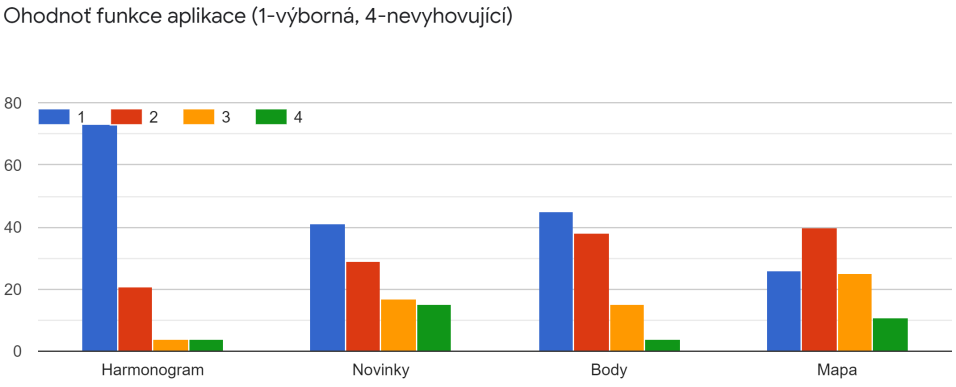
- sloučení aplikací Seznamovák a Zpěvník
- jídelníček
- zobrazení maximálního možného počtu získaných bodů za jednotlivé hry
- posílání notifikací v offline režimu
- offline mapa



Obrázek 2.5: Graf četnosti užívání aplikace



Obrázek 2.6: Graf četnosti užívání jednotlivých funkcí



Obrázek 2.7: Graf hodnocení jednotlivých funkcí

Návrh nové aplikace vychází z požadavků respondentů. Aplikace Seznamovák a Zpěvník jsou vytvořeny organizátory kurzu, přičemž, jak název napovídá, aplikace Zpěvník obsahuje texty písní hraných u večerních táboráků. Aplikace je dostupná na Google Play [4]. Sloučení těchto dvou aplikací je proto vcelku logické a do budoucna realizovatelné, avšak není součástí této práce.

Některé požadavky však není možné uspokojit. Jídelníček není dopředu znám ani organizátorům akce. Maximální počet získaných bodů z jednotlivých her je proměnlivý a pro jednotlivé hráče nedůležitý. Offline notifikace jsou možné pouze v případě harmonogramu, ostatní jsou závislé na změnách dat webové aplikace.

Další otázka se týkala problémů, na které uživatel narazil při používání aplikace a kterých je třeba se vyvarovat při implementaci nové verze. Bezproblémový chod aplikace se týkal pouze 45 dotázaných. Mezi zásadní problém patří především neočekávaný pád aplikace a nefunkčnost mapy včetně neintuitivního způsobu zadávání souřadnic. Dále uživatelé obdrželi notifikaci o přidělení bodů, přičemž se data zobrazená na obrazovce Body neaktualizovala.

V sekci 2.1 byla popsána hra Kolíčkovaná a proces „zabití“, který je nyní zdlouhavý vzhledem k nutnosti najít jednoho konkrétního organizátora mezi stovkou účastníků. Proto by organizátoři uvítali začlenění této hry do mobilní aplikace, přičemž přes 93 % dotázaných bylo stejného názoru.

Kompletní výsledky dotazníku jsou dostupné na přiloženém CD.

2.4 Požadavky

Tato sekce se zabývá požadavky kladenými na aplikaci. Podkladem pro jejich stanovení byla především komunikace s vedoucím práce a výsledky dotazníku. Požadavky obecně dělíme na funkční a nefunkční.

Dle [5] slouží funkční a nefunkční požadavky k návrhu řešení, které obsahuje nezbytné funkcionality v mezích, které jsou dány určitými omezeními. Jednotlivé požadavky by měly být vždy jednoznačně definované a ověřitelné.

2.4.1 Funkční požadavky

Jedná se o požadavky, které se týkají užívání aplikace. Dle [5] představují funkční požadavky pozorovatelné chování, které systém projeví za určitých podmínek a akcí, které systém umožní uživateli provést.

- **F1: Přihlášení**
  Aplikace umožní uživateli přihlášení pomocí emailové adresy, pod kterou se na Seznamovák registroval. K přihlášení je nutné internetové připojení. Přihlášení není zabezpečeno heslem, jelikož aplikace neobsahuje žádná citlivá data.

- **F2: Harmonogram**
  Aplikace umožní zobrazení harmonogramu akce Seznamovák. Harmonogram je rozdělen do jednotlivých dnů. Každá položka harmonogramu obsahuje název a čas konání. Při spuštění bude aplikace otevřena na probíhajícím dni, pokud Seznamovák právě probíhá. V opačném případě bude zobrazen den první.

- **F3: Body**
  Aplikace umožní uživateli zobrazení počtu jím získaných bodů z jednotlivých her. Každý záznam obsahuje název hry a počet získaných bodů. Speciální položkou je počet získaných bodů celkem, který se nachází na konci seznamu.

- **F4: Novinky**
  Aplikace umožní zobrazení seznamu novinek. Jednotlivé položky obsahují její popis a čas vytvoření.

- **F5: Mapa**
  Aplikace umožní uživateli zobrazení jeho aktuální pozice a vyhledání konkrétního místa dle zadaných souřadnic pomocí dialogového okna. Formát souřadnic se bude řídit připravenou šablonou.

- **F6: Notifikace**
  Aplikace bude zasílat notifikace při blížící se položce harmonogramu, vytvoření novinky a přidělení bodů. Notifikace budou získávány z webové aplikace. Jedinou výjimku tvoří harmonogram, jehož notifikace budou vytvářeny na základě dat uložených v zařízení.

- **F7: Kolíčkovaná**
  Aplikace umožní uživateli hrát hru Kolíčkovaná bez nutnosti interakce s organizátorem dané hry.

- **F8: Aktualizace dat**
  Aktualizace dat budou prováděny při spuštění aplikace, pokud má uživatel připojení k internetu, a při zobrazení příchozí notifikace, avšak bude možné i jejich manuální spuštění.

- **F9: Přizpůsobení motivu**
  Aplikace umožní uživateli změnit motiv aplikace. Na výběr bude tmavý a světlý motiv.

- **F10: Odhlášení**
  Aplikace umožní uživateli odhlášení z aplikace. K této akci nebude třeba internetového připojení, avšak k následnému přihlášení bude internetové připojení opět vyžadováno. Po odhlášení budou všechna data týkající se odhlášeného uživatele z mobilní aplikace smazána.

2.4.2 Nefunkční požadavky

Nefunkční požadavky jsou potřebné pro správný chod aplikace a souvisí s omezeními, která jsou na ni kladena. Dle [5] se nefunkční požadavky zaměřují především na výkon, použitelnost, bezpečnost a spolehlivost. Jsou použity pro návrh řešení v rámci omezení, která tyto požadavky přinášejí.

- **N1: Operační systém**
  Aplikace bude vytvořena pro OS Android.
- **N2: Programovací jazyk**
  Aplikace bude napsána v programovacím jazyce Kotlin.
- **N3: Verze**
  Dle [6] je vhodné podporovat cca 90 % všech zařízení. Ke dni 27. 8. 2020 byl tento požadavek splněn při minimální podpoře API 5.1 (Lollipop). Podrobný přehled procentuálního zastoupení všech verzí byl dostupný na stránkách Android Developers [7], avšak v současné době je dostupný v rámci vývojového prostředí Android Studio při tvorbě nového projektu. Tato aplikace se bude řídit požadavkem pro podporu minimálně 90 % všech zařízení.
- **N4: Dostupnost offline**
  Aplikace bude dostupná offline, včetně vyhledávání v mapě do vzdálenosti přibližně 1 km v okolí areálu a zasílání notifikací týkajících se harmonogramu.
- **N5: Rozšiřitelnost**
  Aplikace bude snadno rozšiřitelná a konfigurovatelná pro použití na podobných akcích (např. Magistrovák).
- **N6: Internetové připojení**
  Aplikace bude požadovat přístup k internetu kvůli získávání a aktualizaci dat.
- **N7: Přístup k aktuální poloze**
  Pro zobrazení aktuální pozice bude aplikace požadovat přístup k poloze zařízení.
- **N8: Dostupnost a zisk dat**
  Aplikace bude získávat data z webové aplikace prostřednictvím odpovídajících endpointů. Pro jejich zobrazení bez připojení k internetu budou data perzistentně ukládána do databáze. Data jsou relevantní vzhledem k turnusu, na který je daný uživatel registrován. Číslo turnusu bude získáno z webové aplikace při přihlášení uživatele do aplikace. V případě organizátorů budou ukládána data obou turnusů.

2.5 Případy užití

Tato sekce obsahuje výčet jednotlivých aktérů, následovaný popisem případů užití a tabulkou znázorňující mapování případů užití na požadavky.

Případy užití, neboli Use Cases (UC), dávají představu o funkcích aplikace z hlediska jejího uživatele a slouží k detailnějšímu zachycení funkčních požadavků. Jeden požadavek obvykle obsahuje více UC [8]. Model případu užití se skládá z účastníků a jednotlivých UC. Diagram případů užití je znázorněn na obrázku 2.8.

2.5.1 Aktéři

Aktéři jsou v tomto případě všichni uživatelé aplikace. Uživatel může být nepřihlášený či přihlášený. Aplikace je určena účastníkům kurzu a jeho organizátorům. Většina dostupných funkcí je shodná, avšak organizátorům jsou zobrazena data z obou turnusů, zatímco účastníkům jsou dostupná pouze data turnusu, na nějž se registrovali. Účastníkům jsou navíc přidělovány body, jejichž počet si mohou v aplikaci zobrazit. Tato funkce je pro organizátory irelevantní a proto v jejich případě chybí.

- **Nepřihlášený uživatel**
  Jedná se o uživatele, který získal aplikaci a poprvé ji spustil, či uživatele, který se z aplikace odhlásil. Jeho jediným možným případem užití je přihlášení.

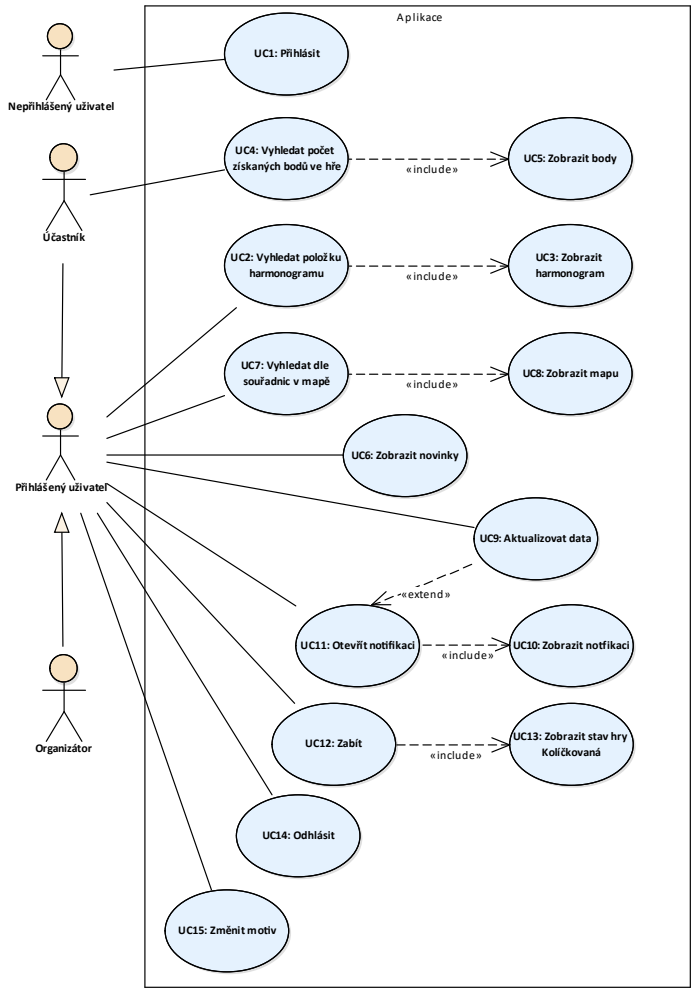
- **Přihlášený uživatel**
  Přihlášení uživatelé se dále dělí na dvě skupiny:

  - **Organizátor**, kterému jsou zobrazena data z obou turnusů, relevantní pro organizátory. Získává notifikace týkající se obou běhů.
  - **Účastník**, jemuž jsou zobrazena data pouze z turnusu, na který je registrovaný. Získává notifikace týkající se pouze daného běhu. Oproti organizátorům je aktérem případů užití týkajících se přidělování bodů.

  Přihlášeným se stane uživatel, který správně vyplní svou emailovou adresu, kterou se registroval na seznamovací kurz, či, v případě organizátora, se jedná o emailovou adresu, kterou používá pro přihlášení do webové aplikace.

2.5.2 Případy užití

Tato sekce detailněji popisuje vybrané případy užití znázorněné diagramem na obrázku 2.8. U složitějších UC jsou uvedeny i scénáře.



Obrázek 2.8: Diagram případů užití

- **UC1: Přihlásit**
  Umožňuje nepřihlášenému uživateli přihlášení do aplikace pomocí emailové adresy, kterou se registroval na seznamovací kurz, či, v případě organizátora, umožňuje přihlášení pomocí emailové adresy, kterou používá k přihlášení do stejnojmenné webové aplikace. K tomuto úkonu je zapotřebí internetového připojení.

- **UC2: Vyhledat položku harmonogramu**
  Umožňuje uživateli zobrazit si harmonogram a procházením vyhledat jeho jednotlivé položky včetně té, jež právě probíhá. Aktuální položka je od ostatních barevně odlišena. Pro snadnější orientaci jsou položky seřazeny dle pořadí, v němž se uskuteční.

- **UC3: Zobrazit harmonogram**
  Uživatel má možnost zobrazit si harmonogram kurzu, který je rozdělen do jednotlivých dnů.

- **UC4: Vyhledat počet získaných bodů ve hře**
  Uživateli je umožněno zobrazit si seznam všech her, které se již uskutečnily, a vyhledat počet bodů, jež v rámci jednotlivých her získal. Tato funkce je dostupná pouze přihlášenému uživateli v roli účastníka.

- **UC5: Zobrazit body**
  Umožňuje uživateli zobrazit si počet bodů z jednotlivých her, které se uskutečnily, a také celkový počet bodů ze všech her. Tato funkce je taktéž dostupná pouze přihlášenému uživateli v roli účastníka.

- **UC6: Zobrazit novinky**
  Uživatel má možnost zobrazit si seznam všech novinek týkajících se turnusu, na který je registrován, či, v případě organizátorů, novinky z obou turnusů.

- **UC7: Vyhledat dle souřadnic v mapě**
  Aplikace umožňuje uživateli vyhledávat konkrétní místa zadáním souřadnic.
  Hlavní scénář:

  1. Uživatel si zobrazí záložku Mapa.
  2. Uživatel otevře dialogové okno pro vyhledání souřadnic.
  3. Uživatel zadá souřadnice.
  4. Aplikace vyhodnotí požadavek.
     (a) V případě, že jsou souřadnice zadány správně a hledané místo se nachází v oblasti přístupné offline mapy, je tento bod zobrazen na mapě.

(b) V případě, že se daný bod nenachází v oblasti dostupné offline mapy, je uživatel o této skutečnosti informován pomocí chybové hlášky.

(c) Pokud je zadán špatný formát souřadnic, je uživateli zobrazena chybová hláška.

Podrobněji je scénář vyhledávání souřadnic zachycen diagramem aktivit v příloze na obrázku B.1.

- **UC8: Zobrazit mapu**
  Aplikace umožňuje uživateli zobrazení offline mapy a aktuální polohy zařízení, pokud uživatel toto oprávnění udělí.

- **UC9: Aktualizovat data**
  Uživatel má možnost manuálně aktualizovat data aplikace. Tato funkce je dostupná pouze v případě, že uživatel má připojení k internetu. Pokud internetové připojení dostupné není, je o této skutečnosti uživatel informován. Uživatel má možnost aktualizovat všechna data aplikace či data týkající se pouze konkrétní domény.

- **UC10: Zobrazit notifikaci**
  Uživatel má možnost zobrazit si přišlé notifikace v Notification Drawer[3].

- **UC11: Otevřít notifikaci**
  Po kliknutí na notifikaci je uživateli aplikace otevřena na obrazovce, které se daná notifikace týká. Notifikace jsou vytvářeny na jednotlivé položky harmonogramu, novinek a bodů z her.

- **UC12: Zabít**
  Umožňuje uživateli „zabít“ jiného hráče (cíl) v rámci hry Kolíčkovaná. Hlavní scénář:

  1. Uživatel si zobrazí záložku hry Kolíčkovaná.
  2. Uživatel klikne na tlačítko „Zabít“.
  3. Aplikace zobrazí uživateli dialogové okno s žádostí o zadání jména následujícího cíle, které uživatel získá od právě „zabitého“.
  4. Uživatel zadá jméno nového cíle.
  5. Uživatel potvrdí dialogové okno.
  6. Nemá-li uživatel připojení k internetu, je o této skutečnosti informován a „zabití“ neproběhlo.
  7. Aplikace vyhodnotí daný požadavek.
     (a) Pokud je zadané jméno správné, je uživateli přiřazen nový cíl.

[3] seznam všech oznámení

(b) Je-li jméno nesprávné je tato informace uživateli sdělena a „zabití“ neproběhlo.

(c) Zadá-li uživatel špatné jméno dvakrát po sobě, nemá následující hodinu možnost „zabíjet“ a je o této skutečnosti informován. Pokusí-li se v této době o opětovné „zabití“, je mu zobrazen počet zbývajících minut, po který zákaz trvá.

Detailněji je scénář „zabíjení“ zachycen diagramem aktivit v příloze na obrázku B.2.

- **UC13: Zobrazit stav hry Kolíčkovaná**
  Umožňuje uživateli zobrazení jeho cíle v rámci hry, pokud je stále „živý“, nebo informace, že je „zabit“, společně se jménem „vraha“. V obou případech je mu zobrazen také počet jím doposud „zabitých“ hráčů.

- **UC14: Odhlásit**
  Aplikace umožňuje uživateli odhlášení z aplikace. Pro opětovné přihlášení je třeba internetového připojení.

- **UC15: Změnit motiv**
  Uživatel má možnost změnit motiv aplikace ze světlého (*light*) na tmavý (*dark*) a naopak.

2.5.3 Mapování případů užití na požadavky

Mapování UC na funkční požadavky slouží ke kontrole, zda došlo ke splnění všech požadavků kladených na aplikaci. Pokrytí jednotlivých funkčních požadavků případy užití je zachyceno tabulkou 2.1.

Tabulka 2.1: Pokrytí požadavků případy užití

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13	UC14	UC15
F1	✓														
F2		✓	✓												
F3				✓	✓										
F4						✓									
F5							✓	✓							
F6									✓	✓	✓				
F7									✓		✓	✓	✓		
F8									✓						
F9															✓
F10														✓	

Kapitola 3

Návrh

Tato kapitola se zabývá návrhem nové verze aplikace, jenž vychází z požadavků na ni kladených. Nejprve se kapitola zabývá výběrem minimální podporované verze OS a volbou programovacího jazyka, v němž bude aplikace napsána. Dále je kapitola rozdělena na návrh uživatelského rozhraní včetně prototypů obrazovek, návrh architektury aplikace a popis získávání a ukládání dat. Uživatelské rozhraní vychází z funkčních požadavků, zatímco architektura je stanovena dle požadavků nefunkčních.

3.1 Verze Android

Při tvorbě mobilní aplikace pro OS Android je třeba stanovit minimální verzi OS, která bude podporována. Její specifikací je zajištěna zpětná kompatibilita a možnost spuštění aplikace na zařízení s danou verzí.

Verze získaly svůj název dle zákusků či sladkostí v abecedním pořadí. Jejich výčet je uveden na obrázku 3.1 společně se zastoupením jednotlivých verzí mezi uživateli a odpovídající verzí API[4]. Dle [9] je ke dni 27.8.2020 minimální verze API pro praktického vývojáře 23, což odpovídá verzi Android 6.0 Marsmallow. Po konzultaci s vedoucím práce však bylo rozhodnuto o minimální podpoře verze OS 5.0 Lollipop odpovídající verzi API 21, což zajišťuje širší uživatelskou základnu a splňuje požadavek, aby aplikace byla dostupná alespoň 90 % uživatelů.

Android 5.0 přinesl významný posun ve tvorbě uživatelského rozhraní[5], jelikož došlo k představení Material Design, což je přehledný styl užitý pro tvorbu přirozenějšího vzhledu aplikace, jehož cílem je zajistit kvalitní uživatelský zážitek[6]. Další významnou změnou bylo nahrazení virtuálního stroje Dalvik novým ART (Android Runtime), který dokáže urychlit spuštění apli-

[4] Application Programming Interface
[5] User Interface (UI)
[6] User Experience (UX)

kace, jelikož umožňuje kompilaci části kódu již při její instalaci, což jeho předchůdce neumožňoval. [10]

3.2 Programovací jazyk

K tvorbě nativních Android aplikací se nejčastěji používají dva programovací jazyky – Java a Kotlin. Dle [12] v dnešní době 60 % vývojářů Android aplikací používá Kotlin. V roce 2017 společnost Google oznámila, že tento jazyk je podporovaným jazykem pro vývoj Android aplikací a o dva roky později dokonce oznámila tzv. *Kotlin-first approach*, což znamená, že se Kotlin stává hlavním podporovaným jazykem pro vývoj Android aplikací, v důsledku čehož byla společně se společností JetBrains založena Kotlin Foundation, která je zodpovědná za vývoj tohoto jazyka. [13]

Mezi přínosy *Kotlin-first approach* patří například přidání Kotlin rozšíření do některých Google knihoven. Tyto knihovny jsou identifikovatelné pomocí zkratky KTX[7]. Dále společnost Google vydala videokurzy na podporu učení vývoje Android aplikací v programovacím jazyce Kotlin dostupných na webových stránkách Udacity [14] a také všechny tutoriály a ukázky vznikají v tomto jazyce. [12]

[7] Kotlin Extensions

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,8%
4.2 Jelly Bean	17	99,2%
4.3 Jelly Bean	18	98,4%
4.4 KitKat	19	98,1%
5.0 Lollipop	21	94,1%
5.1 Lollipop	22	92,3%
6.0 Marshmallow	23	84,9%
7.0 Nougat	24	73,7%
7.1 Nougat	25	66,2%
8.0 Oreo	26	60,8%
8.1 Oreo	27	53,5%
9.0 Pie	28	39,5%
10. Android 10	29	8,2%

Obrázek 3.1: Distribuce jednotlivých verzí Android [11]

Výhody psaní aplikace v jazyce Kotlin dle [15] jsou:

- Snížení množství tzv. boilerplate kódu, což je kód, který se opakuje a snižuje celkovou přehlednost. V jazyce Java je například nutné psát metody *get* a *set*, kdežto v jazyce Kotlin jsou tyto metody generovány kompilátorem automaticky.
- 100% interoperabilita s jazykem Java, která umožňuje v rámci jednoho projektu kombinovat tyto dva jazyky bez omezení. Je tedy čistě na programátorovi, v jaké míře bude daný jazyk zastoupen.
- Bezpečnější kód díky funkcím, které Kotlin obsahuje a které brání programátorovi psát chyby v kódu. Jedná se například o Null Safety, která snižuje počet *NullPointerExceptions*. Ukázka Null Safety je zachycena výpisem kódu 3.1.
- Snadnější práce s asynchronním kódem díky Kotlin Coroutines.

Na základě všech výše zmíněných výhod, které programovací jazyk Kotlin přináší do vývoje Android aplikací, bylo rozhodnuto o užití tohoto jazyka v rámci implementace nové verze aplikace Seznamovák.

3.3 Uživatelské rozhraní

Tato sekce se zabývá správným návrhem UI dle konceptu Material Design, který byl představen společností Google společně s Android verzí 5.0. Jsou zde rozebrány jednotlivé komponenty UI. Následuje sekce s návrhy a popisem jednotlivých obrazovek, v rámci nichž jsou dané komponenty použity.

3.3.1 Material Design

Material Design je systém návrhu nejen Android aplikací vytvořený společností Google, jehož cílem je tvorba konzistentního vzhledu a především kvalitního uživatelského zážitku. Zde jsou představeny jednotlivé komponenty Material Design[8], které byly během návrhu použity.

[8] Material Design Components (MDC)

```
var a: String = "hello"
a = null // compilation error

var b: String? = "world"
b = null // ok
```

Výpis kódu 3.1: Ukázka Null Safety

3.3.1.1 Bottom Navigation Bar

Jedná se o lištu v dolní části obrazovky, která slouží pro navigaci mezi hlavními destinacemi aplikace. Bottom Navigation Bar by měl dle [16] obsahovat tři až pět primárních destinací, což jsou v případě aplikace Seznamovák obrazovky Program, Novinky, Body, Kolíčkovaná a Mapa. Každá položka je reprezentována ikonou a aktuální destinace navíc textovým popisem.

3.3.1.2 App Bar

Další komponentou spojovanou s navigací v rámci aplikace je App Bar, lišta v horní části obrazovky. Zde se obvykle nachází název aktuální destinace pro snadnější orientaci a tlačítka umožňující akce s destinací spojované. App Bar dále může obsahovat navigační ikonu či všeobecné menu[9], reprezentované ikonou se třemi tečkami, které obsahuje akce, jež se nevejdou na lištu či jsou takto úmyslně schované.

3.3.1.3 Floating Action Button

Floating Action Button (FAB), neboli plovoucí tlačítko, dle [17] představuje primární akci, kterou uživatel může vykonat v rámci dané obrazovky. Jedná se obvykle o kulaté tlačítko s ikonou reprezentující daný úkon. V rámci aplikace Seznamovák je FAB použito při návrhu obrazovky reprezentující mapu a slouží k zobrazení dialogu pro vyhledávání souřadnic.

3.3.1.4 Dialog

Dialogy slouží dle [18] k poskytnutí informací uživateli či po něm vyžadují provedení další akce, jako je v případě této aplikace například vyhledání souřadnic. Dialogy mají obvykle několik částí, mezi které se řadí název, tělo dialogu a tlačítka pro potvrzení akce či zavření dialogového okna. V rámci aplikace Seznamovák jsou dialogy použity v návrzích obrazovek, které umožňují vyhledávání souřadnic a „zabíjení“ v rámci hry Kolíčkovaná.

3.3.1.5 Tabs

Tabs, neboli záložky, slouží pro navigaci mezi položkami, které jsou na stejné hierarchické úrovni. V tomto případě jednotlivé záložky reprezentují harmonogram odpovídajícího dne. Jedná se o lištu v horní části obrazovky, přičemž jednotlivé položky mohou obsahovat text, ikonu či oba tyto prvky.

[9] Overflow Menu

3.3.1.6 Notifikace

Notifikace slouží k informování uživatele o určité události aplikace i ve chvíli, kdy není používána. Text notifikace bývá krátký a výstižný. Notifikace má 3 části. Hlavičku, obsah a část akcí.

Hlavička notifikace obsahuje ikonu, název aplikace a volitelně také čas přijetí. Obsah notifikace se skládá z názvu, který by měl být výstižný a krátký. Více podrobností může být obsaženo v textu notifikace. Pokud notifikace upozorňuje například na přijetí zprávy od jiné osoby, může zde být také větší ikona, jež odesílatele reprezentuje. Poslední část může obsahovat až 3 akce reprezentované textem či ikonou. Tyto akce například dovolí uživateli ihned odpovědět na zprávu či ji archivovat.

Po otevření notifikace by uživateli měla být zobrazena vždy destinace aplikace, jež s ní přímo souvisí. V aplikaci Seznamovák se například jedná o obrazovku Novinky po přijetí odpovídající notifikace. Notifikacím lze přiřadit prioritu, která definuje chování, jako je například zvuk či zda se notifikace zobrazí na obrazovce.

Od Android verze Oreo je nutné definovat notifikační kanály[10], aby byly notifikace doručeny. Jejich výhodou je rozdělení notifikací do více skupin, přičemž uživatel má možnost povolit či zamítnout přijímání notifikací pro každý kanál zvlášť. Předchozí verze OS tuto možnost nenabízejí a nastavení se vždy týká všech notifikací aplikace. [19]

3.3.2 Wireframes

Tato část obsahuje návrhy obrazovek aplikace, neboli wireframes. Každá obrazovka představuje určitou funkci aplikace k realizaci stanovených funkčních požadavků. Oproti původní verzi byla přidána obrazovka pro hru Kolíčkovaná. Aplikace se tedy skládá z pěti hlavních obrazovek, mezi nimiž se uživatel pohybuje pomocí lišty Bottom Navigation Bar popsané v sekci 3.3.1.1, a obrazovky Přihlášení. V případě, že je přihlášený uživatel organizátorem, nemá k dispozici obrazovku Body, jelikož ta je určena účastníkům. V původní verzi aplikace byla tato obrazovka v případě organizátorů vyplněna tzv. dummy hodnotami[11]. Návrhy všech obrazovek jsou dostupné na přiloženém CD. Pro jejich tvorbu byl použit webový nástroj Proto.io dostupný na [20], který umožňuje tvorbu nejen mobilních prototypů.

Návrhy obrazovek Přihlášení, Program, Novinky a Body se od původní aplikace příliš neliší. Důvodem je především jednoduchost a přehlednost. Jejich návrhy jsou zobrazeny v příloze na obrazcích B.3 a B.4.

[10] Notification Channels

[11] Jedná se o nepravé hodnoty pouze za účelem vyplnění požadovaných polí.

3.3.2.1 Mapa

Obrazovka reprezentující mapu nyní obsahuje odsazené FAB, jehož funkce je popsána v sekci 3.3.1.3. Po stisknutí tlačítka je uživateli zobrazeno dialogové okno s možností vyplnit hledané souřadnice. Formát souřadnic se řídí připravenou šablonou. Odpovídající obrazovky se nachází v příloze na obrazcích B.5a a B.5b.

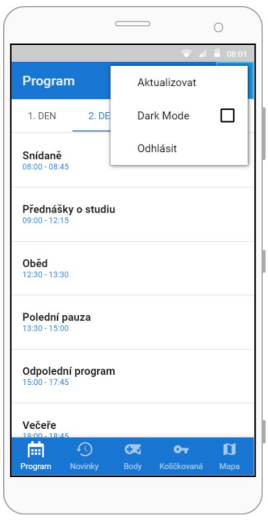
3.3.2.2 App Bar

App Bar obsahuje název obrazovky, na níž se uživatel nachází, společně s Overflow Menu. Návrh obrazovek obsahuje v Overflow Menu akce pro změnu módu, aktualizaci dat a odhlášení z aplikace. Změna módu je realizována zaškrtávacím tlačítkem[12]. Menu je dostupné ze všech obrazovek aplikace kromě obrazovky Přihlášení, která neobsahuje celý App Bar. Obsah Overflow Menu je zachycen obrázkem 3.2.

3.3.2.3 Kolíčkovaná

Zcela nová je obrazovka pro hru Kolíčkovaná. Obrazovka se mění dle stavu hry, který je zde zachycen vždy společně s počtem doposud „zabitých“ hráčů. Pokud je uživatel v rámci hry „naživu“, obsahuje obrazovka jméno cíle a tlačítko „Zabít“. Stisknutí tlačítka vyvolá zobrazení dialogového okna, které žádá o vyplnění jména následujícího cíle dle šablony. Pokud byl uživatel v rámci

[12] z anglického checkbox



Obrázek 3.2: Návrh obsahu Overflow Menu

hry „zabit“, obsahuje obrazovka jméno „vraha“ namísto cíle a tlačítko „Zabít“ zde obsaženo není. Návrhy obrazovek hry jsou zachyceny obrázky 3.3.

3.4 Architecture Components

Architecture Components (AC) je sada knihoven z Android Jetpack, což je sada nástrojů a knihoven, které slouží ke tvorbě moderních Android aplikací a jež jsou zpětně kompatibilní. Obecně ji lze rozdělit do 4 kategorií, které jsou zachyceny na obrázku 3.4.

V této sekci jsou popsány jednotlivé AC, které slouží ke tvorbě testovatelných, rozšiřitelných a udržovatelných aplikací z hlediska architektury a komunikace mezi jednotlivými vrstvami a jež byly následně použity v rámci implementace aplikace Seznamovák. Tato sekce vychází z oficiální dokumentace AC od společnosti Google [22].

3.4.1 ViewModel

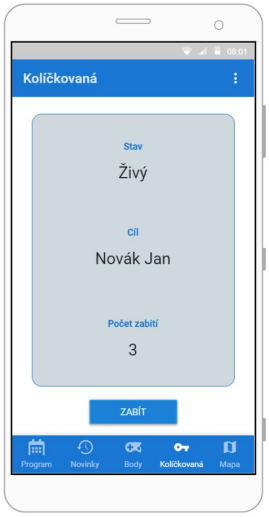
ViewModel je komponentou obstarávající business logiku pro komunikaci s Modelem. Má vlastní životní cyklus, který navíc není ovlivněn konfiguračními změnami, proto během nich nedochází ke ztrátě dat, která jsou uživateli zobrazena v rámci Fragmentů a Activit. ViewModel poskytuje data, ke kterým se jednotlivá Views váží.

3.4.2 LiveData

LiveData se řadí mezi observable AC, což znamená, že změny jejich dat mohou být pozorovány ostatními komponentami bez nutnosti vytvoření pevné vazby a při změně dat dochází k propagování změny na straně pozorovatele. LiveData navíc respektují životní cyklus komponent typu Activity či Fragment a jelikož obsahují logiku pro úklid v případě jejich zničení, nedochází k plýtvání paměti.

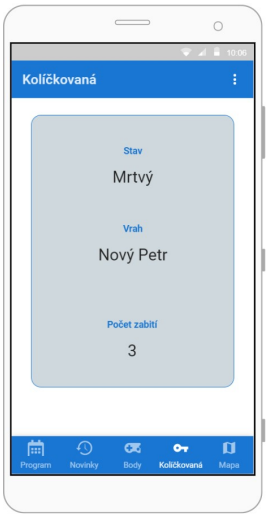
3.4.3 Room

Room představuje SQLite knihovnu, která zajišťuje perzistentní ukládání dat. Dle [23] použití Room předchází psaní boilerplate kódu a v rámci dotazů nad databází navíc provádí jejich kontrolu již během kompilace, čímž předchází odhalení chyb až za běhu aplikace. Návratovou hodnotou těchto dotazů navíc mohou být pozorovatelná LiveData. Room zavádí omezení na přístup k databázi z hlavního vlákna, který by mohl vyústit v horší uživatelský zážitek způsobený „zamrznutím“ aplikace během této doby. K definování jednotlivých částí úložiště a přístupu k nim jsou použity anotace.



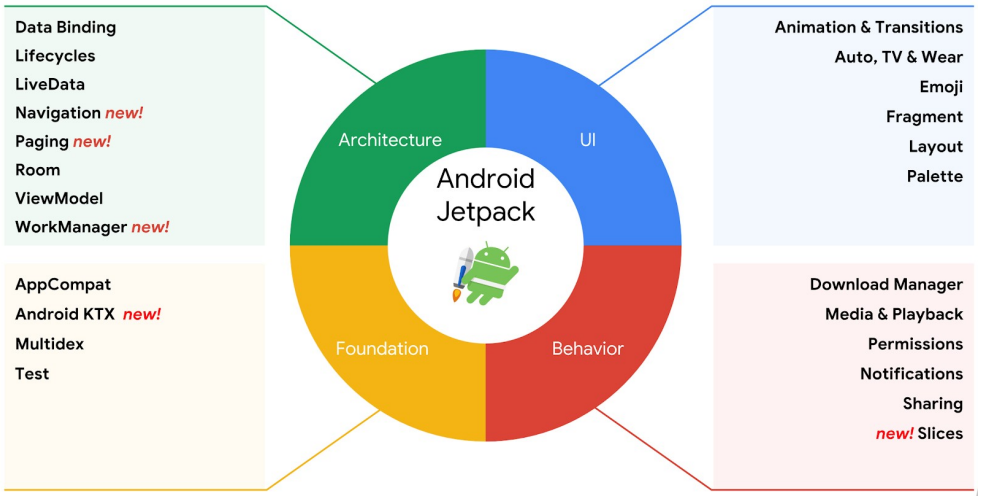
(a) Stav Živý

(b) Zabití



(c) Stav Mrtvý

Obrázek 3.3: Návrh obrazovek hry Kolíčkovaná



Obrázek 3.4: Android Jetpack komponenty [21]

3.4.4 Binding komponenty

Mezi Binding komponenty se řadí View Binding a Data Binding. I přes podobný název se jedná o dvě odlišné komponenty z hlediska použití.

View Binding slouží primárně k odstranění boilerplate kódu z Aktivit a Fragmentů, jelikož nahrazuje nutnost volání metody *findViewById* vygenerovanými binding objekty. Dle [24] má tento přístup výhody z důvodu zabránění vzniku *NullPointerException* a *ClassCastException*, jelikož dochází k odhalení nekonzistencí mezi kódem a XML layouty během kompilace.

Výhody oproti Data Binding jsou především rychlejší kompilace a snadnější použití, jelikož není třeba vytvářet speciální tag v rámci XML souboru, ale binding třída je vytvořena pro všechny tyto layout soubory automaticky.

Data Binding je mocnější nástroj nabízející pokročilejší funkce než View Binding. Slouží jako tzv. binder[13] mezi UI a komponentou ViewModel, jelikož umožňuje deklarovat obsah UI komponent přímo v XML layout souborech pomocí speciální syntaxe, čímž dochází k odstranění přebytečného kódu z Aktivit a Fragmentů. Ty se tak stávají přehlednější a lépe udržovatelné, přičemž tento postup nahrazení *findViewById* má stejné výhody jako v případě výše

[13] prostředník, který zajišťuje vazbu mezi komponentami

zmíněného View Binding. Další výhodou použití této knihovny je poskytnutí metod, které reagují na změny dat a automaticky aktualizují UI. Data Binding existuje také v podobě Two-way Data Binding, což znamená, že je vazba obousměrná a lze reagovat na uživatelovy akce, jako je například zadání textu. [25]

V rámci jednoho projektu je možné použít zároveň obě binding komponenty. Pokud XML layout soubor obsahuje tag layout, je použit Data Binding. V opačném případě je pro soubor automaticky vygenerována instance odpovídající třídy View Binding. V případě, že je třeba pouze nahradit *findViewById*, používá se odlehčenější View Binding. Pokud jsou zapotřebí pokročilejší funkce, jako například Two-way Data Binding, je použita knihovna Data Binding.

Možnou alternativou k těmto komponentám je použití Kotlin Synthetics, které také odstraňují potřebu psaní *findViewById*. Tento postup však není doporučovaný společností Google, jelikož nepřináší Null Safety a typovou bezpečnost, a proto v rámci této práce byly použity doporučované binding komponenty. [26]

3.4.5 Navigation

Další z rodiny AC je Navigation komponenta, která slouží pro navigování v rámci aplikace. Dle [27] má tři základní části:

- **NavGraph** je XML soubor, který obsahuje všechny destinace a možnosti navigace mezi nimi. V rámci Android Studia navíc existuje Navigation Editor, který umožňuje vizualizovat celý graf.
- **NavController**, který slouží k samotné navigaci a může být volán z kódu s definováním cílové destinace či akce, jež má být provedena.
- **NavHostFragment** je kontejner, v rámci kterého jsou vyměňovány jednotlivé Fragmenty, mezi nimiž se uživatel naviguje.

Navigation komponenta je sada knihoven a nástrojů, která slouží například pro správnou práci s pamětí během pohybu v aplikaci, a pokud je zapotřebí použití Navigation Drawer či Bottom Navigation Bar, tato komponenta tuto práci usnadňuje. Například v případě použití Bottom Navigation Bar obstarává zvýraznění správného tlačítka, práci se zásobníkem[14] a synchronizaci s komponentou App Bar.

3.4.6 WorkManager

Tato komponenta obstarává především asynchronní práci na pozadí. Work Manager může vykonávat svou práci, přestože aplikace právě neběží. Této komponentě lze nastavit, za jakých podmínek má práci vykonat. Jedná-li se

[14] Back Stack

například o stažení dat, lze stanovit, že tak má být učiněno pouze v případě připojení přes Wi-Fi apod. Lze také určit, zda má být práce vykonána jednou či opakovaně, popřípadě v jakých časových intervalech.

3.5 Základní Android komponenty

Tato sekce obsahuje popis komponent, jež tvoří základ všech Android aplikací. Jedná se o 4 komponenty, z nichž každá má určitou specifickou funkci. Jsou reprezentovány stejnojmennými třídami a jejich názvy jsou:

- Activities (Aktivity)
- Broadcast Receivers (Přijímače vysílání)
- Content Providers (Poskytovatelé obsahu)
- Services (Služby)

V této části jsou jednotlivé komponenty představeny a jsou zde popsány jejich funkce a případy, v nichž mohou být použity.

3.5.1 Activity

Aktivita představuje základní prvek UI. Jedná se o jednu obrazovku, která je uživateli zobrazena a skrze niž může s danou aplikací interagovat. V současné době bývají Aktivity používány především jako kontejnery, v rámci kterých jsou zobrazovány jednotlivé Fragmenty. K tomu dochází například při použití komponenty Bottom Navigation Bar popsané v sekci 3.3.1.1. Aktivita je také jediná z těchto čtyř komponent, která má uživatelské rozhraní.

Hlavním problémem UI komponent je správa jejich životního cyklu, který je pod kontrolou systému. Během rotace zařízení dochází ke zničení a opětovnému vytvoření daných objektů, což vede ke ztrátě jejich stavu. Proto jsou obvykle data pro UI komponenty získávána z ViewModel tříd, které jsou na daném View nezávislé, a nejsou tedy ovlivněny konfiguračními změnami.

3.5.2 Broadcast Receiver

Broadcast Receiver slouží k zachycení určité změny či události v rámci zařízení. Jedná se o komponentu běžící na pozadí, která nevyžaduje běh aplikace. Pomocí tohoto přijímače může být například vytvořena notifikace na nadcházející událost. V takovém případě je v aplikaci naplánován alarm na určitý čas. V tento čas je poté vyslán záměr[15], jež Broadcast Receiver zachytí a zpracuje na základě akce definované v tomto přijatém záměru. Broadcast Receiver může být dále využit například pro zahájení stahování dat, pokud událostí, na niž reaguje, je stav internetového připojení.

[15] Intent

3.5.3 Content Provider

Content Providers, neboli poskytovatelé obsahu, slouží ke sdílení dat a práci s nimi napříč více aplikacemi. Jedná se o prostředníky mezi daty a aplikacemi, které k nim chtějí přistupovat. Content Provider poskytuje API pro práci s daty, přičemž jejich skutečný zdroj lze měnit.

3.5.4 Service

Service, neboli služba, je komponenta běžící na pozadí. Slouží k běhu procesů s delší dobou trvání, jako je například stahování dat z internetu. Tyto služby mohou běžet paralelně s UI vláknem, tedy nebrání uživateli aplikaci používat i během náročnějších operací. Dalším příkladem užití této komponenty může být spuštění hudebního přehrávače, kdy hudba zůstává hrát, i přestože se uživatel přesměruje pryč z dané aplikace.

3.5.5 Android Manifest

Android Manifest je XML soubor, který obsahuje základní informace o aplikaci. Nachází se zde například seznam všech oprávnění, která aplikace požaduje, či jsou zde specifikovány ikona a název aplikace. Především jsou však v tomto souboru definovány všechny komponenty výše zmíněných typů společně s jejich základními informacemi. Nejsou-li jednotlivé komponenty zapsány v Android Manifest souboru, systém o jejich existenci neví, a tedy nemohou být použity. [28]

3.6 Architektura

Tato sekce se zabývá volbou architektury pro následnou implementaci. Je zde představena doporučovaná architektura Android aplikací od společnosti Google. Dále jsou zde představeny vybrané architektonické vzory a na základě jejich vlastností je vybrán jeden konkrétní, který nejvíce vyhovuje potřebám aplikace Seznamovák.

3.6.1 Doporučovaná architektura

Donedávna nebylo definováno, jakým způsobem správně navrhnout architekturu Android aplikace a veškerý kód společně s logikou byl umístěn v komponentách Activity či Fragment, což vedlo k robustnosti a nepřehlednosti kódu odpovídajících tříd. Společnost Google však vydala návod, jak by měla aplikace vypadat z hlediska architektury, aby byly ve výsledku zajištěny správné principy jako například rozšiřitelnost či testovatelnost. V rámci této architektury jsou použity především Architecture Components představené v sekci 3.4.

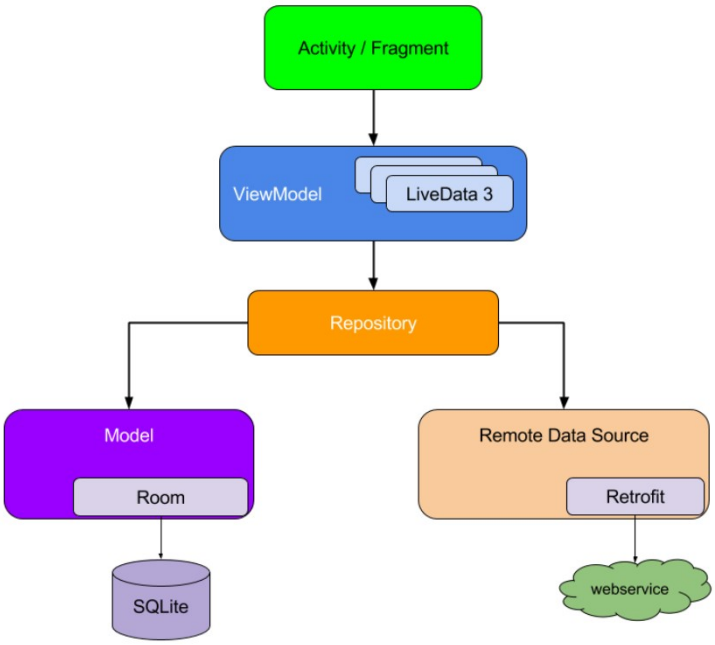
Jednotlivé části doporučované architektury a jejich interakce jsou zachyceny na obrázku 3.5, ze kterého je patrné, že závislosti jsou pouze jednosměrné, vždy shora dolů, čili nevznikají složité vazby mezi jednotlivými vrstvami. Kromě Repository mají navíc všechny pouze jednu závislost.

Repository může mít vazeb více, jelikož představuje komponentu obstarávající zisk dat. Tato data mohou být získávána z více zdrojů, mezi které se řadí například webové služby či perzistentní datové modely. Díky Repository nevědí komponenty implementující poskytované API, odkud jsou data získávána, a rozhodnutí o zdroji dat záleží na implementaci Repository.

AC, které jsou použity v rámci ostatních vrstev, jsou navrženy tak, aby usnadnily implementaci zamýšlené architektury bez nutnosti psaní boilerplate kódu. Vzhledem ke složitosti životních cyklů UI komponent je přínosný také fakt, že obsahují logiku pro správné reakce na změny životního cyklu komponent, díky čemuž napomáhají korektní práci s pamětí. Celkově tato architektura usnadňuje vývojářovu práci, napomáhá udržitelnosti kódu a usnadňuje jeho rozšiřitelnost a případné změny.

3.6.2 Architektonické vzory

V této sekci jsou popsány architektonické vzory, jež se používají při tvorbě nejen mobilních aplikací pro zajištění dobré udržitelnosti kódu. Existují určité



Obrázek 3.5: Doporučovaný návrh architektury [29]

dobré vlastnosti, kterých se každý vzor snaží docílit. Dle [30] se jedná například o tyto principy:

- **Oddělení zodpovědnosti**, neboli Separation of Concerns (SOC), jehož cílem je zajištění, že každá část je zodpovědná za určitou funkcionalitu. Tyto funkcionality se vzájemně nepřekrývají. Důsledkem dodržení SOC je testovatelný, udržitelný a snadno rozšiřitelný kód.
- **Uživatelské rozhraní řízené modelem**, který je nezávislý na objektech UI a tedy není ovlivněn jejich životním cyklem.

Mezi používané vzory v rámci Android aplikací patří především MVC, MVP a MVVM, jež jsou popsány v následujících sekcích.

3.6.2.1 MVC

Model-View-Controller je jeden z nejčastěji používaných návrhových vzorů. Každá část MVC je zodpovědná za jinou funkcionalitu.

- **Model** je zodpovědný za business logiku, manipulaci s databází a síťovým API. Jedná se o datovou vrstvu aplikace.
- **View** reprezentuje UI vrstvu k zobrazení dat z Modelu uživateli. V rámci Android aplikací se jedná o komponenty Activity a Fragment.
- **Controller** slouží jako vrstva logická. Jeho hlavním cílem je zpracování uživatelových akcí a aktualizace Modelu.

Nevýhodou této architektury je závislost View na Controlleru a Modelu zároveň. Změna logiky UI tedy může vyžadovat také změny velkého počtu ostatních tříd. Dále není jasné, kdo je zodpovědný za zpracování dat pro zobrazení v UI. Controller aktualizuje Model, který následně poskytuje data pro View. Logika zobrazení tedy může být v Modelu či View. V případě zpracování dat ve View není možné jej podrobit unit testům. Pokud jsou data zpracována v Modelu, přechází do něj zodpovědnost za logiku zobrazení a Model se tak stává závislým na daném View. [31]

3.6.2.2 MVP

Model-View-Presenter je dalším z rodiny architektonických vzorů. Jednotlivé jeho části jsou:

- **Model**, představující datovou vrstvu, jež je zodpovědný za business logiku a komunikaci s vrstvou síťovou a s databází, stejně jako v případě MVC.

- **View**, který reprezentuje UI vrstvu a slouží k zobrazení dat a upozornění komponenty Presenter o uživatelových akcích. View může být komponenta typu Activity či Fragment.
- **Presenter** získává data z Modelu, rozhoduje, co a jak zobrazit, a zpracovává notifikace o akcích uživatele z View.

Tento vzor řeší nedostatky MVC. Zbavuje View závislosti na Modelu a veškerá logika společně se zpracováním dat k prezentaci je přesunuta do jedné třídy – Presenter. Tato třída se však stává robustní, s velkou zodpovědností, a tedy i velkým počtem řádek kódu. [32]

3.6.2.3 MVVM

Model-View-ViewModel je architektonický vzor, který umožňuje rychlé reakce na změny v návrhu, mající tyto části:

- **Model** zodpovědný za veškerou business logiku, abstrakci zdrojů dat a spolupráci s komponentou ViewModel při zisku a ukládání dat.
- **View** informuje ViewModel o akcích uživatele. Jedná se o UI, reprezentované komponentou Activity či Fragment.
- **ViewModel** vystavuje data, která jsou pro konkrétní View relevantní. V rámci Android aplikace je reprezentován stejnojmennou komponentou.

Tento vzor přináší SOC a tedy snadné testování jednotlivých komponent. Všechna UI logika je navíc přesunuta z View, které slouží pouze pro prezentaci dat uživateli. ViewModel nemá žádnou referenci na View a pouze poskytuje data, jež jsou zobrazena ve View, které reaguje na jejich změny a je podle nich aktualizováno. MVVM využívá především *Observer pattern*[16] ke komunikaci mezi jednotlivými komponentami. [33]

3.6.3 Shrnutí

Pro tvorbu aplikace Seznamovák byl vybrán architektonický vzor MVVM. Hlavním důvodem byla jeho snadná implementace v případě použití Architecture Components. Tento vzor je navíc použit v rámci doporučované architektury, jež je následována i v případě implementace této aplikace.

3.7 Získávání dat

Backend této aplikace, který není součástí práce, ale byl vytvořen organizátory akce Seznamovák, poskytuje REST API pro získávání potřebných dat pro

[16] Jedná se o návrhový vzor k propagaci změn do komponent na nich závislých.

mobilní aplikaci. Data jsou dostupná na jednotlivých endpointech ve formátu JSON[17]. Jmenovitě se jedná o tyto endpointy:

- **/get-mobile-user-data-json/{email}** je endpoint poskytující informace o uživateli s danou emailovou adresou. Data dostupná na tomto endpointu jsou:
  - unikátní id,
  - turnus (v případě organizátorů je turnus 0),
  - mafiaId v případě účastníků,
  - mafiaId1 v případě organizátorů,
  - mafiaId2 v případě organizátorů.

  Položky mafiaId, mafiaId1 a mafiaId2 představují identifikátory v rámci hry Kolíčkovaná, přičemž organizátoři mají identifikátory na každý turnus zvlášť.
- **/news-json/{turnus}** poskytující seznam všech novinek pro daný turnus. Pokud turnus není specifikován, jedná se o novinky určené oběma turnusům.
- **/harmonogram-json/{turnus}**, na kterém se nachází jednotlivé položky harmonogramu, rozdělené dle dnů, a dále pak data jednotlivých dnů, během nichž se turnus odehrává. Každá položka harmonogramu obsahuje informace začátku a konce události a její název.
- **/organizer-harmonogram-json/** endpoint má stejnou strukturu jako harmonogram pro účastníky, avšak obsahuje jiné položky, které jsou relevantní pro organizátory. Tento endpoint navíc obsahuje data obou turnusů.
- **/points-json/{id}** obsahuje seznam her a počet bodů, které v každé z nich získal hráč s daným id.
- **/mafia-info/{turnus}/{id}/{isOrganizer}** je endpointem pro hru Kolíčkovaná, jež poskytuje informace týkající se hráče s daným id. Parametr isOrganizer nabývá hodnot 0 či 1, podle toho, zda je hráč účastníkem Seznamováku nebo zdali se jedná o jeho organizátora. Tato informace je důležitá, jelikož id organizátora a účastníka může mít stejnou hodnotu.
- **/mafia-kill/{turnus}** na tento endpoint je dotazováno během „zabíjení“ v rámci hry Kolíčkovaná. Celý proces je rozebrán v sekci 4.3.

[17] JavaScript Object Notation

3.8 Ukládání dat

Jelikož má být tato aplikace funkční i bez připojení k internetu, je zapotřebí ukládání získaných dat pro jejich offline použití. V této sekci jsou představeny jednotlivé možnosti ukládání dat, jejich výhody a nevýhody a následně je zvolen typ úložiště vhodný pro aplikaci Seznamovák. Sekce vychází z oficiální dokumentace dostupné na [34].

Vývojář má několik možností pro práci s daty. Během rozhodování, jaké úložiště zvolit, hraje roli několik faktorů. Jedná se o data specifická pouze pro danou aplikaci? Jsou data typu klíč–hodnota či se jedná o data strukturovaná? Data mohou být navíc ukládána do interní či externí paměti. Informace nezbytné pro správný chod aplikace by se neměly nacházet v externí paměti, jelikož například po vyjmutí SD karty by nebyla dostupná. U interní paměti naopak může být problém s jejím nedostatkem. Je tedy potřeba vhodně zvážit, jaké úložiště použít.

Na základě přístupu k datům můžeme úložiště dělit na:

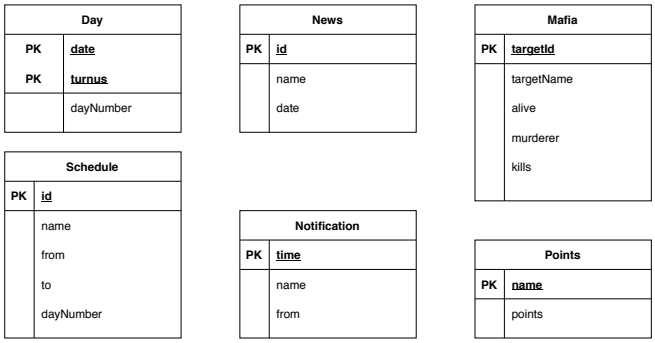
- **App-specific**, které obsahuje data specifická pro danou aplikaci. Po jejím odinstalování jsou odstraněna i všechna data v tomto úložišti.
- **Shared**, neboli sdílené úložiště, obsahuje data, která jsou dostupná napříč více aplikacemi a v případě odinstalování jedné aplikace zůstávají zachována.

Pro perzistentní ukládání dat specifických pro danou aplikaci existuje také více možností, lišících se především strukturou a velikostí dat, která zde mohou být uložena. Jedná se o:

- **Shared Preferences** sloužící pro ukládání dat typu klíč–hodnota. Toto úložiště bývá používáno v případě malého množství dat primitivních typů, jako je int, boolean, float apod., jelikož jeho velikost je omezená. Naopak výhodou Shared Preferences je přímočarý a rychlý přístup k datům na základě klíče.
- **Databáze** (DB) bývá používána v případě, že je zapotřebí perzistentně uložit větší množství strukturovaných dat. V rámci doporučované architektury od společnosti Google je použita Room Database Library, popsaná v sekci 3.4.3. Oproti Shared Preferences je nevýhodou DB delší doba trvání přístupu k datům v ní uložených.

V rámci aplikace Seznamovák může být na jednom zařízení přihlášen pouze jeden uživatel. Informace o uživateli proto budou uložena do Shared Preferences, jelikož k nim bude často přistupováno, jedná se o jednoduchá data typu klíč–hodnota a objem těchto dat je malý. V rámci Shared Preferences tedy budou uložena data získaná z odpovídajícího endpointu, popsaného v části 3.7, společně s emailovou adresou, kvůli následné aktualizaci dat.

Informace týkající se počtu bodů, harmonogramu, novinek a hry Kolíčkovaná budou naopak uloženy v lokální databázi, jelikož se jedná o větší množství dat stejných typů, která mohou být reprezentována databázovými entitami. Oproti informacím získaných z REST API endpointů budou do databáze ukládány také jednotlivé notifikace k harmonogramu. Každá taková položka obsahuje název notifikované události, čas začátku dané akce a čas, ve který má být notifikace uskutečněna. Model databáze se nachází na obrázku 3.6.



Obrázek 3.6: Návrh databáze

Kapitola 4

Implementace

Tato kapitola se zabývá popisem tvorby mobilní aplikace Seznamovák. Nejprve je čtenář seznámen s použitými nástroji a technologiemi. Dále jsou v této kapitole popsány realizace jednotlivých zajímavých částí aplikace. Nakonec je zde také představeno řešení pro implementaci aplikace Magistrovák.

4.1 Nástroje

V této sekci jsou představeny jednotlivé nástroje použité během vývoje mobilní aplikace Seznamovák.

4.1.1 Vývojové prostředí

Zde jsou popsána některá nejznámější vývojová prostředí[18] pro tvorbu Android aplikací a je zde také představeno vybrané IDE použité v rámci implementace aplikace Seznamovák.

Existuje celá řada vývojových prostředí pro tvorbu Android aplikací, mezi které se řadí například Android Studio, IntelliJ IDEA, či Eclipse IDE. Do roku 2015 společnost Google doporučovala použití Eclipse IDE společně s Android Development Tools (ADT) pluginem, což se však ještě téhož roku změnilo, byl ukončen vývoj ADT a nyní je oficiálně doporučovaným vývojovým prostředím Android Studio, které je dostupné ke stažení na stránkách Android Developers [35].

IntelliJ IDEA je IDE, které není určeno pouze Android vývojářům, a je tedy vhodné i pro tvorbu multiplatformních aplikací. Android Studio z tohoto vývojového prostředí vychází a na jeho základě přináší další vylepšení speciálně pro vytváření Android aplikací. Jelikož tato práce se zabývá tvorbou mobilní aplikace pouze pro OS Android, mezi těmito vývojovými prostředími

[18] Integrated Development Environment (IDE)

je vhodnější variantou Android Studio. Díky výše zmíněné oficiální podpoře je také vítězem v porovnání s Eclipse IDE.

Pro vývoj aplikace Seznamovák bylo vybráno Android Studio. Jedná se o IDE, které se specializuje na vývoj aplikací pro OS Android. Android Studio má oproti ostatním vývojovým prostředím několik výhod, mezi něž patří například oficiální podpora či fakt, že implementace ukázek kódu u tutoriálů od společnosti Google jsou tvořeny výhradně v tomto IDE. Android Studio taktéž podporuje různé verzovací systémy. V rámci Android Studia je navíc již integrovaný nástroj na automatizaci Gradle, který slouží pro automatické sestavení, zavedení a spuštění aplikace. [10]

4.1.2 Verzovací systém

Aby vývojář předešel nechtěným ztrátám zdrojových kódů nebo aby nedocházelo k jejich přepsání bez možnosti návratu, bývá běžnou praxí používat během vývoje systém na správu verzí[19], který předchází výše zmíněným nepříjemnostem a umožňuje práci z více zařízení. Zde byl použit verzovací systém Git společně s webovou službou GitLab pro zálohování.

Pro implementaci jednotlivých funkcionalit byly vytvářeny samostatné větve[20], které byly následně spojeny[21] do vývojové větve *develop*. V rámci zvoleného IDE Android Studio je navíc po správné integraci dostupné Version Control okno, které přehledně zobrazuje jednotlivé větve a změny, které v rámci nich byly učiněny, a usnadňuje jednotlivé verzovací úkony například pomocí klávesových zkratek, jimiž toto IDE disponuje.

4.2 Technologie

V této části práce jsou rozebrány jednotlivé technologie, jež byly použity v rámci implementace aplikace Seznamovák, a jejich fungování.

4.2.1 Firebase

Firebase je platforma s řadou funkcí. Lze ji využít například pro zasílání notifikací či testování aplikace. Dále může sloužit jako cloudové úložiště, lze s její pomocí realizovat bezpečné přihlášení do aplikace apod. V rámci této bakalářské práce byla platforma Firebase použita pro zasílání notifikací, testování aplikace a její distribuci během uživatelského testování.

K zasílání notifikací do zařízení slouží služba Firebase Cloud Messaging (FCM). Jelikož se jedná o řešení fungující již v původní verzi aplikace, v rámci této práce byla nutná pouze implementace na straně klienta. FCM slouží k zasílání zpráv na libovolnou platformu, lze tedy zasílat zprávy na zařízení

[19] Version Control System (VCS)
[20] z anglického branches
[21] z anglického merge

s OS Android, iOS či na webovou aplikaci. Zprávy mohou být určeny jednotlivým zařízením, skupinám zařízení či mohou být zasílány na témata[22], k jejichž odběru se zařízení přihlásí.

Přijatá zpráva může být typu notifikace či datová zpráva, jež obsahuje data typu klíč–hodnota. V případě notifikační zprávy je uživateli daná zpráva zobrazena. Jedná-li se o datovou zprávu, je její zpracování zcela v kompetenci dané aplikace. Použití této služby v rámci aplikace Seznamovák je popsáno v sekci 4.6.1. [36]

4.2.2 Retrofit

Jak již bylo zmíněno v sekci 3.7. Data pro mobilní aplikaci jsou získávána pomocí REST API z odpovídajících endpointů. Pro komunikaci s tímto API byla použita knihovna Retrofit.

Retrofit užívá anotace k definování jednotlivých HTTP požadavků. Umožňuje také dynamické změny částí URL na základě parametrů metod. Jelikož jednotlivé endpointy vracejí data ve formátu JSON, je zde použita knihovna Moshi pro konverzi JSON stringu na Kotlin objekty.

Ukázka kódu týkající se komunikace se serverem v rámci hry Kolíčkovaná je zachycena výpisem kódu 4.1. HTTP metoda je specifikována pomocí anotace a v rámci URL jsou použity parametry, které jsou opět specifikovány pomocí anotací.

4.2.3 Osmdroid

Jedním z požadavků byla implementace mapy, jež bude funkční i offline. Za tímto účelem byla použita knihovna Osmdroid, jež umožňuje nahrazení Android MapView a je založena na OpenStreetMap, která požadavek na offline dostupnost bezplatně splňuje. Popis použití této knihovny v rámci implementace aplikace se nachází v sekci 4.4. Tato knihovna byla použita především díky dobré dokumentaci s ukázkami použití. Umožňuje do mapy přidat různé prvky, jakými jsou například aktuální pozice zařízení, minimapa, měřítko či ikona představující určité místo.

4.2.4 Koin

Koin je nástroj umožňující vkládání závislostí[23], což znamená, že jednotlivé instance tříd, na kterých je jiná třída závislá, jsou do ní vloženy, než aby si třída dané instance vytvářela sama. Tento postup vede k lepší udržitelnosti kódu, znovupoužitelnosti a usnadňuje testování, jelikož lze do testovaných tříd vložit mockované instance[24].

[22] z anglického topics

[23] Dependency Injection (DI)

[24] instance tříd s předdefinovaným chováním pro usnadnění testovaní

```
interface MafiaApiService {
    @GET("mafia-info/{keyword}/{turnus}/{id}/{isOrganizer}")
    suspend fun getMafia(
        @Path("id") id: Int,
        @Path("turnus") turnus: String,
        @Path("isOrganizer") isOrganizer: Int,
        @Path("keyword") keyword: String = Keys.readAPIKey()
    ): Mafias?

    @FormUrlEncoded
    @POST("mafia-kill/{keyword}/{turnus}")
    suspend fun killTarget(
        @Path("turnus") turnus: String,
        @Field("user_id") user: String,
        @Field("target_id") target: String,
        @Field("check_name") newTarget: String,
        @Path("keyword") keyword: String = Keys.readAPIKey()
    ): ReturnCode
}
```

Výpis kódu 4.1: Ukázka použití knihovny Retrofit

Výhodou nástroje Koin je snadná počáteční implementace, jednoduché použití a srozumitelná dokumentace. Zde je použit především na vytváření single instancí, což znamená, že je vytvořena jedna instance třídy označené jako *single* a ta je následně vložena pomocí tohoto nástroje do třídy, jež ji vyžaduje a je na ní závislá. V aplikaci Seznamovák se jedná například o vkládání závislostí na Repository třídy v rámci ViewModel tříd. Pro zajištění správného odstínění jednotlivých částí je zde navíc vkládána vždy závislost pouze na interface. Samotná použitá instance je definována jinde a vložena pomocí Koinu. Ukázka použití tohoto nástroje je zachycena výpisem kódu 4.2.

4.3 Kolíčkovaná

Jedním z hlavních požadavků na aplikaci byla implementace hry Kolíčkovaná. Původní návrh obsahoval nutnost potvrzení „zabití“ uživatelem, který byl „zabit“, avšak tento proces by mohl být zdlouhavý, pokud by tak neučinil ihned. Proto pro potvrzení „zabití“ bude dostačující zadat jméno následujícího cíle, které se ověří v rámci webové aplikace bez nutnosti potvrzení od „zabitého“.

Data pro tuto hru jsou dostupná na odpovídajícím endpointu. Obsahují jméno a id hráče, jež má být „zabit“, počet doposud „zabitých“ osob, jméno vraha, pokud byl daný hráč „zabit“ a stav hráče, který nabývá hodnoty 0 v případě, že hráč byl již „zabit“, či 1 v opačném případě. Pro „zabití“ je třeba

```
val userModule: Module = module {
    fun provideApi(retrofit: Retrofit): UserApiService {
        return retrofit.create(UserApiService::class.java)
    }

    single { provideApi(get()) }
    single<IUserRepository> { UserRepository(get(),
        get(named("shared"))) }
    viewModel { LoginViewModel(get()) }
}
```

Výpis kódu 4.2: Ukázka použití nástroje Koin

kliknout na odpovídající tlačítko a zadat jméno nového cíle dle připravené šablony ve formátu *Příjmení Jméno*. Aplikace však akceptuje i zadání jména v opačném pořadí, jelikož někteří organizátoři namítali, že na kolíčku je formát *Jméno Příjmení* a v případě cizokrajných jmen není jasné, která část je jméno a která příjmení. Aplikace proto akceptuje jméno cíle v libovolném pořadí jednotlivých částí.

Pomocí metody POST jsou serveru předány identifikátory přihlášeného uživatele, původního cíle a jméno osoby, které uživatel zadal jako svůj nový cíl. Tato metoda vrací JSON se dvěma položkami, kterými je kód a popis. Jedná se o enum[25], který nabývá hodnoty dle výsledku akce. Může nastat několik situací:

- Uživatel zadal jméno správně. V takovém případě jsou data aktualizována a uživateli je přiřazen nový cíl.
- Uživatel zadal jméno nového cíle špatně. Zde je situace odlišná v případě opakovaného zadání špatného jména.
  (a) Při prvním zadání chybného jména „zabití“ neproběhne.
  (b) Při druhém zadání chybného jména „zabití“ neproběhne a uživateli je udělen zákaz „zabíjet“ na následující hodinu.
  (c) Pokusí-li se uživatel zabíjet v době, kdy má udělen zákaz, „zabití“ neproběhne i v případě zadání správného jména a je pouze zobrazen počet zbývajících minut zákazu.
- Nehledě na zadané jméno, může nastat chyba na straně serveru.
- Uživatel nemá připojení k internetu, a tedy nemůže dojít k provedení akce.

[25] výčtový typ

- Uživateli byl udělen úplný zákaz zabíjet.

Ve všech výše zmíněných případech je uživatel o výsledku provedené akce informován pomocí Toastu[26].

4.4 Mapa

V rámci aplikace je zapotřebí správné fungování mapy i bez připojení k internetu. Za tímto účelem je použit offline zdroj map, nezávislý na stavu tohoto připojení.

Nejprve bylo potřeba vytvořit offline archiv požadovaných map. Za tímto účelem byl použit nástroj Mobile Atlas Creator (MOBAC) [37], který umožňuje výběr zdroje map, oblasti a jednotlivých přiblížení. Následně lze vytvořit atlas požadovaného formátu. V tomto případě se jedná o formát Osmdroid ZIP, který je kompatibilní s použitou knihovnou Osmdroid. Použitá mapa je dostupná na stránkách [38]. V MOBAC byl vytvořen vlastní zdroj se staženou mapou České republiky a byla vybrána požadovaná oblast cca 1 km v okolí ubytovacího areálu. Vygenerovaný atlas byl dále použit jako offline zdroj v aplikaci Seznamovák.

Uživateli je v rámci aplikace dostupný tento výřez map s omezeným přiblížením. Kdyby zde toto omezení nebylo, uživatel by mohl mapu libovolně oddalovat a přibližovat, přičemž v případě, že pro danou hodnotu přiblížení neobsahuje offline zdroj map potřebná data, byla by mapa rozostřená. Centrálním bodem je hlavní budova rekreačního areálu. Pokud uživatel povolí přístup k polohovým údajům, je mu v rámci mapy zobrazena také poloha zařízení. Vyhledávání v mapě je uskutečněno pomocí dialogového okna, do něhož uživatel zadá souřadnice dle definovaného formátu. Požadovaný bod je následně na mapě zobrazen. Pokud se jedná o souřadnice mimo dostupnou mapu nebo jsou-li zadány nesprávné souřadnice, je u dané položky zobrazena chybová zpráva.

Jelikož offline mapa byla vytvořena z OpenStreetMap, bylo nutné na základě autorských práv tuto skutečnost uvést pomocí odkazu na požadovanou webovou stránku [39]. Tento odkaz se nachází v dolním rohu obrazovky Mapa.

4.5 Navigace

Aplikace Seznamovák má 5 hlavních částí (Program, Novinky, Body, Kolíčkovaná, Mapa), mezi nimiž se uživatel pohybuje pomocí lišty Bottom Navigation Bar, která je popsána v sekci 3.3.1.1. Tuto práci velmi usnadňuje použití Navigation komponenty z Android Architecture Components. V rámci aplikace jsou vytvořeny navigační grafy, v nichž jsou definovány jednotlivé destinace a možné přechody mezi nimi.

[26] krátká zpráva zobrazená v dolní části obrazovky

Na základě dotazníku vznikl požadavek, aby během navigace mezi jednotlivými destinacemi zůstávala zachována pozice v rámci seznamu položek. Tedy aby uživatel mohl pokračovat v procházení položek tam, kde skončil. Ačkoliv se jedná o smysluplný požadavek, v současnosti jej nelze realizovat použitím navigační komponenty společně s Bottom Navigation Bar, jelikož není implementována práce s více zásobníky, na které jsou ukládány navštívené Fragmenty [40]. Vývojáři společnosti Google implementovali prozatímní řešení, v rámci něhož má každá destinace vlastní NavHostFragment. Řešení je dostupné v projektu na platformě GitHub [41] v rámci souboru NavigationExtensions a celý vývoj tohoto problému lze sledovat na stránkách Issue Tracker [42].

Použití tohoto řešení má výhodu z hlediska chování. Při stisknutí tlačítka zpět je uživateli zobrazena první záložka, jež zde odpovídá obrazovce Program. Při opětovném stisknutí tohoto tlačítka je aplikace ukončena. Toto chování neporušuje principy správné navigace, avšak nelze jej prohlásit za nejlepší, jelikož v současné době má každá aplikace jiné chování. Aplikace YouTube například při stisknutí tlačítka zpět postupně prochází dříve navštívené záložky v opačném pořadí. Aplikace Google Fotky je naopak ukončena již při prvním stisknutí tlačítka zpět, nehledě na fakt, zda je aktuální obrazovka primární či nikoliv. Návrat na primární záložku při prvním stisknutí tlačítka zpět je proto dobrým kompromisem mezi těmito dvěma přístupy.

Jelikož je tento problém velmi častý, existují i jiná řešení. Jedním z nich je například vytvořit navigační ViewModel, jehož životní cyklus se váže k danému navigačnímu grafu. V rámci této ViewModel třídy lze ukládat jednotlivé proměnné potřebné pro opětovné načtení Fragmentu s původním stavem. Jedná se například o pozici v rámci seznamu či hledané místo na mapě. Toto řešení však znamená, že při každém navigování mezi záložkami je daný Fragment a odpovídající ViewModel zničen, což není ideální v případě častého navigování mezi jednotlivými položkami komponenty Bottom Navigation Bar. Dalším nedostatkem tohoto řešení při snaze o jeho implementaci byl fakt, že se nepodařilo načíst pozici v rámci položek harmonogramu. Při každém návratu byl zobrazen naposledy viděný den, avšak vždy byl vrácen na začátek. Z těchto důvodů nebylo toho řešení shledáno ideálním a nebylo tedy použito v rámci implementace aplikace Seznamovák.

Druhým možným řešením je ukládání stavů perzistentně například do Shared Preferences. Zde by však mohl nastat problém, pokud by bylo třeba uložit větší množství dat, které by navíc nebylo možné jednoduše uložit ve formátu klíč–hodnota.

Další možností je definování vlastní třídy typu Navigator, která je poté přidána ke komponentě NavController. Tento postup však není uspokojivě zdokumentován a nebyl nalezen návod, jak takový Navigator správně implementovat.

Na základě výše zmíněných faktů bylo rozhodnuto o použití řešení, jež nabízí společnost Google, tedy implementaci pomocí více navigačních grafů. Toto

řešení je jednoduché a srozumitelné a přestože je funkční a splňuje požadované chování, lze jej vnímat jako dočasné a pokud bude v budoucnu požadavek na práci s více zásobníky uspokojen, lze jej nahradit. V takovém případě by postačil jeden navigační graf a jemu odpovídající NavController.

4.6 Notifikace

V rámci aplikace jsou všem účastníkům zasílány notifikace týkající se přiřazení bodů, vytvoření novinky a blížící se položky harmonogramu. V případě organizátorů notifikace týkající se bodů zasílány nejsou. První dva typy notifikací jsou zajištěny webovou aplikací skrze Firebase Cloud Messaging, zatímco notifikace harmonogramu jsou oproti původní verzi vytvářeny přímo v rámci mobilní aplikace na základě dat uložených v databázi.

K tomuto rozdělení došlo především proto, aby uživatelé byli informováni o položkách harmonogramu bez závislosti na internetovém připojení. V této sekci jsou popsány implementace obou typů použitých notifikací.

V obou případech je na základě přijatého Intentu vždy zjištěno, které domény se notifikace týká. Podle tohoto údaje je aplikace po kliknutí na notifikaci otevřena na správné záložce (Body, Novinky či Program).

4.6.1 Firebase Cloud Messaging

V rámci aplikace Seznamovák je využíváno zasílání zpráv na jednotlivá témata. Konkrétně se jedná o témata s názvy *news-<číslo_turnusu>*, *points-<číslo_turnusu>* a *harmonogram-<číslo_turnusu>*. Číslo turnusu je v případě účastníka turnus, na který je registrován, a v případě organizátorů jsou odebírány zprávy z témat obou turnusů. Po přihlášení uživatele do aplikace se zařízení přihlásí k odběru vybraných témat. V případě nově vzniklé Android aplikace se jedná o notifikace týkající se novinek a přidělení bodů. Jak již bylo zmíněno výše, upozornění na jednotlivé položky harmonogramu jsou vytvářena přímo v Android aplikaci, a tedy zařízení se k odběru tohoto tématu nepřihlašuje.

V původní verzi aplikace byly zprávy posílány jakožto zprávy notifikační. Ty jsou vytvořeny a zobrazeny automaticky. Zde vývojář nemá kontrolu nad přijímáním těchto zpráv, může však v rámci souboru Android Manifest definovat, jak se mají přijaté notifikace zobrazit. Lze tedy definovat, na jaký kanál budou přijaté notifikace defaultně zasílány a také jaká bude barva a ikona notifikace. Pokud je například zapotřebí notifikaci zobrazit i v případě, že je aplikace na popředí, je nutné přepsat metodu *onMessageReceived* a zde definovat požadované chování.

Přijímání notifikačních zpráv má několik nevýhod. Lze určit, na jaký kanál mají být zprávy zasílány, avšak nelze je posílat na více různých kanálů. V případě, že je zapotřebí společně s přijatou notifikací učinit další akci, kterou může být například stažení nových dat, nelze takového chování dosáhnout

pomocí notifikačních zpráv, jelikož není volána metoda *onMessageReceived* v případě, že aplikace není na popředí.

Zmíněné nedostatky notifikačních zpráv zapříčinily změnu na straně serveru. Nově jsou posílány datové zprávy a všechny potřebné informace pro vytvoření notifikace jsou ukládány do datové části zprávy ve formátu klíč–hodnota. Jedná se především o název a text notifikace. Datová zpráva je vždy přijata a zpracována v přepsané metodě *onMessageReceived* třídy FCMService, jež dědí od třídy FirebaseMessagingService. Přijatá zpráva obsahuje název tématu, kterého se týká. Na základě této informace je vytvořena notifikace, jež je poslána na kanál odpovídající domény.

Díky vlastnímu zpracování přijaté zprávy je možné notifikaci přizpůsobit požadovanému chování a rozhodnout, na který kanál má být poslána. Zde se například liší ikony jednotlivých kanálů, jež odpovídají ikonám použitým v rámci navigační komponenty Bottom Navigation Bar.

Zároveň jsou prostřednictvím Worker třídy, jež dědí od třídy CoroutineWorker, stažena aktuální data z endpointu, jež se váže k dané doméně. Ke stažení dat dochází i v případě, že uživatel danou notifikaci neotevře. Tento postup je vhodný, jelikož pokud by data nebyla aktualizována ihned, mohlo by se stát, že uživatel otevře notifikaci s odstupem času ve chvíli, kdy nemá připojení k internetu, a tedy položka novinek či bodů, na niž byla notifikace vytvořena, ještě není uložena v databázi, a tak nemůže být uživateli zobrazena.

V případě použití třídy Worker a specifikace požadované práce prostřednictvím třídy WorkManager lze nastavit, aby v případě neúspěchu stažení dat byla daná práce zkoušena opakovaně. Jelikož však zpráva dorazí do zařízení pouze ve chvíli, kdy má připojení k internetu, lze očekávat, že data budou taktéž ihned aktualizována. Použití služby Firebase Cloud Messaging je znázorněno výpisem kódu 4.3.

4.6.2 Interní notifikace

Pro zobrazení notifikací týkajících se harmonogramu nebyla využita služba FCM, jelikož tyto notifikace jsou frekventované a pokud zařízení nemá připojení k internetu, nedorazily by v zamýšlený čas, a tedy by ztratily svůj hlavní význam, kterým je informování uživatele 15 minut před každou událostí harmonogramu o jejím brzkém konání. Notifikace jsou proto vytvářeny přímo v aplikaci na základě dat uložených v databázi a při otevření přijaté notifikace je uživateli spuštěna aplikace Seznamovák na záložce právě probíhajícího dne.

Za tímto účelem aplikace obsahuje službu NotificationService, jež dědí od třídy JobIntentService. Tato služba je volána při každé změně notifikačních dat v DB, přičemž nastaví alarm na čas 15 minut před nejbližší položkou harmonogramu. Aplikace obsahuje navíc Broadcast Receiver, jež zpracovává Intent, který je nastaveným alarmem v požadovaný čas vyslán. Následně je notifikace vytvořena, zobrazena uživateli a tento proces se opakuje s následující

```
<!--  definice služby v souboru AndroidManifest.xml   -->
<service
    android:name=".core.fcm.FCMService"
    android:exported="false">
    <intent-filter>
        <action
            android:name="com.google.firebase.MESSAGING_EVENT"/>
    </intent-filter>
</service>

// soubor FCMService.kt
class FCMService : FirebaseMessagingService() {
    override fun onMessageReceived(
        remoteMessage: RemoteMessage
    ) {
        super.onMessageReceived(remoteMessage)
        val channelInfo = FlavourSpecific.participantTopics
            .find {
                remoteMessage.from?.contains(it.from, true)
                    ?: false
        } ?: return
        val data = remoteMessage.data
        val title = data[KEY_TITLE] ?: return
        val body = data[KEY_BODY] ?: return
        // create notification
        createNotification(title, body, channelInfo)
        //refresh stored data of the domain
        refreshData(channelInfo)
    }
    //...
}
```

Výpis kódu 4.3: Ukázka použití služby Firebase Cloud Messaging

položkou harmonogramu. Zmíněný Broadcast Receiver také zpracovává akci BOOT_COMPLETED, jež je realizována po restartování zařízení. V takovém případě jsou totiž všechny nastavené alarmy zrušeny a je třeba je opětovně nastavit.

4.7 Nastavení

Oproti původnímu návrhu byla v průběhu implementace do aplikace přidána obrazovka Nastavení. Tato obrazovka je dostupná ze všech destinací přihlášeného uživatele prostřednictvím Overflow Menu.

Nastavení bylo přidáno, aby uživatel mohl snadno upravit své preference týkající se vzhledu aplikace a přijímaných notifikací. V rámci této obrazovky může změnit motiv aplikace ze světlého na tmavý a naopak (viz sekce 4.8). Tato funkcionalita byla přesunuta z Overflow Menu na obrazovku Nastavení a je realizována přepínačem[27] namísto zaškrtávacího políčka z původního návrhu. Uživatel má dále možnost vypnout či zapnout přijímání notifikací týkajících se přiřazení bodů, jedná-li se o uživatele v roli účastníka. Notifikace týkající se harmonogramu a vytvoření novinky nelze na této obrazovce vypnout. Důvod je ten, že jednou z hlavních funkcí aplikace Seznamovák má být usnadnění organizace kurzu a informování účastníků, což je realizováno právě prostřednictvím těchto notifikací, a organizátoři si nepřejí, aby bylo tímto způsobem usnadněno jejich vypnutí. Uživatel stále má možnost vypnout přijímání oznámení v aplikaci Nastavení, jež spravuje celý OS, čemuž nelze zabránit. Tato možnost by však neměla být nabízena přímo v aplikaci Seznamovák.

K vytvoření Nastavení byla použita knihovna Androidx Preference, která obstarává ukládání preferencí a vzhled obrazovky, aby aplikace byla konzistentní napříč zařízeními a různými verzemi OS [43]. Struktura preferencí je definována v XML souboru, v němž lze jednotlivé položky sdružovat v rámci kategorií. Implementace XML souboru pro obrazovku Nastavení je zachycena výpisem kódu 4.4. Zde se jedná o kategorie Vzhled a Notifikace. Každá preference je definována klíčem, jež ji identifikuje, a názvem, který je zobrazen uživateli.

Lze také určit, zda jednotlivé položky budou viditelné či nikoliv. Toho je využito v případě notifikací týkajících se přidělování bodů. Pokud je přihlášený uživatel organizátor, tato položka s klíčem *points* mu zobrazena není, jelikož organizátoři se neúčastní her, nezískávají body, a tedy nejsou přihlášeni k odběru zpráv z tématu dané domény a toto nastavení by pro ně bylo irelevantní. Jelikož v současné době jsou Body jedinou položkou kategorie Notifikace, je organizátorům skryta celá tato kategorie.

Nastavení jsou perzistentně ukládána do Shared Preferences. SettingsFragment implementuje listener na změny hodnot preferencí. Na základě klíče je

[27] z anglického switch

```
<PreferenceScreen
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <PreferenceCategory
        app:key="@string/settings_ui_key"
        app:title="@string/settings_ui_title">
        <SwitchPreferenceCompat
            app:defaultValue="@bool/default_dark_theme"
            app:key="@string/dark_theme"
            app:title="@string/dark_theme_label" />
    </PreferenceCategory>
    <PreferenceCategory
        app:key="@string/settings_notification_key"
        app:title="@string/settings_notifications_title">
        <SwitchPreferenceCompat
            app:defaultValue="@bool/default_fcm_subscription"
            app:key="@string/topic_points_name"
            app:title="@string/points_label" />
    </PreferenceCategory>
</PreferenceScreen>
```

Výpis kódu 4.4: XML soubor obrazovky Nastavení

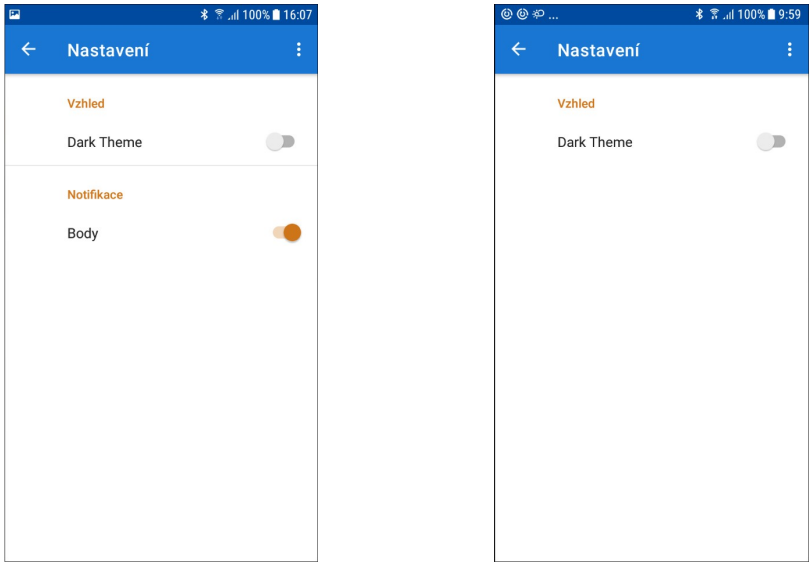
každá taková změna vyhodnocena a zpracována. Jednotlivé položky nastavení mohou být definovány přepínačem, obyčejným textem, který kupříkladu vyvolá přechod na jiný Fragment, zaškrtávacím políčkem či například editovatelným textem, jejž lze použít při definici uživatelského jména apod. Obrazovky Nastavení výsledné aplikace pro uživatele a účastníky jsou zachyceny na obrázcích 4.1a a 4.1b.

4.8 Dark Theme

Jedním z hlavních požadavků na novou verzi aplikace byla možnost nastavení Dark Theme, neboli změna motivu na tmavý. Jedná se o funkci, jíž disponuje většina moderních aplikací. Možnost změny motivu byla v původním návrhu realizovaná v rámci Overflow Menu, avšak posléze byla přesunuta na obrazovku Nastavení. Změna je vždy perzistentně uložena do Shared Preferences, tudíž po opětovném spuštění aplikace je toto nastavení zachováno. Defaultně je motiv aplikace nastaven na světlý.

Aby mohl být režim aplikace měněn, je třeba, aby aplikační motiv (App Theme) dědil z DayNight Theme. Tento motiv má předdefinované chování, které však lze přizpůsobit. V rámci zdrojů[28] lze například vytvořit soubor co-

[28] z anglického resources



(a) Nastavení účastníka (b) Nastavení organizátora

Obrázek 4.1: Obrazovka Nastavení

lors.xml zvlášť pro denní a noční motiv. Dle aktuálního nastavení jsou poté použity odpovídající zdroje. Tento postup byl použit taktéž u aplikace Seznamovák.

Změna motivu je realizovaná pomocí volání metody *setDefaultNightMode*. V důsledku jejího volání je daná aktivita zničena a znovuvytvořena s opačným motivem. Motiv může být obecně nastaven na světlý, tmavý či na stejnou hodnotu jako zbytek systému.

4.9 Magistrovák

Jak již bylo zmíněno v sekci 2.2, existuje také aplikace Magistrovák používaná v rámci úvodního kurzu studentů magisterského studia. Součástí této práce byl požadavek na snadnou použitelnost aplikace Seznamovák se stejnými či podobnými funkcemi také v rámci podobných akcí, kterou Magistrovák bezesporu je.

Za tímto účelem jsou používány Product Flavors (PF), které slouží například pro tvorbu aplikace ve volně dostupné a placené verzi či právě pro tvorbu podobných aplikací, lišících se kupříkladu pouze v použitých zdrojích. Jednotlivé PF se definují v konfiguračním souboru build.gradle v bloku productFlavors. Uvnitř bloku jsou definována nastavení jako například applicationId. Dále zde musí být každé Flavor přiřazena dimenze. Dimenze je pojmenovaná skupina PF. Příkladem dimenze může být verze aplikace, její typ či verze API. Gradle v takovém případě kombinuje jednotlivé Flavors z různých dimenzí,

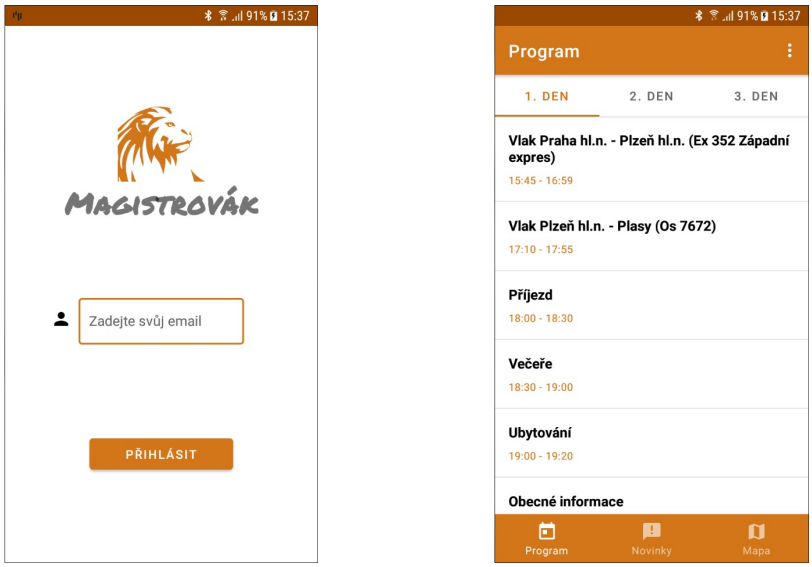
na základě kterých vzniknou všechny možné build varianty. Kdyby například dimenze *appVersion* obsahovala 2 Flavors a dimenze *API* 3 Flavors, výsledkem bude $2 * 3 = 6$ build variant pro jeden build typ. Obvykle existují minimálně 2 build typy: release a debug. V takovém případě je tedy výsledných build variant $2 * 3 * 2 = 12$. Pro každou takto vzniklou build variantu lze vytvořit sadu zdrojů[29]. Lze tedy například definovat pro každou build variantu jinou barevnou sadu, či jinou URL pro stahování dat. Při rozhodování, který zdroj bude použit, je postupováno hierarchicky v tomto pořadí:

1. kombinace Product Flavors a build typu,
2. build typ,
3. kombinace PF,
4. jednotlivé PF dle priority, která je dána pořadím dimenzí při jejich definici (PF náležící do dimenze definované více vlevo má vyšší prioritu),
5. main, neboli hlavní sada zdrojů.

V rámci této aplikace byly vytvořeny dvě PF: *seznamovak* a *magistrovak*, náležící do jedné dimenze. Tyto Flavors se liší barevnou sadou, ikonou, počtem záložek v rámci obrazovky Program (Magistrovák je o den kratší) či URL pro zisk dat. Aplikace Seznamovák navíc obsahuje oproti aplikaci Magistrovák více funkcí. Má o 2 obrazovky více, jelikož v rámci kurzu pro studenty magisterského studia není hrána Kolíčkovaná a také zde není zapotřebí počítat body z jednotlivých her. Ukázky obrazovek aplikace Magistrovák se nachází na obrázcích 4.2a a 4.2b.

Díky tomuto přístupu rozdělení aplikace do více PF je možné použít aplikaci Seznamovák i na podobných akcích pouze s drobnými úpravami, přičemž jádro aplikace se nezmění. Definice jednotlivých Product Flavours je znázorněna výpisem kódu 4.5.

[29] z anglického source set



(a) Přihlášení (b) Program

Obrázek 4.2: Obrazovky aplikace Magistrovák

```
// soubor build.gradle
android {
    flavorDimensions "event"
    productFlavors {
        seznamovak {
            dimension "event"
            applicationIdSuffix ".seznamovak"
        }

        magistrovak {
            dimension "event"
            applicationIdSuffix ".magistrovak"
        }
    }
    //...
}
//...
```

Výpis kódu 4.5: Ukázka definice Product Flavours

Kapitola 5

Testování

Tato kapitola se zabývá testováním aplikace Seznamovák. Nejprve jsou zde popsány testovací nástroje použité během vývoje aplikace. Následuje sekce popisující uživatelské testování, jemuž byla aplikace podrobena.

K testování aplikace během jejího vývoje byla používána reálná zařízení. Přehled testovacích zařízení, jež měl vývojář k dispozici, je zachycen v tabulce 5.1. Při vývoji Android aplikací je důležité jejich testování na zařízeních různých velikostí, rozlišení a verzí OS, jelikož některé funkce jsou implementovány jinak pro různé verze API a také UI se může lišit na obrazovkách různých velikostí.

Tabulka 5.1: Testovací zařízení

Jméno zařízení	Číslo modelu	Verze Android (API)	Rozlišení (px)	Velikost obrazovky (in)
Samsung Galaxy A3 (2016)	SM-A310F	7.0 (24)	720 x 1280	4.70
Samsung Galaxy A5 (2016)	SM-A510F	7.0 (24)	1080 x 1920	5.20
Samsung Galaxy S6 Edge	SM-G925F	7.0 (24)	1440 x 2560	5.10
Samsung Galaxy S10e	SM-G970F	10 (29)	1080 x 2280	5.74

5.1 Leak Canary

Leak Canary je nástroj pro testování správy paměti, který dokáže detekovat její úniky[30], jež mohou vést k pádům aplikace způsobeným nedostatkem paměti [44]. Leak Canary dokáže rozlišit mezi chybami způsobenými knihovnami třetích stran a chybami vlastní implementace. Chybou se v tomto případě myslí držení reference na objekt, který již není potřeba a má být zničen, avšak v důsledku této reference nemůže být paměť alokovaná pro daný objekt uvolněna.

První zmíněné jsou označené jako Library Leaks. Jedná se o známé chyby, které však nelze vždy opravit. Danou chybu je možné nahlásit a požadovat její

[30] z anglického memory leaks

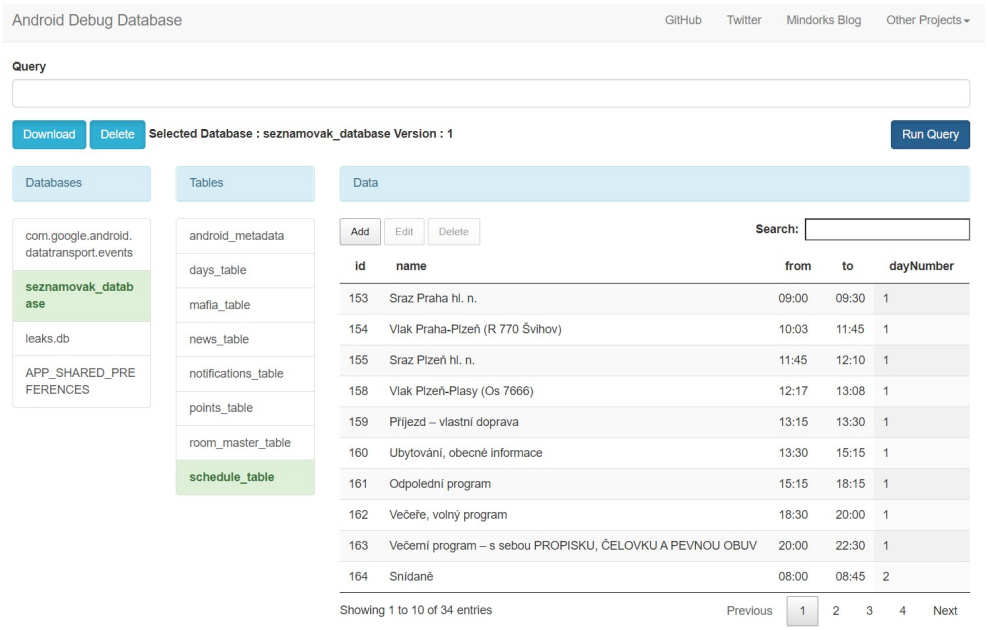
odstranění [45]. S odstraněním úniků paměti druhého typu napomáhá právě tento nástroj. S jeho pomocí lze vzniklou chybu trasovat, zjistit, jaký objekt ji způsobil, a následně chybu opravit.

5.2 Android Debug Database

Android Debug Database (ADD) je knihovna, která umožňuje zobrazení Shared Preferences a všech databází společně se všemi obsaženými daty. S využitím ADD lze však také editovat obsah databáze a Shared Preferences, spouštět dotazy nad databází či přidávat záznamy typu klíč–hodnota do Shared Preferences. [46]

Použití této knihovny je velmi jednoduché. Je nutné na ni pouze přidat závislost do souboru build.gradle. Po spuštění aplikace z Android Studia pak stačí otevřít webový prohlížeč na URL adrese zobrazené v záložce Logcat. Ukázka obsahu databáze je na obrázku 5.1.

V rámci vývoje aplikace Seznamovák byl tento nástroj využíván pro zjištění obsahu DB a Shared Preferences a zda jsou všechna data správně mazána při odhlášení uživatele.



Obrázek 5.1: Obsah databáze zobrazený pomocí ADD

5.3 Timber

Timber je knihovna, jejímž autorem je vývojář Jake Warthon a která slouží ke snadnému logování. Její použití je podobné použití Android třídy Log, avšak s několika výhodami.

Pokud vývojář používá k logování třídu Log, je nucen všechna volání metod této třídy z kódu odstranit před jejím nahráním na Google Play Store, aby si uživatel nemohl jednotlivé výpisy zobrazit. Toto je důležité především, pokud by logy obsahovaly citlivá data. Timber tuto práci nevyžaduje, jelikož lze přímo definovat, že logy mají být vytvářeny pouze v rámci debugovací verze aplikace[31]. To je učiněno nejlépe ihned ve volání funkce *onCreate* aplikace, v níž lze definovat podmínku, kdy má být logování uskutečněno. Použití knihovny Timber pouze na debugovací verzi aplikace je ukázáno výpisem kódu 5.1.

Timber obsahuje stejné metody jako třída Log, určující prioritu (např. *Timber.e()* značí Error, kdežto *Timber.i()* znamená Info apod.). V rámci volání těchto metod však není třeba specifikovat TAG. Timber sám pozná, v rámci které třídy byl log vytvořen. Tato skutečnost tedy odstraňuje nutnost definovat proměnnou TAG v každé třídě, v níž chce vývojář log vytvořit, avšak Timber umožňuje explicitně definovat proměnnou TAG, a tedy změnit její název. [47]

5.4 Test Lab

Test Lab je nástroj pro testování aplikace na platformě Firebase popsané v sekci 4.2.1. Umožňuje otestovat aplikaci na různých zařízeních s různými verzemi API. Zařízení mohou být virtuální či fyzická, přičemž existuje denní

[31] verze určená k ladění a vývoji aplikace

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()

        if (BuildConfig.DEBUG) {
            Timber.plant(DebugTree())
        }
        //...
    }
    //...
}
```

Výpis kódu 5.1: Ukázka vytvoření debugovacího stromu

omezení na počet provedených testů na obou typech zařízeních. Pomocí tohoto nástroje lze simulovat chování aplikace v reálném prostředí. [48]

5.4.1 Robo Test

V rámci aplikace Seznamovák byly použity Robo testy [49]. Robo test je automatický test, při kterém dochází k simulaci chování uživatele, procházení aplikací, klikání na tlačítka či vyplňování editovatelných polí.

Pro spuštění testu je třeba nahrát APK aplikace. Dále vývojář nakonfiguruje zařízení, na kterém má test běžet. Lze zvolit více zařízení s odlišným rozlišením a verzí OS. Jak již bylo zmíněno výše, v bezplatné verzi však existuje omezení na použití maximálně 5 fyzických a 10 virtuálních zařízení za den. Je také možné zvolit orientaci zařízení, zdali má být zařízení na výšku či na šířku[32]. Dále je možné určit, jak mají být vyplněna určitá editovatelná pole či zda má být kliknuto na určitý prvek UI. V rámci testování aplikace Seznamovák bylo nastaveno, aby pole s identifikátorem *email_input_field* bylo vyplněno správným emailovým účtem, a tedy aby došlo k přihlášení do aplikace. Dále byl stejný postup použit k zadání správných souřadnic.

Výsledkem testu je video zachycující obrazovku zařízení, snímky obrazovky, graf průchodu jednotlivými obrazovkami, informace o výkonu (resp. o práci s pamětí, výkonu CPU a použití internetové sítě), dostupnost jednotlivých prvků na základě barev či velikosti písma a v neposlední řadě údaj, zda byl test úspěšně dokončen. Vybraná videa a grafy se nachází na přiloženém CD. [50]

5.5 Uživatelské testování

Uživatelské testování aplikace slouží k získání zpětné vazby od jejích potenciálních uživatelů. V rámci těchto testů je ověřeno, zda je ovládání a pohyb v aplikaci intuitivní. Uživatelské testování aplikace Seznamovák proběhlo po dokončení implementace. V průběhu vývoje byla aplikace poskytnuta pouze několika testerům za účelem zjištění, zda je design aplikace srozumitelný.

K přípravě uživatelského testování bylo využito rad dostupných na webových stránkách společnosti AITOM [51]. Zde je popsáno uživatelské testování webu, avšak většinu doporučení lze aplikovat i na testování mobilní aplikace.

5.5.1 Distribuce aplikace

Pro distribuci aplikace mezi testery byla využita platforma Firebase. Distribuce probíhá nahráním APK a zadáním emailových adres, na které je následně zaslána pozvánka k instalaci dané aplikace. Tyto emaily lze sdružovat do skupin. Je tedy možné vytvořit například skupinu testerů organizátorů

[32] anglicky hovoříme o orientaci obrazovky portrait a landscape

a účastníků. Je také možné na platformu nahrát více verzí aplikace. V případě aplikace Seznamovák však byly pozvánky zasílány jednotlivě, vždy na začátku testování, aby testeři neměli možnost aplikaci vyzkoušet a prozkoumat ještě před uskutečněním samotného uživatelského testování.

5.5.2 Testeři

Jedním z prvních kroků uživatelského testování je výběr vhodných testerů. Vhodnými jsou myšleny osoby, které reprezentují určitou skupinu potenciálních uživatelů aplikace. Aplikace Seznamovák je určena organizátorům této akce a jejím účastníkům, což jsou budoucí studenti prvního ročníku FIT ČVUT. Vhodnými kandidáty jsou potom osoby, které se neúčastnily vývoje a setkávají se s aplikací prvně právě během testování. V případě organizátorů je však tento požadavek nesplnitelný, jelikož se všichni setkali s původní verzí aplikace Seznamovák v rámci předchozích ročníků akce.

Mezi testery by tedy měly být zastoupeny obě skupiny – účastníci i organizátoři. Zastoupena by taktéž měla být obě pohlaví, jelikož každé má obvykle jiná hodnotící kritéria. Tímto způsoben vzniknou 4 role potenciálních uživatelů: žena-účastnice, žena-organizátorka, muž-účastník a muž-organizátor. Dle [52] je ideální počet testerů 5. V případě uživatelského testování aplikace Seznamovák byly zastoupeny všechny výše zmíněné role a jeden účastník navíc, přičemž všichni testeři se akce Seznamovák v minulosti účastnili a 2 z nich jsou jejími současnými organizátory. Tato skutečnost byla výhodou, jelikož nebylo třeba testerům popisovat akci samotnou. V rámci zpětné vazby navíc mohli čerpat z vlastních zkušeností při odpovídání na otázku, zda aplikace postrádá určitou funkci, jež by na Seznamováku využili. Dva testeři se s aplikací setkali prvně během testování. Zbylí 3 používali původní verzi, avšak pouze minimálně.

5.5.3 Scénáře

Pro testování aplikace je třeba vytvořit scénáře, které v optimálním případě pokryjí co nejvíce možných případů užití definovaných v sekci 2.4. Scénáře by neměly obsahovat nápovědu či návod, jak má být dosaženo jejich splnění. Testeři dostali za úkol splnit tyto scénáře:

1. Na Seznamovák jste se registroval/a pomocí emailového účtu exampleMail@gmail.com. Přihlaste se do aplikace Seznamovák.

2. Zjistěte, který program právě probíhá a který bude následovat. V kolik hodin programy začínají a končí?

3. Zjistěte, jaký je váš bodový zisk ze hry Infiltrátor.

4. Zjistěte, jaký je váš celkový počet bodů ze všech her.

5. Nechcete dostávat notifikace týkající se přidělování bodů. Vypněte je.
6. Zjistěte, v kolik hodin je druhý den večeře.
7. Zjistěte, zda se od 1.11. neobjevily nějaké novinky.
8. Setmělo se a z bílého pozadí aplikace vás akorát bolí oči. Změňte motiv aplikace na tmavý.
9. Po delší době jste se připojil/a k internetu. Aktualizujte všechna data aplikace.
10. Zjistěte, kdo je váš cíl v rámci hry Kolíčkovaná.
11. Podařilo se vám zabít svůj cíl, hráče AB[33], v rámci hry Kolíčkovaná. AB měl za úkol zabít hráče se jménem XY. Realizujte zabití.
12. V rámci hry Chuck jste dostal/a za úkol dostat se na místo o souřadnicích 49°56,345 s. š. a 13°22,424 v. d. Vyhledejte dané místo.
13. Seznamovák skončil. Odhlaste se z aplikace Seznamovák.

Tyto scénáře byly použity v případě testerů-účastníků. V případě organizátorů byly vynechány scénáře týkající se přidělování bodů a jejich notifikací.

5.5.4 Průběh testování

Vzhledem k vládním nařízením týkajících se koronavirové pandemie nebylo možné provést uživatelské testování osobně, a bylo tedy nutné jej realizovat prostřednictvím vhodné komunikační platformy. Před samotným testováním byly testerům poslány emaily s informacemi, jak bude testování probíhat, dotaz na mobilní zařízení, které bude použito v rámci testování, a jaká je jeho verze OS. Dále byli testeři požádáni o instalaci aplikace na nahrávání obrazovky a v neposlední řadě byli taktéž požádáni o udělení souhlasu s nahráváním průběhu testování.

V předem dohodnutý čas byl s každým testerem zahájen videohovor prostřednictví platformy MS Teams, jež byl nahráván pro pozdější zpracování. Následně byla testerovi na email zaslána pozvánka ke stažení aplikace Seznamovák prostřednictvím platformy Firebase, konkrétně služby App Distribution. Po úspěšné instalaci tester spustil nahrávání obrazovky mobilního zařízení. Pomocí sdílení obrazovky byla testerovi ukázána prezentace, jejíž snímky obsahovaly jednotlivé scénáře, jež musel splnit. Jednotlivé scénáře jsou popsány v sekci 5.5.3. Tester byl vyzván k přečtení každého scénáře před jeho plněním a také komentování svého chování, na jaké ikony kliká, co se mu právě ukazuje apod.

[33]Během testování byly zkratky AB a XY nahrazeny skutečnými jmény.

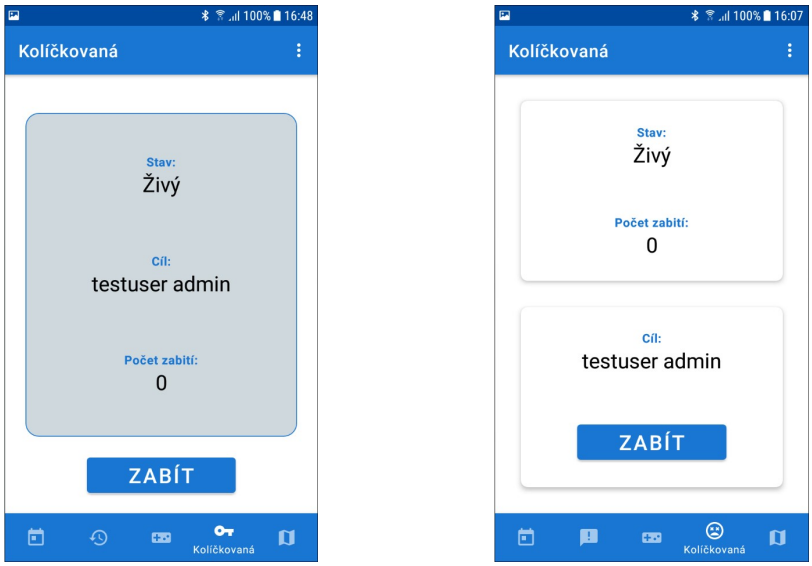
Po úspěšném dokončení všech úkolů byl tester požádán o opětovné přihlášení do aplikace a bylo vyzkoušeno zasílání notifikací. Následoval prostor pro zpětnou vazbu, v rámci které se mohl tester vyjádřit ke vzhledu aplikace, jejímu výkonu, zda se mu aplikace zdá intuitivní a uživatelsky přívětivá či zda měl s některým scénářem problém při jeho plnění.

5.5.5 Výsledky testů

Testování měla hladký průběh a testeři zdárně splnili všechny scénáře bez větších problémů. Jediné větší nedorozumění nastalo v případě Kolíčkované. Někteří testeři považovali zobrazený stav a počet zabití za atributy cíle, jelikož se tyto informace nacházely na společné kartě. Na základě tohoto poznatku byly informace týkající se uživatele odděleny do samostatné karty. Ukázky obrazovek hry Kolíčkovaná testované i výsledné verze aplikace se nachází na obrázcích 5.2a a 5.2b.

V rámci uživatelského testování byla použita zařízení různých verzí OS, od nejnižší podporované verze Android 5 po verzi 10, díky čemuž bylo vyzkoušeno zasílání notifikací na zařízení, jež podporují notifikační kanály, jakožto i na zařízení, která touto funkcionalitou nedisponují.

V jednom případě tester zpočátku netušil, jak vyhledat souřadnice, jelikož si myslel, že FAB s lupou představuje přiblížení mapy, avšak toto nedorozumění bylo zcela ojedinělé a po krátké chvíli tester pochopil původní význam tohoto tlačítka.



(a) Testovaná verze (b) Výsledná verze

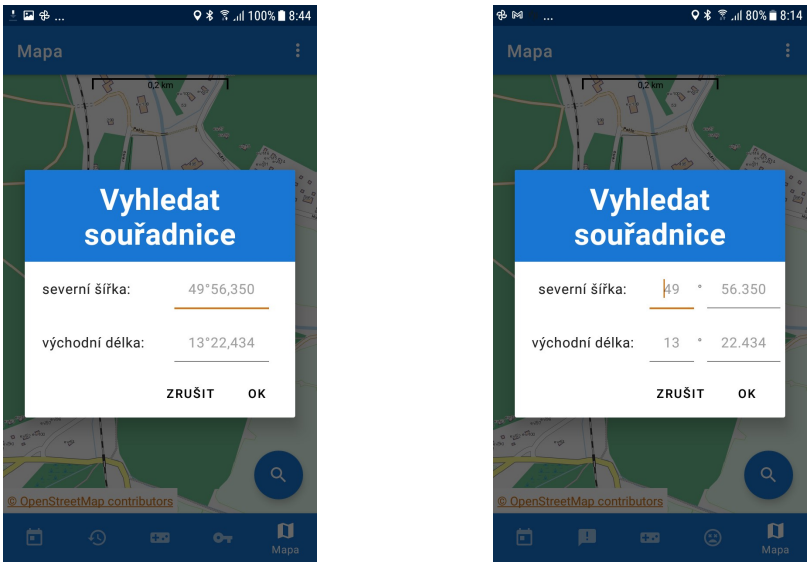
Obrázek 5.2: Obrazovka hry Kolíčkovaná

Během testování však bylo vypozorováno, že zadávání souřadnic zabere uživateli delší čas, jelikož je třeba zadat jeden konkrétní formát, který obsahuje znak stupňů °, přičemž většině testerů chvíli trvalo, než daný symbol na klávesnici nalezla. Vyjádřili se proto, že by bylo vhodné, aby nebylo zapotřebí symbol zadávat. Na základě tohoto požadavku bylo dialogové okno pro vyhledávání upraveno do podoby, ve které jsou zadávány stupně a minuty zvlášť, bez nutnosti psaní znaku pro stupně. Tento přístup má taktéž výhodu z hlediska implementace, jelikož je možné specifikovat, že zadaná hodnota musí být číslo. V takovém případě je uživateli zobrazena pouze numerická klávesnice a zadávání souřadnic je proto pohodlnější, bezpečnější a rychlejší. Obrazovky realizující vyhledávání souřadnic testované verze aplikace a výsledné upravené aplikace se nachází na obrazcích 5.3a a 5.3b.

Dále byly nahrazeny ikony záložek Novinky a Kolíčkovaná v rámci Bottom Navigation Bar, aby více odpovídaly daným doménám, přestože testeři neměli problémy s asociací jednotlivých ikon s odpovídajícími doménami, avšak vyjádřili se, že by tyto změny uvítali.

V případě scénáře týkajícího se aktualizace dat si testeři nebyli jistí, zda skutečně došlo k aktualizaci dat, jelikož po úspěšné aktualizaci pouze zmizí Progress Bar[34]. V případě neúspěšné aktualizace je uživateli zobrazen Toast s chybovou zprávou. Na dotaz, zda by měl být zobrazen Toast i v případě úspěchu, však testeři z větší části odpověděli, že intuitivně pochopili, že se data aktualizovala úspěšně a že stejný přístup je realizován i v jiných Android

[34] načítací kolečko



(a) Testovaná verze (b) Výsledná verze

Obrázek 5.3: Obrazovka realizující vyhledávání souřadnic

aplikacích. Proto v rámci zajištění konzistence s ostatními aplikacemi bylo toto chování ponecháno beze změn.

Testeři dále ocenili možnost změny motivu aplikace a celkově hodnotili vzhled, rychlost i funkce aplikace kladně. Po provedení výše uvedených změn byla aplikace testerům opět poskytnuta ke kontrole, zda provedené změny odpovídají jejich požadavkům a zdali upravená verze zdárně řeší problémy, jež během testování nalezli. Testeři uvedli, že rozdělení Kolíčkované do dvou částí je přehlednější, nové ikonky více vystihují dané domény a vyhledávání souřadnic bez nutnosti zadávání znaku pro stupně taktéž uvítali. Čtyři videa obrazovek jednotlivých testů se nacházejí na přiloženém CD. V jednom případě se testerovi nepodařilo záznam pořídit. Na přiloženém CD se taktéž nachází prezentace se scénáři užitými během testování.

Závěr

Hlavním cílem této práce bylo vytvoření mobilní aplikace Seznamovák pro potřeby organizátorů a účastníků přípravného kurzu Fakulty informačních technologií ČVUT v Praze. Mezi dílčí cíle patřila analýza původního řešení, stanovení požadavků kladených na aplikaci na základě komunikace s organizátory a účastníky kurzu, implementace nového řešení pro OS Android a v neposlední řadě také podrobení aplikace uživatelským testům. Mezi hlavní rozšíření nového řešení oproti původní verzi patří přidání hry Kolíčkovaná, možnost změny módu, funkční offline mapa a zasílání notifikací harmonogramu taktéž offline.

Na začátku práce byla provedena analýza existující aplikace a byl vytvořen dotazník pro uživatele této verze, z něhož vzešly požadavky kladené na aplikaci a také problémy, kterým bylo třeba se vyvarovat během implementace nového řešení. Na analýzu bylo navázáno návrhem nové aplikace, v rámci kterého byly vytvořeny prototypy obrazovek, jež využívají principy Material Design, a byla zvolena architektura vhodná pro tuto aplikaci. Dále byla na základě návrhu aplikace implementována v programovacím jazyce Kotlin. V rámci implementace byla taktéž vytvořena aplikace Magistrovák, jež má podobné funkce a stejný základ jako aplikace Seznamovák. Výsledná aplikace byla podrobena uživatelským testům, na jejichž základě byly objevené nedostatky opraveny ke spokojenosti uživatelů aplikace.

Mobilní aplikace Seznamovák bude sloužit organizátorům a účastníkům úvodního kurzu určeného studentům nastupujícím do bakalářského studia FIT ČVUT v Praze. Jejím hlavním přínosem je usnadnění určitých úkonů, jako například hraní hry Kolíčkovaná či upozorňování na změny v harmonogramu, a lepší informovanost účastníků díky stálému přístupu k programu celého kurzu, novinkám a počtu získaných bodů z jednotlivých her.

Do budoucna jsou možná další rozšíření aplikace. V úvahu přichází sloučení s aplikací Zpěvník, přidání kontaktů na hlavní organizátory či možnost uzpůsobení barevného vzhledu uživatelovým preferencím. Dále je možné použití této aplikace i na akcích podobného typu, k čemuž je aplikace Seznamovák uzpůsobena díky vhodné implementaci s ohledem na požadavek snadné

modifikace a rozšiřitelnosti. V blízké budoucnosti bude aplikace poskytnuta organizátorům akce Seznamovák, kteří spravují její původní verzi, jež bude nahrazena na Google Play Store tímto novým řešením.

Seznam použité literatury

1. *Seznamovák* [online]. Google Play, 2020 [cit. 2020-01-04]. Dostupné z: https://play.google.com/store/apps/details?id=cz.cvut.fit.seznamovak.mobile.
2. *Magistrovák* [online]. Google Play, 2020 [cit. 2020-09-26]. Dostupné z: https://play.google.com/store/apps/details?id=cz.cvut.fit.magistrovak.mobile&hl=cs.
3. *AppBar* [online]. Google Inc. Android Developers, 2019 [cit. 2019-11-23]. Dostupné z: https://developer.android.com/training/appbar.
4. *Zpěvník* [online]. Google Play, 2020 [cit. 2020-01-04]. Dostupné z: https://play.google.com/store/apps/details?id=cz.seznamovak.seznamovkzpvn.
5. WIEGERS, K.; BEATTY, J. *Software Requirements*. Pearson Education, 2013. Developer Best Practices. ISBN 978-0-7356-7962-7. Dostupné také z: https://books.google.cz/books?id=nbpCAwAAQBAJ.
6. *Support different platform versions* [online]. Google Inc. Android Developers, 2019 [cit. 2019-11-23]. Dostupné z: https://developer.android.com/training/basics/supporting-devices/platforms.
7. *Distribution dashboard* [online]. Google Inc. Android Developers, 2019 [cit. 2019-11-23]. Dostupné z: https://developer.android.com/about/dashboards/index.html.
8. MLEJNEK, Jiří. *Softwarové inženýrství 1: Analýza a sběr požadavků - případy užití* [online]. 2020 [cit. 2020-09-16]. Dostupné z: https://moodle-vyuka.cvut.cz/pluginfile.php/312257/mod_resource/content/4/03.prednaska.pdf [Soubor přístupný po přihlášení do sítě ČVUT – kopie souboru uložena na přiloženém CD].
9. *Minimum SDK version* [online]. Google Inc. Android Developers, 2020 [cit. 2020-08-27]. Dostupné z: https://twitter.com/minsdkversion.

10. LACKO, Ľuboslav. *Mistrovství-Android: kompletní průvodce vývojáře*. 1. vydání. Brno: Computer Press, 2017. ISBN 978-80-251-4875-4.

12. *Android's commitment to Kotlin* [online]. Android Developers Google Blog, 2019 [cit. 2020-09-14]. Dostupné z: `https://android-developers.googleblog.com/2019/12/androids-commitment-to-kotlin.html`.

13. *Kotlin Foundation* [online]. Kotlang [cit. 2020-09-14]. Dostupné z: `https://kotlinlang.org/foundation/kotlin-foundation.html`.

14. *Udacity* [online]. Udacity, Inc., 2020 [cit. 2020-09-14]. Dostupné z: `https://www.udacity.com/`.

15. *Kotlin First* [online]. Google Inc. Android Developers, 2020 [cit. 2020-09-14]. Dostupné z: `https://developer.android.com/kotlin/first`.

16. *Bottom navigation* [online]. Google Inc., 2020 [cit. 2020-09-15]. Dostupné z: `https://material.io/components/bottom-navigation/#usage`.

17. *Buttons: floating action button* [online]. Google Inc., 2020 [cit. 2020-09-15]. Dostupné z: `https://material.io/components/buttons-floating-action-button`.

18. *Dialogs* [online]. Google Inc., 2020 [cit. 2020-09-15]. Dostupné z: `https://material.io/components/dialogs#usage`.

19. *Android notifications* [online]. Google Inc., 2020 [cit. 2020-10-07]. Dostupné z: `https://material.io/design/platform-guidance/android-notifications.html#usage`.

20. *Proto.io* [online]. Protoio, Inc., 2020 [cit. 2020-09-17]. Dostupné z: `https://proto.io/`.

22. *Guide to app architecture* [online]. Google Inc. Android Developers, 2020 [cit. 2020-09-15]. Dostupné z: `https://developer.android.com/jetpack/guide`.

23. *Android Architecture Components* [online]. Google Inc. Android Developers, 2020 [cit. 2020-09-24]. Dostupné z: `https://developer.android.com/topic/libraries/architecture`.

24. *View Binding* [online]. Google Inc. Android Developers, 2020 [cit. 2020-09-20]. Dostupné z: `https://developer.android.com/topic/libraries/view-binding#findviewbyid`.

25. *Data Binding* [online]. Google Inc. Android Developers, 2020 [cit. 2020-09-20]. Dostupné z: `https://developer.android.com/topic/libraries/data-binding`.

26. MCQUILLAN, Sean. *Use view binding to replace findViewById* [online]. 2020 [cit. 2020-09-25]. Dostupné z: `https://medium.com/androiddevelopers/use-view-binding-to-replace-findviewbyid-c83942471fc`.

27. *Navigation* [online]. Google Inc. Android Developers, 2020 [cit. 2020-09-24]. Dostupné z: https://developer.android.com/guide/navigation.

28. *Application Fundamentals* [online]. Google Inc. Android Developers, 2020 [cit. 2020-09-26]. Dostupné z: https://developer.android.com/guide/components/fundamentals#Components.

30. SURWASE, Rohit. *Best Architecture For Android* [online]. ProAndroidDev [cit. 2020-09-14]. Dostupné z: https://proandroiddev.com/best-architecture-for-android-mvi-livedata-viewmodel-71a3a5ac7ee3.

31. MUNTENESCU, Florina. *Android Architecture Patterns Part 1: Model-View-Controller* [online]. Medium [cit. 2020-09-14]. Dostupné z: https://medium.com/upday-devs/android-architecture-patterns-part-1-model-view-controller-3baecef5f2b6.

32. MUNTENESCU, Florina. *Android Architecture Patterns Part 2: Model-View-Presenter* [online]. Medium [cit. 2020-09-14]. Dostupné z: https://medium.com/upday-devs/android-architecture-patterns-part-2-model-view-presenter-8a6faaae14a5.

33. MUNTENESCU, Florina. *Android Architecture Patterns Part 3: Model-View-ViewModel* [online]. Medium [cit. 2020-09-14]. Dostupné z: https://medium.com/upday-devs/android-architecture-patterns-part-3-model-view-viewmodel-e7eeee76b73b.

34. *App data and files* [online]. Google Inc. Android Developers, 2020 [cit. 2020-09-21]. Dostupné z: https://developer.android.com/guide/topics/data.

35. *Download Android Studio* [online]. Google Inc. Android Developers, 2020 [cit. 2020-09-17]. Dostupné z: https://developer.android.com/studio.

36. *Firebase Cloud Messaging* [online]. Google Inc., 2020 [cit. 2020-09-30]. Dostupné z: https://firebase.google.com/docs/cloud-messaging.

37. *Mobile Atlas Creator* [software]. Source Forge, 2020 [cit. 2020-11-26]. Dostupné z: https://mobac.sourceforge.io/.

38. SKALA, Jan. *Offline mapy pro Android* [online] [cit. 2020-09-19]. Dostupné z: https://osm.paws.cz/ [V práci byla použita mapa České republiky czech_republic.map].

39. *Autorská práva a licence* [online]. OpenStreetMap Foundation, 2020 [cit. 2020-09-27]. Dostupné z: https://www.openstreetmap.org/copyright.

40. Android Navigation Componemt: How save fragment state. In: *Stackoverflow* [online]. 2019 [cit. 2020-09-27]. Dostupné z: https://stackoverflow.com/a/56195906/13488918.

41. *Navigation Advanced Sample* [online]. The Android Open Source Project, Inc., 2018 [cit. 2020-09-27]. Dostupné z: https://github.com/android/architecture-components-samples/tree/master/NavigationAdvancedSample.

42. *Support multiple back stacks for Bottom tab navigation* [online] [cit. 2020-09-27]. Dostupné z: https://issuetracker.google.com/issues/80029773.

43. *Settings* [online]. Google Inc. Android Developers, 2020 [cit. 2020-10-22]. Dostupné z: https://developer.android.com/guide/topics/ui/settings.

44. *Leak Canary* [online]. Square, Inc., 2015 [cit. 2020-10-04]. Dostupné z: https://square.github.io/leakcanary/.

45. *LeakCanary: How LeakCanary works* [online]. Square, Inc., 2015 [cit. 2020-10-04]. Dostupné z: https://square.github.io/leakcanary/fundamentals-how-leakcanary-works/#4-categorizing-leaks.

46. SHEKHAR, Amit. *Android-Debug-Database: A library for debugging android databases and shared preferences* [online]. 2019 [cit. 2020-10-04]. Dostupné z: https://github.com/amitshekhariitbhu/Android-Debug-Database.

47. RAJORA, Sidhant. *Better Logging In Android Using Timber* [online]. 2018 [cit. 2020-10-20]. Dostupné z: https://medium.com/mindorks/better-logging-in-android-using-timber-72e40cc2293d.

48. *Firebase Test Lab* [online]. Google Inc., 2020 [cit. 2020-11-01]. Dostupné z: https://firebase.google.com/docs/test-lab?authuser=0.

49. *Róbert Selvek* [online]. LinkedIn, 2020 [cit. 2020-12-04]. Dostupné z: https://www.linkedin.com/in/r%C3%B3bert-selvek-5277b3191/ [LinkedIn Profile].

50. *Getting started with Robo tests* [online]. Google Inc., 2020 [cit. 2020-11-01]. Dostupné z: https://firebase.google.com/docs/test-lab/android/robo-ux-test?authuser=0.

51. *Uživatelské testování krok za krokem* [online]. AITOM Digital s.r.o., 2020 [cit. 2020-10-08]. Dostupné z: https://www.pojdmetestovat.cz/file/16.

52. NIELSEN, Jakob. *How Many Test Users in a Usability Study?* [online]. 2020 [cit. 2020-10-27]. Dostupné z: https://www.nngroup.com/articles/how-many-test-users/.

Seznam použitých obrázků

11. *Android Studio. Version 4.0.1* [software]. Google, 2020 [cit. 2020-09-15]. Dostupné z: `https://developer.android.com/studio`. [Distribuci verzí lze zobrazit při tvorbě nového projektu].

21. *Android Jetpack* [online obrázek]. Google, Inc., 2018 [cit. 2020-09-25]. Dostupné z: `https://raw.githubusercontent.com/android/sunflower/master/screenshots/jetpack_donut.png`.

29. *Recommended App Architecture* [online obrázek]. Google Inc. Android Developers [cit. 2020-09-15]. Dostupné z: `https://developer.android.com/topic/libraries/architecture/images/final-architecture.png`.

Příloha A

Seznam použitých zkratek

AC Architecture Components

ADD Android Debug Database

ADT Android Development Tools

API Application Programming Interface

APK Android Package

CPU Central Processing Unit

DB Database

FAB Floating Action Button

FCM Firebase Cloud Messaging

HTTP Hypertext Transfer Protocol

IDE Integrated Development Environment

JSON JavaScript Object Notation

MOBAC Mobile Atlas Creator

MVC Model-View-Controller

MVP Movel-View-Presenter

MVVM Model-View-ViewModel

OS Operating System

REST Representational State Transfer

SOC Separation of Concerns

UC Use Case

UI User Interface

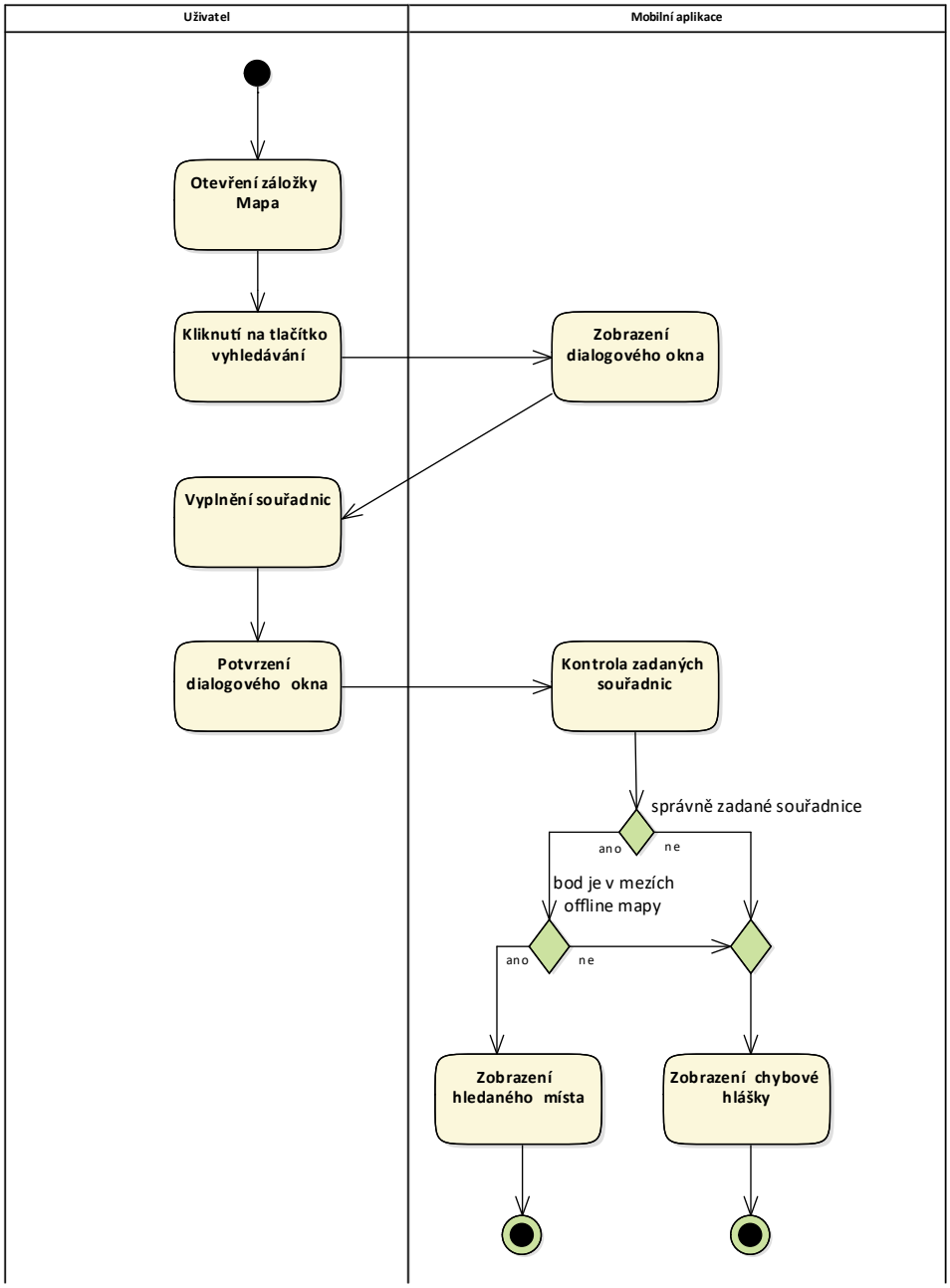
URL Uniform Resource Locator

UX User Experience

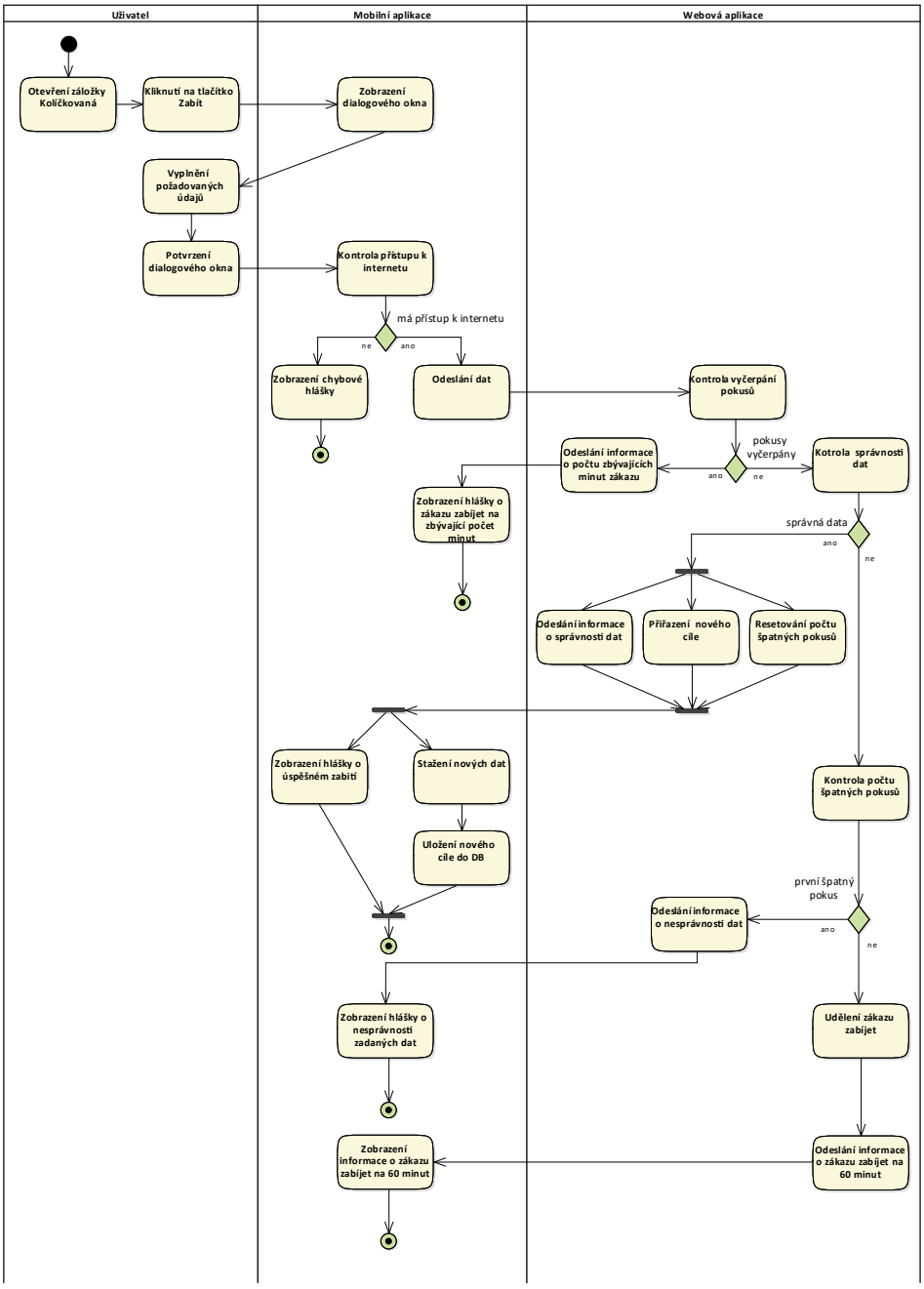
XML Extensible Markup Language

Příloha B

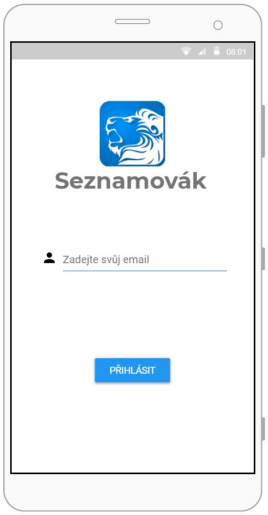
Obrázky



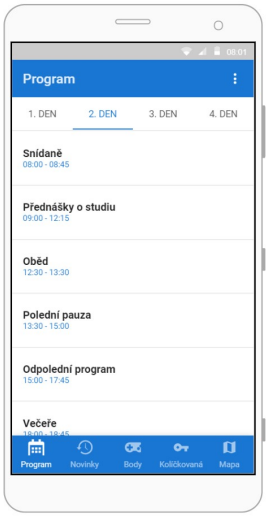
Obrázek B.1: Diagram aktivit pro vyhledávání souřadnic



Obrázek B.2: Diagram aktivit pro „zabití“ v rámci Kolíčkované

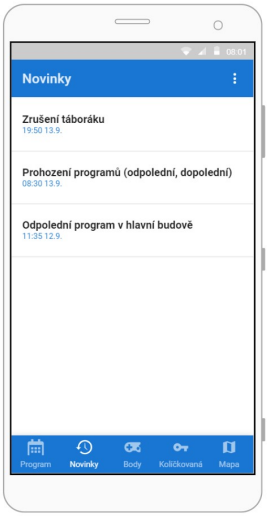


(a) Přihlášení

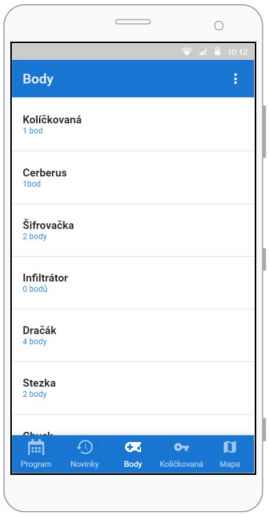


(b) Program

Obrázek B.3: Návrhy vstupních obrazovek

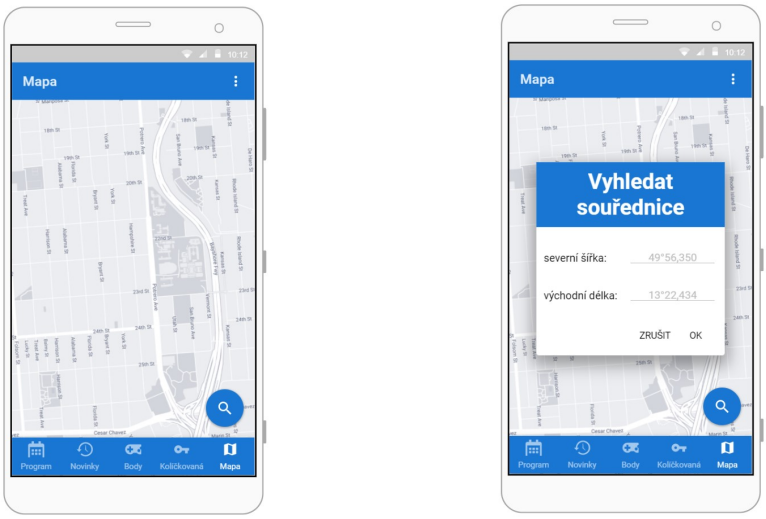


(a) Novinky



(b) Body

Obrázek B.4: Návrhy obrazovek Novinky a Body

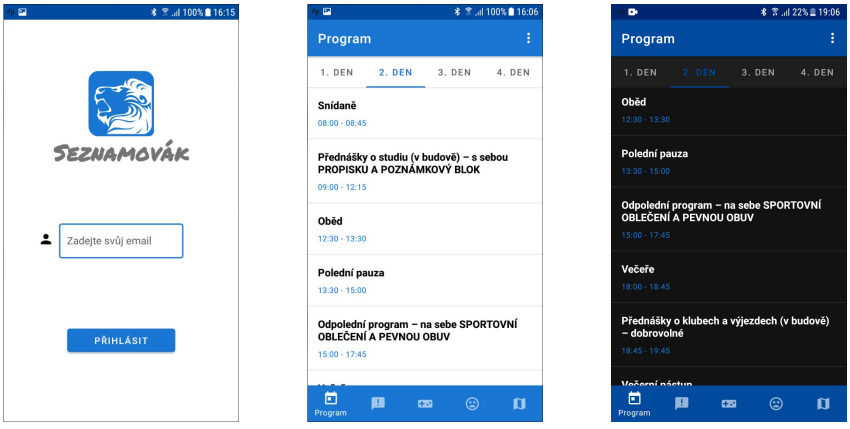


(a) Mapa (b) Vyhledávání souřadnic

Obrázek B.5: Návrhy obrazovek Mapa a Vyhledávání souřadnic

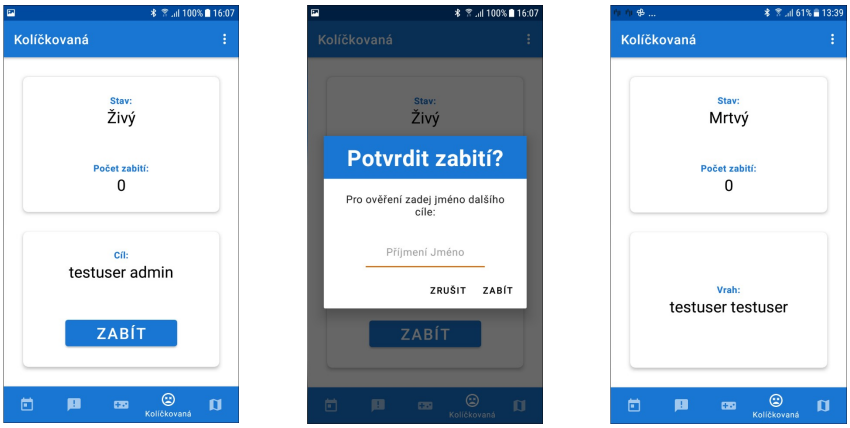
Příloha C

Snímky výsledné aplikace



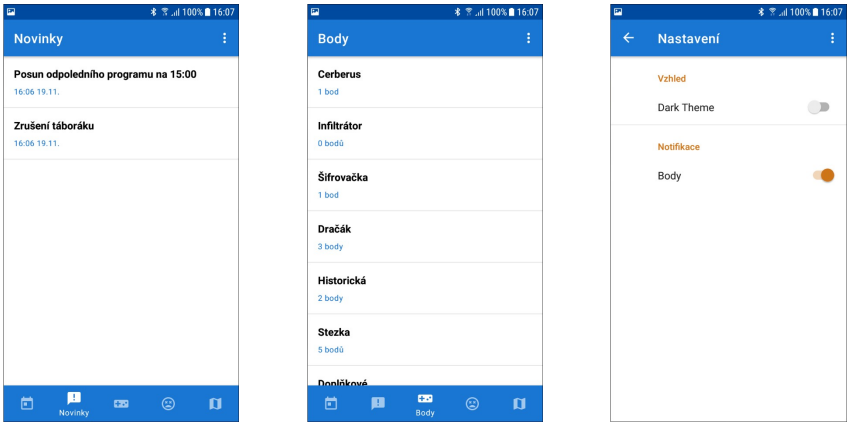
(a) Přihlášení (b) Program (c) Dark Theme

Obrázek C.1: Vstupní obrazovky



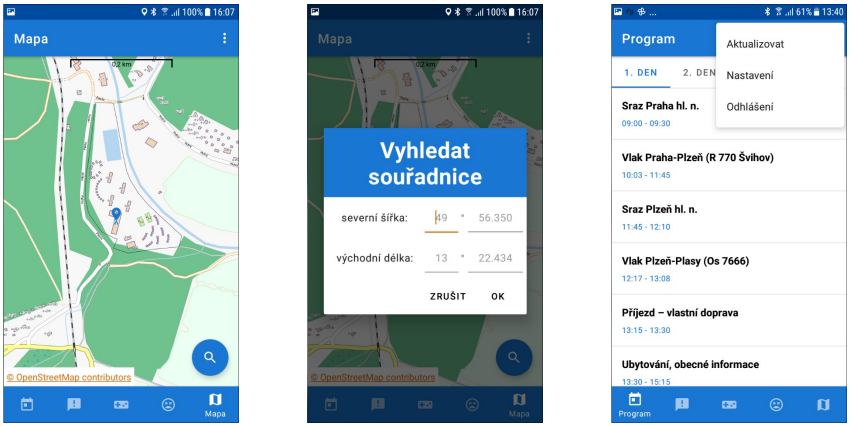
(a) Stav Živý (b) Zabíjení (c) Stav Mrtvý

Obrázek C.2: Hra Kolíčkovaná



(a) Novinky (b) Body (c) Nastavení

Obrázek C.3: Obrazovky Novinky, Body a Nastavení



(a) Mapa (b) Vyhledávání (c) Overflow Menu

Obrázek C.4: Obrazovky Mapa, Vyhledávání souřadnic a obsah Overflow Menu

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `apk` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `tests` ..... adresář obsahující výsledky testů
  - `userTests` ..... adresář s daty uživatelského testování
  - `testLab` ..... adresář s výsledky testů z Test Lab
- `screens` ..... adresář souborů zachycující obrazovky aplikace
  - `appScreens.pdf` ..... snímky výsledné aplikace
  - `wireframes.pdf` ..... exportované návrhy obrazovek
- `sources` ..... adresář zdrojů odkazovaných v této práci
  - `questionnaireResults.xlsx` ..... výsledky dotazníku
  - `SI1-prednaska.pdf` ..... přednáška předmětu SI1
- `videoRecords` ..... adresář videí použití aplikace
  - `SeznamovakUsage.mp4` ..... video použití aplikace Seznamovák
  - `MagistrovakUsage.mp4` ..... video použití aplikace Magistrovák