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

Bakalářská práce

Mobilní hra s využitím GPS navigace

Plzeň 2018 Zdeněk Ziegler

Místo této strany bude zadání práce.

Prohlášení

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

V Plzni dne 26. června 2018

Zdeněk Ziegler

Abstract

Mobile game using GPS navigation.

Goal of this work is the creation of a game for mobile devices with Android operating system, which will use GPS of mobile device, thanks to it application will scan player's location. In the game player will be able to choose a mission from missions offer. Mission's assignment contains geographical points, which player has to go through in given order or randomly. Fulfillment of these missions players collect points that determinate thier position in result list. In the theoretical part of this work, I focus on analysis and comparation of similar applications in the market, designing my own solution of the game and describing the resulting solution. This work contains also chapter with suggestiions for possible extensions of application and chapter with testing results of the resulting application as well.

Abstrakt

Cílem této práce je vytvoření hry pro mobilní zařízení s operačním systémem Android, která bude pracovat s GPS mobilního zařízení, díky kterému bude snímat hráčovu geografickou polohu. Ve hře si hráč bude moct vybírat z nabídky misí, které obsahují ve svém zadání geografické body, které hráč má za úkol projít v zadaném či libovolném pořadí. Splněním těchto misí hráči sbírají body, které určují jejich umístění ve výsledkové listině. V teoretické části se věnuji rozboru a porovnání podobných aplikací na trhu, návrhu vlastního řešení hry a také popisem výsledného řešení. Práce dále obsahuje kapitolu s návrhy pro možné rozšíření aplikace, a také kapitolu s výsledky testování výsledné aplikace.

Poděkování

Tímto bych rád poděkoval svému vedoucímu bakalářské práce Ing. Ladislavu Pešičkovi za cenné rady a nápady, které mi během tvorby bakalářské práce poskytl.

Obsah

1 Úvod

V současné době mobilní telefony neslouží pouze k telefonování a psaní SMS zpráv, ale nabízí velmi rozsáhlý počet možností jejich využití, ať už je to například pořizování fotografií, přístup k internetu nebo hraní her.

Tato bakalářská práce se zabývá návrhem a implementací aplikace mobilní hry pro platformu Android s využitím GPS navigace v zařízení telefonu.

V práci bude proveden rozbor vybraných mobilních her a poznávacích aplikací na trhu, které využívají GPS navigaci telefonu. Z výsledků provedeného rozboru a analýzy zadání bude vytvořen návrh řešení výsledné aplikace, který bude obsahovat scénář hry, popis vlastností aplikace a návrh komunikace. Poté budou čtenáři popsány technologie, které bude aplikace využívat.

Další část této práce bude popisovat implementaci aplikace, v tomto případě aplikace klienta a serveru, kde budou popsány jejich projekty, použité architektury, významné balíky tříd spolu s jejich třídami, způsoby ukládání dat a komunikace mezi nimi. Dále zde bude vysvětlen komunikační protokol využívaný v komunikaci mezi serverem a klientem spolu s příklady komunikací a přehledem zpráv protokolu. A na konci této části budou uvedené použité databáze aplikací.

Poslední kapitoly práce bude obsahovat popisy a výsledky testovacích scénářů, kterými bude aplikace testována, a popis možných rozšíření, kterými by mohla být aplikace v budoucnu rozšířena.

2 Rozbor aplikací využívající GPS

V této kapitole se zaměříme na rozbor mobilních her a poznávacích aplikací pro platformu Android využívající globální polohový systém, zkráceně GPS [13], skenování Wi-Fi nebo mobilních sítí a další technologie k určení polohy uživatele. Mobilní hry a poznávací aplikace, které budeme v této kapitole analyzovat jsou: Pokémon GO, Resources, Geocaching, Field Trip, Geofun, Landlord a Galileo. Výsledek rozboru s výčtem vlastností aplikací naleznete v tabulce 2.1.

2.1 Pokémon GO

Pokémon GO je mobilní hra založená na principu rozšířené reality [2], kde aplikace spojuje herní svět se světem reálným pomocí GPS, kamery a dalších senzorů mobilního zařízení.

Aplikace byla vyvinuta společností Niantic. Tato společnost se také proslavila svou dřívější mobilní hrou Ingress, ze které Pokémon GO značně vychází. Hra je založena na populárním seriálu Pokémon.

Jejím hlavním cílem je chodit po reálném světě a v něm nacházet a sbírat pokémony. Příšerky ze světa seriálu zde hráč dále trénuje a s jejich pomocí poráží své protivníky ve vzájemných soubojích.

Aplikace monitoruje pohyb hráče v mapě pomocí polohy určené z GPS, skenování Wi-Fi a mobilních sítí, který zobrazuje na herní mapě, mapě reálného světa vykreslené vektorově ve 3D pohledu. Na mapě jsou také umístěny ukazatele herních bodů, a to pokéstopů, arén nebo samotných pokémonů, kteří se zobrazují náhodně v okolí hráče viz obrázek 2.1(a).

S těmito body může hráč interagovat pouze pokud je v jejich blízkosti. Pokéstopy mají nejen význam ve hře, ale také mohou hráče upozornit na zajímavé místo nebo památku v jeho okolí. Bod totiž obsahuje fotografii a pojmenování daného místa.

Hra je také ovlivněna faktory jako je čas a počasí. Pokud hráč je aktivní po 19. hodině, hra se automaticky přepne do režimu noci a začnou se v okolí objevovat noční pokémoni. Dále po aktualizaci vydané v prosinci 2017, hra také reaguje na reálné počasí, kdy při dešti ve hře také prší a objevují se tak více vodní pokémoni viz obrázek 2.1(b).

Lov pokémonů je ve hře možný ve dvou režimech. V režimu normálním, kde je chytání pokémona zobrazeno na animované ploše nebo v režimu AR. V tomto režimu se pomocí kamery mobilního zařízení snímá obraz před hráčem a do něj je vykreslen pokémon, tím vniká dojem, že pokémon je opravdu před ním.

Herní klient také hráči zobrazuje jeho historii průběhu hry, různé statistiky, jako je například datum startu hraní hry nebo nachozené kilometry při hraní.

Aplikace kvůli své náročnosti rychle vybíjí baterii zařízení. Proto je zde možnost, pouze u telefonů obsahující gyroskop, zapnutí spořiče baterie. Ten funguje tak, že když hráč otočí zařízení k zemi, obrazovka zčerná, a tím sníží spotřebu baterie. Pokud se u hráče objeví pokémon, aplikace upozorní hráče vibrací.

Mimo jiné aplikace také využívá GPS k detekci rychlosti pohybu. Pokud se hráč pohybuje se zařízením příliš rychle, aplikace jej upozorní, aby nehrál hru při jízdě autem. Také díky sledování rychlosti hráče jsou zde nastaveny limity pro sbírání sladkostí pro pokémony nebo pro líhnutí pokémona z vajec. Obě tyto činnosti přestávají být aktivní, pokud se hráč pohybuje příliš rychle. Takto hra zamezuje podvádění.

(a) Herní prostředí (b) Počasí ve hře

Obrázek 2.1: Hra Pokémon GO

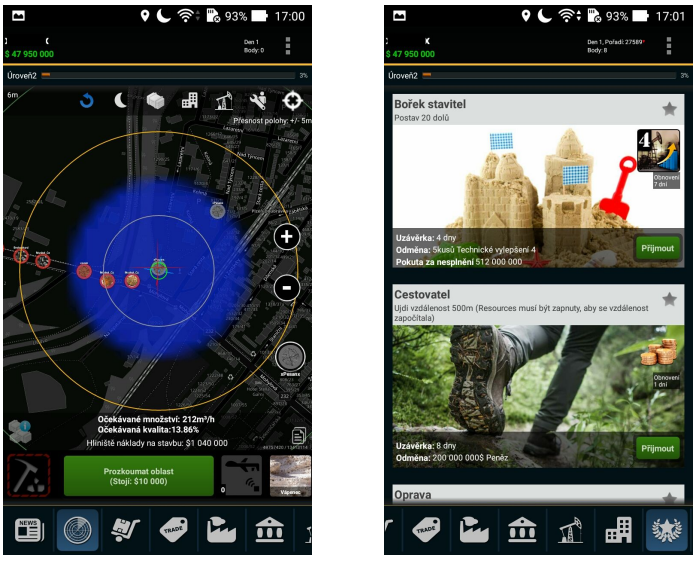
2.2 Resources

Resources je ekonomická simulace mezi více hráči, založena na GPS pozici udanou přístrojem. Hra naskenuje okolí hráče ve skutečném světě, a do něj přidá ložiska surovin, jejich doly a ostatní prvky hry. Hráč ve světě hledá nejbohatší ložiska surovin a staví na nich své doly, aby je mohl vytěžit. S vytěženými surovinami pak obchoduje nebo z nich vytváří výrobky v továrnách. Takto získává peníze a snaží se s nimi co nejlépe hospodařit.

Hra sleduje a zobrazuje hráčovo polohu na mapě. Mapa je zobrazena v pohledu 2D a hráč má možnost si zde nastavit jaké mapy má aplikace využívat nebo si může stáhnout off-line verzi mapy. Klient také podporuje změnu spektra barev pro barvoslepé. Do herní mapy jsou přidány herní body zobrazující suroviny, doly, poklady a továrny viz obrázek 2.2(a).

Hráč si ve hře vybírá z nabídky úkolů viz obrázek 2.2(b), které po jejich započetí musí splnit do určitého časového limitu. Nesplní-li hráč úkol, musí zaplatit herní měnou.

Hra nabízí režim dovolené, který umožní hráči pozastavit hru maximálně na 30 dní. Mimo to obsahuje globální chat a výsledkovou listinu všech hrajících hráčů. Dále také zde lze zobrazit historii a statistiky hráče. Herní klient plně podporuje český jazyk.



(a) Herní prostředí (b) Nabídka úkolů

Obrázek 2.2: Hra Resources

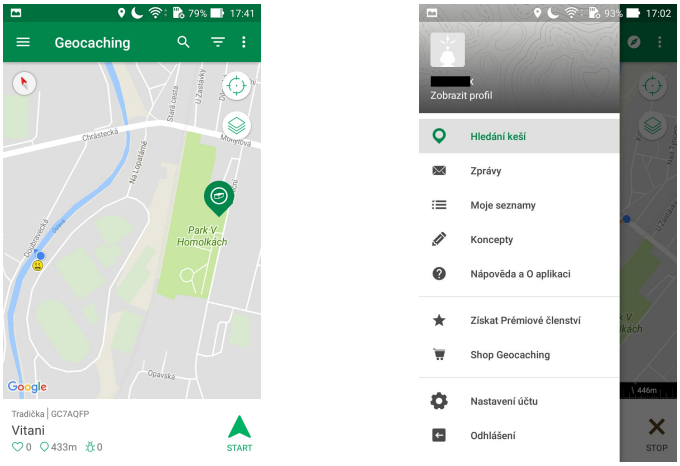
2.3 Geocaching

Aplikace je oficiální mobilní klient pro geocaching [17]. Geocaching je hra na pomezí sportu a turistiky, která spočívá v použití navigačního systému GPS při hledání skryté schránky ve světě, nazývané keše. Pokud se hráči keš podaří najít, zapíše se do seznamu hráčů, kteří ji našli, umístěného ve schránce, a případně vymění její obsah za nový (svůj). Poté schránku opět schová a zamaskuje na dané místo.

Aplikace monitoruje pohyb hráče po mapě pomocí polohy z GPS, skenování Wi-Fi a mobilních sítí, který zobrazuje na mapě. Ta je zobrazena ve 2D pohledu. Uživatel má možnost si vybrat mezi zobrazením ve vektorovém formátu, terénním formátu (uliční mapa s topografickými značkami) nebo ve formátu satelitním. Mapa,viz obrázek 2.3(a), je obohacena o značky, herní body, které znázorňují keše a jejich typ (tradiční keš, event, multi-keš atd.) nebo stav (splněno).

Vývoj hry si určuje hráč tím, že si z nabídky vybere keš, kterou se chystá najít. Jestliže hráč tak učiní, zobrazí se na mapě čára spojující místo, kde se zrovna nachází a kde je umístěna keš. Pokud se hráč k ní dostatečně přiblíží, aplikace jej upozorní. Klient mimo jiné zobrazuje hráči informaci o jeho vzdálenosti mezi ním a keší.

Aplikace umožňuje hráči psát zprávy ostatním hráčům ve hře. Dále umožňuje hráči vytvářet si své seznamy keší, například pro plánované hledání v určité lokalitě. Mimo to je orientace v aplikaci usnadněna podporou českého jazyka – viz obrázek 2.3(b).



(a) Herní prostředí (b) Navigační menu

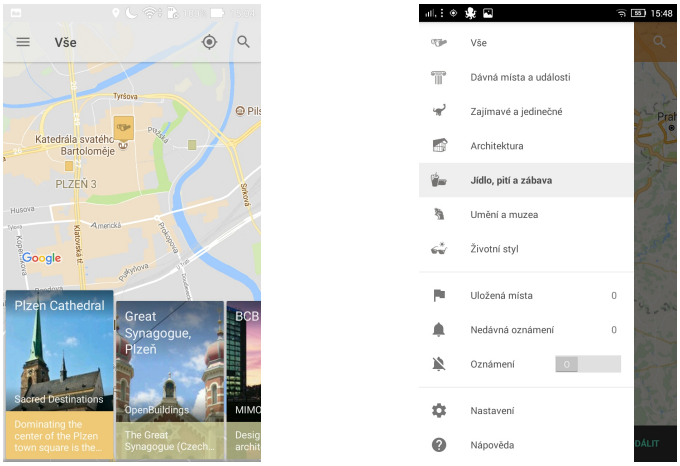
Obrázek 2.3: Aplikace Geocaching

2.4 Field Trip

Aplikace Field Trip funguje jako mobilní průvodce turistům, který poskytuje informace o zajímavých lokalitách, památkách a místům k občerstvení.

Aplikace zobrazuje polohu uživatele na mapě. Mapa je zobrazena v pohledu 2D a je obohacena o ukazatele zajímavých míst.

Každé zajímavé místo má také svoji položku v liště interaktivních karet viz obrázek 2.4(a). Tam je zobrazena fotografie daného místa a jeho popis. Je zde i možnost navigace k němu pomocí aplikací třetích stran např. *Google Maps*. Aplikace mimo jiné umožňuje filtraci zobrazených míst a podporuje český jazyk – viz obrázek 2.4(b).



(a) Mapa s lištou interaktivních karet

(b) Navigační menu

Obrázek 2.4: Aplikace Field Trip

2.5 Další podobné aplikace

Výběr podobných aplikací byl poměrně rozsáhlý, proto jsem se rozhodl zde uvést několik dalších příkladů aplikací.

Geofun

Geofun vychází z geocachingu, který je popsán v kapitole 2.3, ale funguje zcela virtuálně. Jednotlivé mise hry se stahují, a poté fungují zcela off-line. Za každou splněnou misi ve hře se získávají body a podle nich se pak určuje úroveň hráče. Velikost dat těchto misí může být až několik megabytů, neboť

v sobě mohou obsahovat videa, obrázky a zvuky, proto je lepší je stahovat pře Wi-Fi.

Všichni hráči, kteří se do mise zapojí, mohou hrát o hodnotné ceny, víkendové pobyty, poukázky do restaurací apod.

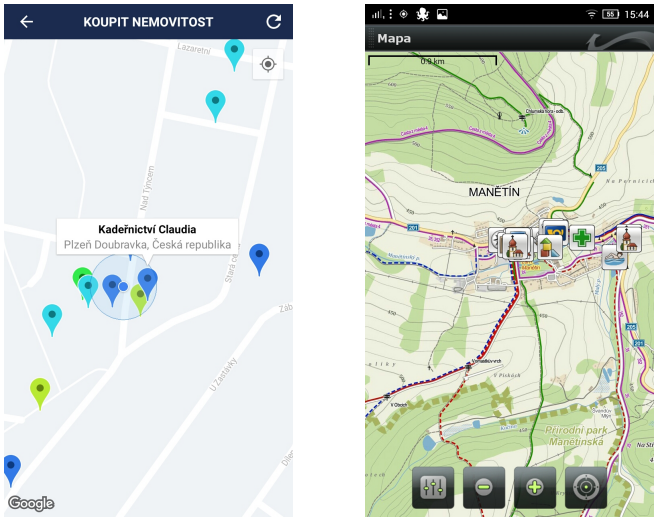
Zajímavost u této hry, mimo off-line režimu, je také český dabing ve hře.

Landlord

Landlord je jako společenská hra, ale pro reálný svět. Aplikace naskenuje hráčovo okolí a reálné nemovitosti, které se v něm nacházejí viz obrázek 2.5(a). Tyto nemovitosti poté dovoluje virtuálně koupit a hospodařit s nimi. Informace o nemovitostech aplikace získává z aplikace Foursquare.

Galileo

Mobilní turistický průvodce poskytující uživateli informace o zajímavých lokalitách, o kulturním a přírodním bohatství nebo i aktuality z dění v regionu viz obrázek 2.5(b). Aplikace podporuje český jazyk, automatické přečtení informací uživateli nebo off-line režim.



(a) Hra Landlord (b) Aplikace Galileo

Obrázek 2.5: Ukázky dalších aplikací

2.6 Shrnutí funkcí

Výsledkem analýzy her a aplikací využívající GPS je výčet funkcí, který naleznete v tabulce 2.1.

Funkce	Popis
Pozice hráče	Sledování aktuální pozice pomocí GPS, skenování Wi-Fi a mobilních sítí
Mapa	Zobrazení reálné mapy v 2D pohledu
Ovládání mapy	Přibližování, oddalování a otáčení mapy
Ukazatele	Zobrazení herních bodů na mapě
Geofence	Hlídání povolené vzdálenosti k bodům či zobrazení pouze určitých herních bodů v dané oblasti
Statistiky	Zobrazení zajímavých statistik ve hře např. nachozené kilometry nebo strávený čas ve hře.
Historie	Zobrazení posledních aktivit ve hře
Systém úspěchů	Možnost plnění speciálních úkolů, které můžou, ale nemusí, být spjaté se scénářem hry, např. Ujdi 10 km
Systém úrovní hráče	Sběr zkušenostních bodů, a tím možnost zvyšování úrovně hráče ve hře
Mise a úkoly	Mise ve hře, díky jejich plnění hráč získává odměny
Systémové zprávy	Důležité informace ve hře pro hráče
Globální chat	Globální komunikace mezi hráči
Výsledky	Výsledková listina hráčů
Jazyk aplikace	Podpora více jazyků
Tréninkový program	Jednoduchý průvodce hrou pro nové hráče
Noční režim	Možnost ztmavení obrazovky pro lepší hratelnost v noci

Tabulka 2.1: Výčet funkcí aplikací

2.7 Možnosti využití GPS v aplikacích

Všechny z výše uvedených aplikací a her využívají GPS k určení hráčovi geografické polohy. Tuto informaci, kterou mohou zjišťovat téměř v reálném čase a přináší jim to bohaté možnosti, jak jí využít.

U poznávacích aplikací tuto polohu využijí tak, že naskenují okolí uživatele a zobrazí mu blízká, pro něj nějakým způsobem zajímavá, místa, která by se mohl rozhodnout navštívit. Pokud by se uživatel rozhodl nějaké takové místo navštívit, aplikace by mohla využít informaci o jeho poloze a nabídnout mu možnost navigace k danému místu. A to navigaci zobrazující přímo trasu, po které se má vydat, nebo například pokud by byla možnost, zobrazit spoje linek městské hromadné dopravy k onomu místu.

Zaměříme-li se na hry, zde je mnohem širší oblast využití informaci o poloze hráče. Hra může být navržena například tak, že bude zobrazovat herní body, lokace či místa, která jsou pro hráče nějakým způsobem zajímavá, v závislosti na tom, kde se rovna nachází. Příklad ze hry Pokémon GO, uvedené výše. Pokud se hráč pohybuje u vodní plochy, zobrazují se mu častěji pokémoni, kteří mají rádi vodu a naopak pokémoni, kteří vodu nemají rádi, se nezobrazí.

Další využití, znovu ze hry Pokémon GO, je kontrola rychlosti pohybu hráče. Hra může reagovat na rychlost tak, že některé aktivity ve hře může hráč plnit pouze při nějaké rychlosti pohybu, nebo jej může upozornit na nebezpečí spjaté s jeho danou rychlostí.

Hra také může využít hráčovu polohu k určení vzdálenosti mezi hráčem a herními body. Takto se může kontrolovat, zda hráč má vůbec povoleno s daným herním bodem nějakým způsobem pracovat.

Poslední zajímavé využití, které bych zde chtěl uvést, je opět ze hry Pokémon GO. Hra přizpůsobuje herní mapu a také i herní režim podle počasí a času v dané oblasti. Takže pokud v dané oblasti prší nebo je noc, je tomu tak i ve hře a hra je ovlivněna tak, že se objevují jiné druhy pokémonů.

3 Návrh aplikace

Na základě průzkumu her a aplikací využívající GPS, který můžete najít v předchozí kapitole, se v této kapitole zaměříme na návrh aplikace.

V první řadě bude nutné nadefinovat jaký bude celkový scénář a cíle hry. Poté jaké funkce a prvky bude aplikace obsahovat. A nakonec jak bude aplikace komunikovat s okolním světem.

3.1 Scénář hry

Hlavním cílem hry bude nasbírat co nejvíce bodů z misí. Hra jako taková konec teoreticky nemá, neboť správce serveru může neustále přidávat nové mise. Hráč tedy může jenom udržovat své skóre, tak aby byl první ve výsledkové tabulce.

Poté, co hráč zaktivoval misi, začne plnit zadané herní body. Herní body mise splní tak, že na dané místo fyzicky dojde dostatečně blízko. Jakmile se tak stane, aplikace pošle serveru zprávu pro validaci. Projde-li validace, bod se označí jako splněný a hráč pokračuje k dalšímu. Pokud daný bod byl poslední nesplněný, mise se tímto splní a hráč dostane odměnu.

Během plnění mise se hráč bude moci kdykoliv podívat, zda někdo už danou misi splnil nebo jí právě plní, a jak je dotyčný na tom.

3.1.1 Typy misí

Ve hře bude mít uživatel možnost si vybrat mise typu *multipoint* a *strict-multipoint*, které jsou popsány níže. Aplikace bude navržena tak, že bude možné přidávat další možné typy misí.

Multipoint mise

Tento typ mise má definované herní body, které hráč musí splnit. Pořadí bodů si hráč určuje sám. Mise může, ale nemusí, mít definovaný startovní bod, cílový bod a nebo obojí. Pokud definovány jsou, hráč je musí dodržet.

Strict-Multipoint mise

Tato mise má definované herní body, které hráč musí splnit. Oproti *multipoint* misi má však striktně definované jejich pořadí, které hráč musí dodržet.

3.2 Vlastnosti aplikace

Dle provedeného průzkumu aplikací a her jsem se rozhodl pro použití následujících prvků v aplikaci.

Aplikace by měla sledovat aktuální lokaci hráče a vykreslovat ji na herní mapě, která by měla být z pohledu 2D. Herní mapu by hráč měl mít možnost ovládat, tzn. přibližovat, oddalovat a otáčet, pomocí tradičních gest, které se standardně využívají na dotykových zařízeních.

Hra bude obsahovat různé mise (úkoly), které v sobě budou mít definovány herní body. Splněním těchto misí hráč obdrží odměny ve formě zkušeností, které sbírá pro zvyšování své úrovně, a ve formě bodů, které určují hráčovo ohodnocení v globální výsledkové listině.

Na mapě by dále měly být vykresleny herní body pomocí interaktivních ukazatelů, které budou sloužit k navigaci hráče. Tyto body zároveň budou sloužit k zobrazení podrobných informací o daném bodu např. název, pořadí a popis. V případě že hráč nějaký bod splní, ukazatel změní svoji barvu, zařízení zavibruje a zobrazí se hláška, že byl bod splněn.

K nastavení určitých herních pravidel by měla aplikace být schopna počítat vzdálenost mezi hráče a herním bodem. Pro splnění jednotlivých bodů mise se bude muset hráč k bodům přiblížit alespoň na 25 metrů. Jiným způsoben nebude možnost body splnit.

Zvyšováním úrovně hráče se mu odemykají mise, které jsou určeny pro hráče s úrovní vyšší. Mise mohou mít různé ohodnocení v závislosti na jejich obtížnosti.

V aplikaci bude možnost zobrazit uživateli různé informace např. datum registrace, počet nachozených kilometrů, stav aktuálně plněné mise, výsledkovou listinu hráčů nebo systémové zprávy. Systémové zprávy bude moci odesílat pouze administrátor na serveru, a to všem přihlášeným uživatelům např. upozornění o budoucí aktualizaci misí, plánované odstávce serveru apod.

3.3 Návrh komunikace

Aby hru mohlo hrát více hráčů najednou na svých zařízení nezávisle na sobě, rozhodl jsem se aplikaci napsat v síťové architektuře klient-server.

Noví hráči se pomocí klientské aplikace zaregistrují na serveru. Poté se budou moci na server přihlásit a začít hrát.

Hráč si v aplikaci vybere ze seznamu misí misi, kterou se rozhodl hrát, a aktivuje jí. Ke stažení seznamu dostupných misi, k jejich aktivaci i k plnění daných misí je potřeba mít aktivní spojení se serverem.

Plnění misí, validace dosaženého herního bodu, probíhá na serveru. Když tedy klient zaznamená, že hráč je v blízkosti herního bodu mise, odešle zprávu serveru. Ten zprávu a její obsah validuje a odešle výsledek klientovi zpět.

Komunikace mezi klientem a serverem bude využívat textový protokol. Zpráva protokolu bude vždy na začátku obsahovat hlavičku zprávy, která identifikuje její typ, a za ní parametry dané zprávy. Mezi jednotlivými prvky zprávy bude použit znak středníku jako oddělovač.

4 Popis použitých technologií

V následující kapitole jsou popsány technologie, které byly použity při vývoji aplikace.

Aplikace klienta byla vyvíjena pro platformu *Android* v jazyce *Java* ve verzi 8 a aplikace serveru byla rovněž vyvíjena v jazyce *Java* ve verzi 8 s využitím technologie *JavaFX*.

Jako vývojové prostředí pro vývoj klienta bylo použito *Android Studio* verze 3.0.1, které je v dnešní době používáno jako standard. Pro vývoj serveru bylo použito IDE *IntelliJ IDEA* od firmy JetBrains verze 2018.1.2, kde byla použita studentská licence zaregistrovaná na univerzitní e-mail.

4.1 Android

Android je v dnešní době nejrozšířenější operační systém pro mobilní telefony, dle serveru StatCounter v květnu letošního roku je Android OS používán na 76,53 % mobilních zařízení [15].

Android je open-source operační systém běžící linuxovém jádře, který byl navržen pro mobilní telefony a později i pro tablety [1]. V dnešní době je však rozšířen i mezi dalšími typy zařízení jako jsou chytré hodiny a chytré televize.

První verze pro komerční užití byla vydána společností Google v září roku 2008. V dnešní době je Android již ve verzi 8.1 jehož označení je *Android O* neboli *Android Oreo*, avšak kvůli jeho rychlému vývoji tato verze není příliš rozšířená mezi mobilními zařízeními, v květnu tohoto roku je to pouze 5.7 % viz obrázek 4.1.

4.2 Výběr úrovně API platformy Android

Před začátkem vývoje aplikací pro Android je velmi důležité vhodně zvolit tzv. *minimální úroveň API* – Min version, tak aby aplikaci mohlo využívat co nejširší spektrum uživatelů.

Minimální úroveň API definuje minimální hranici úrovně API Androidu, kterou musí zařízení uživatele mít nebo mít vyšší, tak aby aplikace na něm šla spustit. Má-li zařízení uživatele úroveň API vyšší, aplikace bude také podporována.

Zvolí-li se minimální úroveň příliš nízko, aplikace přichází o prvky novějších verzí. Naopak pokud se úroveň zvolí příliš vysoko, snižuje se spektrum uživatelů, kterým aplikace poběží.

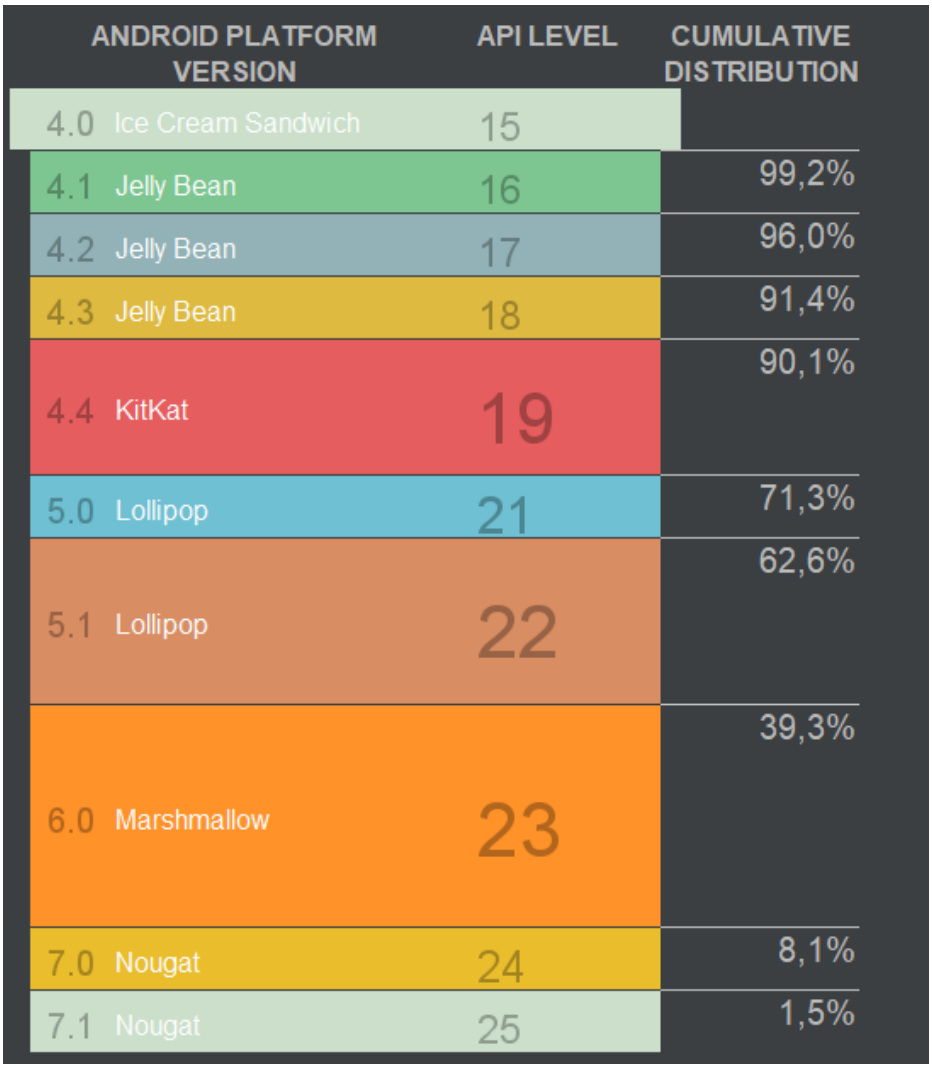
U aplikace se navíc i definuje tzv. *cílová úroveň API* – Target version, která predikuje, jaká zařízení jsou očekávána jako nejčastěji používaná.

Pro volbu vhodné verze API pro aplikaci nám napomáhá společnost Google, která na svých stránkách poskytuje a pravidelně aktualizuje statistiky zastoupení jednotlivých verzí Androidu na trhu viz obrázek 4.1.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.5%
4.2.x		17	2.2%
4.3		18	0.6%
4.4	KitKat	19	10.3%
5.0	Lollipop	21	4.8%
5.1		22	17.6%
6.0	Marshmallow	23	25.5%
7.0	Nougat	24	22.9%
7.1		25	8.2%
8.0	Oreo	26	4.9%
8.1		27	0.8%

Obrázek 4.1: Distribuce verzi Android OS pro květen 2018 [3]

Mimo této statistiky nám také napomáhá nabídka *help me choose*, kterou je k dispozici při zakládání nového projektu v Android Studiu. Tato nabídka nám zobrazí celkové procentuální pokrytí distribuce při výběru konkrétní verze Androidu viz obrázek 4.2.



Obrázek 4.2: Nabídka *help me choose* v Android Studiu

4.3 Aktivita (Activity)

Aktivita je jeden ze základních stavebních bloků aplikace, která zastupuje její prezentační vrstvu. Jedná se o jednu obrazovku aplikace, skrze kterou probíhá interakce s uživatelem.

Aplikace může obsahovat více aktivit, obrazovek, které jsou mezi sebou vázány. V případě že tomu je, musí být být jedna z nich označena jako hlavní, která se zobrazí jako první po spuštění aplikace. Pokud má jedna aktivita nahradit jinou, předchozí aktivita se pozastaví a uloží se do zásobníku aktivit, pokud není definováno jinak, a je nahrazena novou. Jestliže se uživatel chce vrátit k předchozí aktivitě, stiskne tlačítko zpět. Tím se ze zásobníku vybere a spustí poslední vložena aktivita a současná aktivita se zruší. Je-li zásobník prázdný, uživateli se zobrazí domovská stránka zařízení.

Obsah aktivity je definován ve XML souborech s rozvržením (*layout*), kde jsou popsány jednotlivé prvky obrazovky *View*. Mezi těmito prvky aktivity může být jeden, takzvaný `FrameLayout`, který umožňuje aktivitě do sebe vložit tzv. *Fragmenty*, které jsou popsány v další podkapitole.

4.4 Fragment

Od Androidu 3.0, který reagoval na vzestup poptávky po tabletech, musel Google řešit problém, jak umožnit vývojářům co nejjednodušeji vytvářet aplikace, jejichž rozhraní se dokáže přizpůsobit jak fyzicky velkým, tak malým displejům. A jako řešení tohoto problému zavedli fragmenty [4].

Fragmenty celkově označují nový přístup ke tvorbě uživatelského rozhraní, kdy mezi *Activity* a *View* vstupuje ještě jedna vrstva, a to *Fragment*. Ten má za úkol „obalit“ nějaký funkční celek, například zobrazení seznamu položek, zobrazení jedné položky či další možné funkcionality. A to jak k tomu potřebné prvky obrazovky, tak metody s nimi související [4].

Každá Aktivita může obsahovat libovolné množství fragmentů, které se chovají víceméně jako *View*. Prvky obrazovky fragmentů se také dají deklarovat v XML souboru a nebo přidat přímo v kódu. Aktivita může volat jejich metody, nastavovat jim posluchače událostí nebo jednotlivé fragmenty mezi sebou zaměňovat.

Díky více fragmentům v jedné aktivitě aplikace nabývá dojmu, že v ní je více obrazovek (aktivit), ačkoliv tomu tak ve skutečnosti není.

Fragmenty, stejně jako aktivity, mohou využívat zásobník, který slouží k návratu do fragmentu předchozího. Na rozdíl od aktivit toto není zajištěno operačním systémem, ale vývojář musí obsluhu zásobníku implementovat sám, a to v nadřazené aktivitě.

Současné aplikace, i tato, se zaměřují spíše na využití většího počtu fragmentů, než aktivit. Jeden z možných důvodů je ten, že aplikace nabývá modernějšího dojmu pro uživatele. Například díky fragmentům jde relativně snadno docílit výměny obrazovek pomocí tažení (swipe).

4.5 Služba (Service)

Služba [8] je součást aplikace nemající uživatelské rozhraní, která může provádět dlouhotrvající nebo blokující operace na pozadí, jelikož tyto operace jsou v Androidu zakázány v hlavním vlákně aplikace v tzv. *UIThread* [9]. Pokud takové operace vývojář použije v *UIThread*, a aplikace kvůli tomu na okamžik přestane reagovat, správce procesů zobrazí ANR dialog, který se uživatele zeptá, jestli se má aplikace ukončit nebo počkat na její odezvu.

Aktivita či jiná součást aplikace může službu spustit a nechat jí provést své operace nebo jí může spustit a připojit se k ní (*bind*). Spuštěná služba může běžet dál na pozadí, i když uživatel se přepnul do úplně jiné aplikace, není-li definované její ukončení. Součást aplikace, která má připojenou službu, může volat její metody a komunikovat s ní.

Služby se v aplikacích využívají k síťovým transakcím, přehrávání hudby nebo souborovým I/O operacím, které provádějí na pozadí aplikace.

Služby můžeme rozdělit na služby na popředí (*foreground*), služby na pozadí (*background*) a služby vázané (*bound*).

Služby na popředí provádí nějaké operace na pozadí, ale na rozdíl od služeb na pozadí, musí služba vyvolat notifikaci, která upozorní uživatele, že běží. Využití mají například u přehrávače hudby nebo sledování polohy uživatele. Služby na pozadí provádí operace bez jakéhokoliv vědomí uživatele.

Služby vázané mohou být jak služby na pozadí, tak na popředí. Jedná se o služby, které k sobě připojili jiné součásti aplikace (aktivity, fragmenty, ostatní služby aj.). Tyto součásti s ní mohou komunikovat, odesílat požadavky a přijímat výsledky.

4.6 Broadcast Receiver

Broadcast receivery jsou součásti aplikace, které přijímají zprávy od systému, ostatních aplikací nebo od aplikace samotné a vyvolávají reakce na ně. Jejich využití je velmi rozsáhlé. Například mohou vyvolat reakci aplikace na změnu stavu připojení k internetu, rozsvícení displeje nebo díky nim mohou být vyvolány reakce na zprávy od serveru.

Součást aplikace, která chce reagovat na zprávy musí nejprve zaregistrovat instanci broadcast receiveru pomocí metody `registerReceiver()`, které se předají, jako parametry, instance receiveru a filtr zpráv (`IntentFilter`, na které receiver bude reagovat. Součást, která si zaregistrovala broadcast receiver, je povinna na konci svého životního cyklu jej také odregistrovat.

4.7 Práva aplikace

Účelem oprávnění aplikací [7] je ochrana soukromí uživatele. Aplikace, která chce přistupovat k citlivým údajům uživatele jako jsou například seznam kontaktů a SMS, nebo chce využívat nějaké systémové funkce jako fotoaparát nebo snímání polohy uživatele, si musí vyžádat oprávnění od uživatele. V závislosti na typu požadavku systém může udělit některé oprávnění automaticky, nebo může být daný požadavek předán uživateli. Aplikace potřebná oprávnění registruje ve svém manifestu [6].

Do verze Androidu 6.0, byla žádost o povolení oprávnění zobrazena při instalaci aplikace, a jestliže je uživatel neschválil, aplikace se vůbec nenainstalovala. Od Androidu 6.0 tomu tak není, oprávnění uživatel může povolit či zakázat v nastavení systému a je pouze na vývojáři aplikace jak tuto problematiku řešit. Aplikace si při startu může oprávnění zkontrolovat a v případě, že je nemá povolena, může vyžádat jejich povolení. Nebo je může vyžádat až tehdy, kdy jsou oprávnění nezbytně nutná, například pouze před pořizováním fotografie.

Oprávnění jsou rozdělena dle úrovně ochrany na tři typy a to na: *normální*, *podpisové* a *nebezpečné*.

Do *normálních* spadá velké množství oprávnění. Jsou to taková oprávnění, u kterých existuje velmi malé riziko pro soukromí uživatele. Příkladem je například využívání služby Bluetooth [18], stav sítě, vibrace a další.

Podpisová oprávnění uděluje systém v době instalace aplikace, a to pouze v případě, že aplikace žádající dané oprávnění je podepsaná stejným certifikátem jako aplikace, která oprávnění definuje. Příkladem je oprávnění `CLEAR_APP_CACHE`, které umožňuje aplikaci vymazat mezipaměti všech nainstalovaných aplikací v zařízení.

Nebezpečná oprávnění se týkají oblasti, kde aplikace žádá o přístup k datům nebo zdrojům, která zahrnují i soukromé informace uživatele, nebo které by mohli ovlivňovat již uložená data nebo chod ostatních aplikací. Mezi tyto oprávnění je například právo číst kontakty nebo polohu uživatele.

4.8 Práce s GPS

Pokud aplikace chce získat informace o poloze uživatele, musí se do manifestu přidat jedno z následujících oprávnění `ACCESS_FINE_LOCATION`, které povoluje přístup k přesné poloze z místních zdrojů, jako je GPS, vysílače mobilní sítě a Wi-Fi, nebo `ACCESS_COARSE_LOCATION`, které poskytuje informace o přibližné poloze uživatele.

V dalším kroku pro získání polohy si musíme vytvořit instanci třídy `FusedLocationProviderClient`, která slouží jako vstupní bod pro interakci s *Fused Location Provider API* [10], které je součástí *Google Play Services*. Toto API kombinuje GPS, Wi-Fi a další možné prostředky pro získání informací o poloze uživatele.

Po vytvoření instance k získání polohy je potřeba zavolat její metodu `getLastLocation()` vracející úkol(*Task*) s polohou uživatele, na kterou navěsíme `OnSuccessListener`, ve kterém provedeme operace nad získanou polohou.

Více informací o této problematice naleznete zde [5].

4.9 Práce s mapou

K zobrazení mapy v aplikaci společnost Google poskytuje *Google Maps API*, které je součástí*Google Play Services*. Pro využití služeb tohoto API, musí vývojář jako první získat tzv. *API KEY*, kterým se aplikace identifikuje.

Poté součást aplikace (aktivita nebo fragment), která bude zobrazovat mapu musí implementovat rozhraní `OnMapReadyCallback`, které v sobě definuje metodu `onMapReady(GoogleMap)`, zde probíhá nastavení a přidávání prvků do mapy. Dále se do rozvržení (layout), definované XML souborem, součásti musí přidat prvek `MapView` zobrazující mapu, kterému se musí nastavit získaný *API KEY*.

Jako poslední se v metodě `onCreateView()` dané součásti aplikace, získá instance daného `MapView`, nad kterým se zavolají metody `onCreate()` a `getMapAsync()`. Metoda `getMapAsync(OnMapReadyCallback)` nastaví objekt zpětného volání (*callback*), který bude spuštěn poté, co je instance `GoogleMap` připravena k použití.

Více informací o této problematice naleznete zde [11].

4.10 SQLite databáze

SQLite databáze, na rozdíl od běžných databází jako jsou MySQL/MariaDB nebo PostgreSQL, které běží jako služba, je pouze malá knihovna. Databáze je ukládána do jednoho obyčejného souboru na disk, který je bez jakýkoliv problémů přenositelný mezi operačními systémy.

Databáze je určena pro menší aplikace nebo pro aplikace, kde není kladen důraz na uživatelská oprávnění přístupu, neboť neobsahuje některé funkce, které obsahují ostatní velké databázové systémy. Často se využívá u aplikací pro mobilní zařízení, a to jak pro Android, tak pro iPhone.

SQLite má plnou podporu v Android OS a bude využita i v této práci na klientské aplikaci. Bude rovněž použita v aplikaci serveru, která bude naprogramována v Javě, kde budeme muset použít knihovnu `SQLite JDBC Driver`.

4.11 Sockety a komunikace server-klient

Komunikace mezi serverem a klientem této aplikace bude probíhat pomoci TCP protokolu, který využívá *proud* (*stream*) v komunikaci.

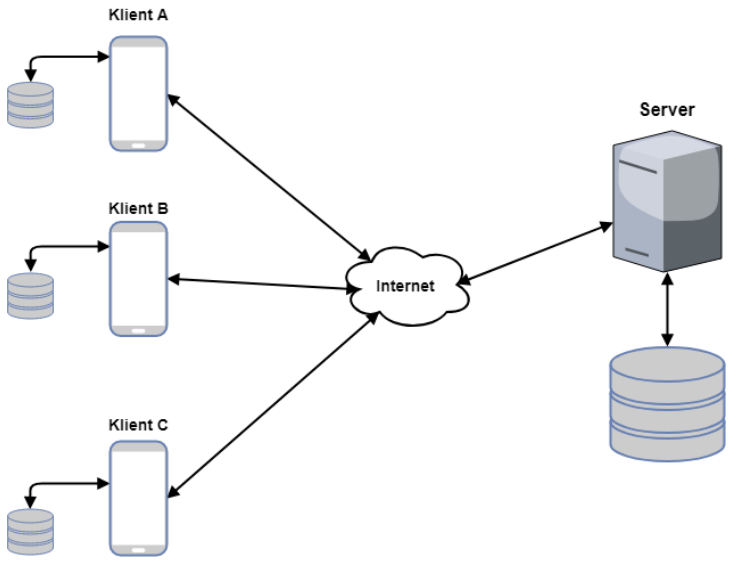
V této komunikaci existují právě dva sockety. Socket, který poslouchá žádosti o připojení (server) a druhý, který vysílá požadavek o připojení (klient). Po navázání spojení této dvojice, mohou být oba použity k odesílání dat skrze proud, a to oběma směry.

Komunikace pomocí TCP protokolu přináší vývojáři určité výhody. Je garantováno, že odeslané zprávy přijdou v pořadí, v jaké byli odeslány. Naopak je zde náročnější režie na vytvoření socketů a navázání jejich spojení.

5 Popis implementace

Výsledná aplikace bude implementována v architektuře server-klient. Komunikace mezi nimi bude probíhat skrze TCP protokol, ve kterém budou posílány textové zprávy s daty. Aplikace serveru a klienta budou obsahovat své vlastní databáze pro ukládání dat. Diagram této architektury naleznete na obrázku 5.1.

Tato kapitola se bude zabývat popisem projektů, jednotlivých balíků tříd, nejvýznamnějších tříd a funkcí u aplikací klienta a serveru. Poté bude popsán komunikační protokol s jeho jednotlivými zprávami a nakonec databáze obou aplikací.



Obrázek 5.1: Diagram architektury aplikace

5.1 Klient

V této kapitole bude popsána struktura projektu aplikace, architektura aplikace, struktura jednotlivých balíků tříd a jejich nejvýznamnější třídy. Dále bude popsána realizace komunikace v rámci aplikace a způsob ukládání dat.

5.1.1 Struktura projektu

Projekt aplikace klienta využívá předem definovanou strukturu generovanou Android Studiem. Tato struktura obsahuje poměrně velké množství slo-

žek a souborů. Popis se bude vázat pouze na složky `java`, `res` a soubor `AndroidManifest.xml`, které jsou umístěny ve složce `app\src\main`. Popis obsahu těchto složek a souboru viz tabulka 5.1.

Složka/Soubor	**Obsah**
`\java`	soubory se zdrojovým kódem, které jsou dále členěny do balíků tříd
`\res`	zdroje aplikace (obrázky, definice řetězců, definice barev aj.)
`\res\drawable`	obrázky
`\res\layout`	definice rozvržení součástí aplikace
`\res\menu`	definice rozvržení menu nabídek
`\res\minimap`	ikony
`\res\raw`	definice nočního režimu mapy
`\res\values`	definice barev, stylů, řetězců apod.
`\AndroidManifest.xml`	název balíčku aplikace a deklaraci součástí, práv, hardwarových či softwarových požadavků [6].

Tabulka 5.1: Struktura projektu aplikace klienta

5.1.2 Architektura

Architektura klientské aplikace dodržuje model MVC, neboli Model-View-Controller. V této architektuře aplikace dělí na tři hlavní celky a to na *model*, *view* a *controller*. *Model* je část aplikace, která má za úkol operace nad daty aplikace (logika aplikace). *View* zobrazuje nebo sbírá data od uživatele a *controller* funguje jako prostředník mezi modelem a view, který mezi nimi předává data.

V této aplikaci reprezentuje vrstvy view a controlleru aktivita, která zobrazuje a sbírá data od uživatele, ale také volá funkce a předává data modelu. Navíc jsou v aplikaci použity ve značné míře fragmenty, popsané v kapitole 4.4, které rovněž volají funkce a předávají data modelu. Model reprezentují třídy v balíku `model`, který je dále popsán v kapitole 5.1.3. Aktivity i fragmenty mají své rozvržení (layout) prvků obrazovky (tlačítka, položky formuláře apod.) definovaný v externích XML souborech umístěných ve složce `\res\layout` .

5.1.3 Balíky tříd

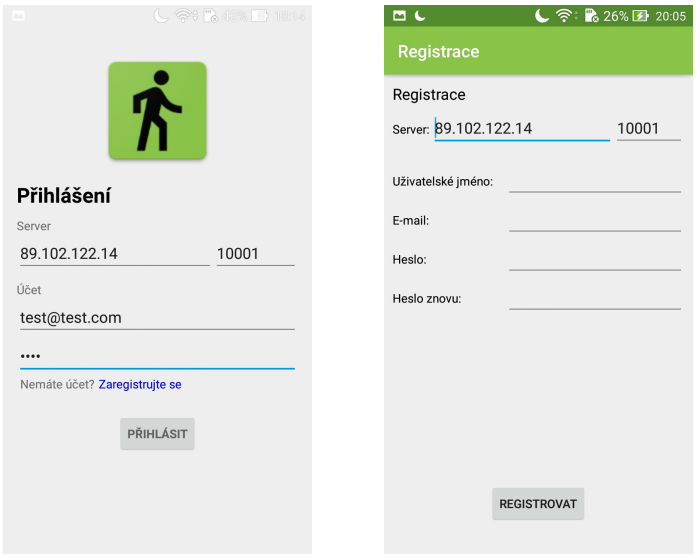
Balíky tříd jsou umístěny ve složce `\java` v balíku `com.zieglerz.kivbp`. Dále popsané balíky vznikly seskupením více tříd, které měly podobný význam, funkcionalitu nebo spadaly do stejné oblasti aplikace.

`activities`

Tento balík tříd obsahuje veškeré aktivity (viz kapitola 4.3) aplikace, a to `LoginActivity`, `RegistrationActivity` a `MainActivity`.

`LoginActivity` obsahuje formulář s tlačítkem, kterými se uživatel může přihlásit do hry – viz obrázek 5.2(a). Pokud jej uživatel vyplní správnými údaji a přihlášení hráče na serveru proběhne v pořádku, aktivita spustí `MainActivity` a ukončí se. Pokud při přihlášení nastane jakákoliv chyba: nesprávné přihlašovací údaje, nedostupný server aj., zobrazí se na obrazovce chybová hláška. Aktivita také nabízí možnost *Zaregistrujte se*, která ukončí aktivitu a nahradí ji za `RegistrationActivity`. Kromě přihlašování do hry má také na starost kontrolu, jestli jsou uživatelem povolena práva pro zjišťování polohy a jestli je povoleno sledování polohy na zařízení. Pokud tomu tak není, zobrazí dialogové okno pro jejich povolení.

`RegistrationActivity` nabízí novému uživateli možnost zaregistrovat se na server. Na obrazovce je zobrazen pouze registrační formulář s tlačítkem pro registraci – viz obrázek 5.2(b). Po vyplnění formuláře a stisknutí



(a) LoginActivity (b) RegistrationActivity

Obrázek 5.2: Obrazovky aktivit aplikace

tlačítka proběhne validace uživatelských údajů, a poté se odesílá registrační požadavek na server. Jestliže údaje byly správné a přijme se od serveru potvrzení o úspěšné registraci, aktivita se ukončí, zobrazí se zpráva o úspěšné registraci a spustí se `LoginActivity`. V případě že validace skončila chybou nebo byla přijata zpráva s chybou, zobrazí se uživateli chybová hláška.

`MainActivity` je hlavní aktivitou aplikace. Má na starosti: zobrazení jednotlivých fragmentů, distribuci instancí vázaných služeb fragmentům, obsluhu tlačítka zpět mezi fragmenty, obsluhu navigačního menu a zobrazení některých notifikací (ztráta spojení se serverem, nová systémová zpráva atd.). Jako výchozí fragment aktivita zobrazuje `GameMapFragment` – popsaný níže v balíku `fragments`. Vybere-li uživatel možnost z navigačního menu, stiskne tlačítko pro zobrazení systémových zpráv a nebo stiskne tlačítko pro zobrazení nastavení, aktivní fragment se uloží do zásobníku a nahradí se jiným. Pokud se hráč bude chtít vrátit zpět do předchozího fragmentu, stiskne tlačítko zpět. Aktivita se poté podívá do zásobníku, jestli v něm je nějaký fragment uložený. V případě že je, vybere ho ze zásobníku, současný fragment ukončí, a vybraný obnoví. Jestliže v zásobníku žádný není, aktivita se pozastaví a přestane být viditelná.

`adapters`

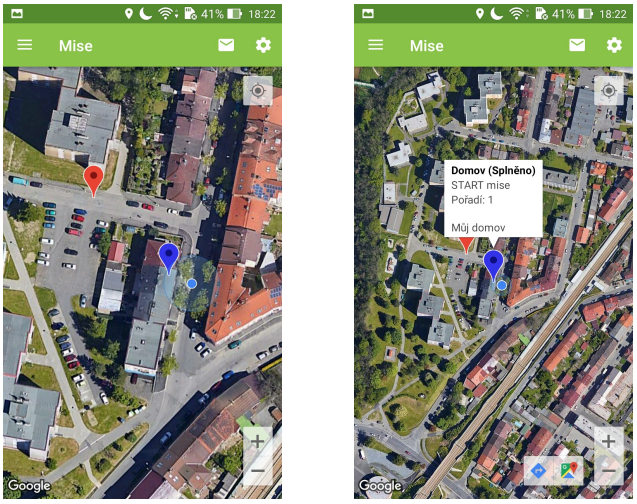
Balík `adapters` obsahuje adaptéry pro seznamy objektů. Adaptéry jsou potomky třídy `BaseAdapter` a využívají se k propojení dat a seznamu. Načtou data do své interní proměnné – např. do pole, poté je zpracují a zobrazí je v rozvržení položky seznamu. Tyto položky následně v sobě zobrazí prvek seznamu (*ListView*) na obrazovce.

`fragments`

V balíku `fragments` jsou uloženy všechny fragmenty aplikace. Podrobnosti o fragmentech naleznete v kapitole 4.4. Nejvýznamnější fragmenty aplikace jsou: `GameMapFragment`, `MissionListFragment`, `MissionDetailFragment` a `ScoreListFragment`.

`GameMapFragment` je výchozí fragment aktivity `MainActivity` zobrazující herní mapu – viz obrázek 5.3(a). Pokud hráč nemá aktivní žádnou misi, na mapě se zobrazuje pouze jeho poloha, ale pokud nějakou aktivní misi má, zobrazují se mu na mapě ještě jednotlivé body mise pomocí interaktivních ukazatelů. Tyto ukazatele mají červenou nebo modrou barvu v závislosti na tom, jestli daný bod hráč splnil – modrá nebo ne – červená. Také pokud hráč na nějaký ukazatel „tapne“, dotkne se ho na obrazovce, zobrazí

se mu podrobnosti o daném bodu – viz obrázek 5.3(b). Po zobrazení fragmentu probíhá načítání jeho aktuální polohy, přičemž se na mapě zobrazuje *ProgressBar*. Po načtení polohy *ProgressBar* zmizí a na se mapě přehraje animace přiblížení. Informace o aktuální poloze hráče, o splněných bodech nebo misích a nebo o deaktivování mise fragment získává pomocí Broadcast Receiverů – viz kapitola 4.6. Přepne-li se hráč do jiného fragmentu nebo aplikaci pozastaví, zobrazí se notifikace jako upozornění, že aplikace, konkrétněji služba zjišťující jeho polohu, stále běží na pozadí. Tato notifikace zmizí pouze, pokud se hráč vrátí do tohoto fragmentu nebo pokud ukončí aplikaci.



(a) Herní mapa

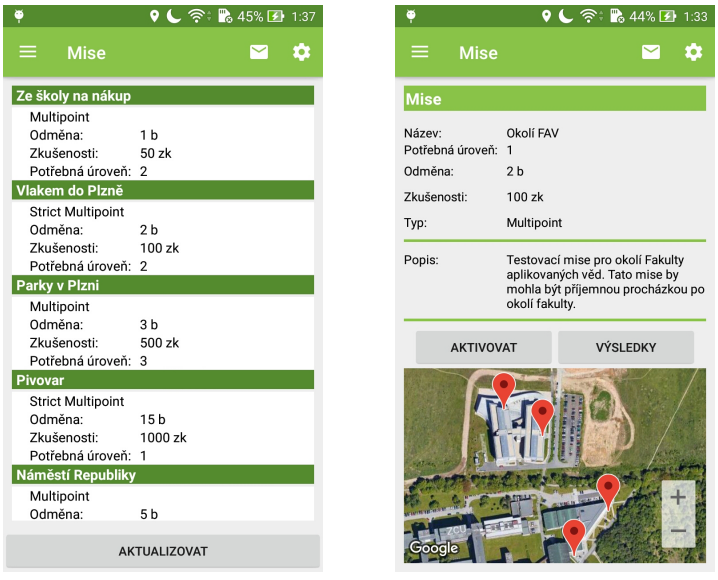
(b) Zobrazení informací o bodu

Obrázek 5.3: Obrazovky fragmentu GameMapFragment

`MissionListFragment` obsahuje seznam misí hry. Jedná se o potomka třídy `ListFragment` – více na [14], který zobrazuje seznam hlaviček misí a tlačítko pro aktualizaci zobrazených dat – viz obrázek 5.4(a). Fragment inicializuje svůj adaptér ve své metodě `onActivityCreated()`, kde se provede jak inicializace adaptéru, tak odeslání požadavku o zaslání hlaviček misí ze serveru. Tento požadavek může také vyvolat hráč stisknutím tlačítka *Aktualizovat*. Přijatá data od serveru přeposílá komunikační služba (`ServerConnectionService`) skrze broadcasty, které poslouchá broacast receiver registrovaný ve fragmentu. Ten data předá adaptéru a aktualizuje seznam. Položka seznamu hlaviček misí obsahuje: název mise, její typ, počet bodů jako odměnu a počet zkušeností za její splnění a potřebnou minimální úroveň hráče. Mimo to se u hlavičky mise zobrazuje, jestli mise byla uživatelem již splněna nebo je její pozadí barevně odlišeno, pokud je mise aktivní.

Data hlavičky mise obsahují datum její poslední editace, která se společně se jménem mise předává fragmentu `MissionDetailFragment`. Tento datum se využívá k aktualizaci dat již stažené mise, která byla, v době po jejím stažení, na serveru upravena administrátorem.

`MissionDetailFragment` slouží k zobrazení detailních informací o misi hráči nebo k její aktivaci či deaktivaci. V metodě `onCreate()` se ověří, jestli fragmentu byla předána data s názvem a datem poslední úpravy (na serveru) mise od nadřazeného fragmentu. Poté proběhne ověření, jestli aplikace má uložena data o dané misi nebo jestli jsou data stále validní (nebyla upravena na serveru). Pokud data neexistují nebo nejsou validní, fragment odešle na server požadavek o zaslání dat o dané misi, které se poté získávají od komunikační služby pomocí broadcast receiveru. Pokud data byla uložena a validní nebo byla právě získána od komunikační služby, provede se jejich načtení a zobrazení na obrazovce fragmentu. V detailu jsou zobrazeny: podrobnosti o misi, tlačítko *Výsledky* odkazující na obrazovku s výsledky této mise, tlačítko *Aktivovat/Deaktivovat* a mapa zobrazující body mise – viz obrázek 5.4(b). Je-li mise již splněna, tlačítko pro aktivaci či deaktivaci mise se hráči skryje. Na mapě se zobrazí body mise ve stejném stylu jako je tomu ve fragmentu `GameMapFragment` – popsáno výše. Pokud je zobrazená mise typu *strict-multipoint*, zobrazí se na mapě navíc šipky mezi body, které znázorňují pořadí bodů.



(a) MissionListFragment (b) MissionDetailFragment

Obrázek 5.4: Obrazovky fragmentů aplikace

`ScoreListFragment` zobrazuje hráči tabulku s výsledky všech registrovaných hráčů na serveru spolu s tlačítky *Aktualizovat* a *Skóre misí*. V metodě `onCreateView()` nebo po stisknutí tlačítka *Aktualizuj* se odešle na server požadavek pro zaslání dat s výsledky všech hráčů. Tato data se po jejich přijetí získávají od komunikační služby pomocí broacast receiverů, kterými se poté vyplní tabulka zobrazená hráči. Stiskne-li hráč tlačítko *Skóre misí*, fragment se vymění za `MissionScoreListFragment`, kde se zobrazí seznam názvů všech misí. Jestliže hráč „tapne“ na misi ze seznamu, zobrazí se `MissionScoreFragment`, ve kterém hráč může vidět výsledky všech hráčů hrajících danou misi.

Výsledky v těchto fragmentech slouží k informování hráče o celkovém průběhu hry. Výsledky ve `ScoreListFragment` obsahují počty nasbíraných bodů (odměn z misí) všech hráčů a výsledky ve `MissionScoreListFragment` zobrazují počty splněných herních bodů mise všech hráčů v dané misi.

`model`

Tento balík obsahuje veškeré třídy obsahující logiku aplikace. Třídy jsou zde dále členěny do: `model.connection`, `model.database`, `model.location` a `model.secure` nebo jsou uloženy přímo v tomto balíku.

`model.connection`

Balík `model.connection` obsahuje třídy mající na starost komunikaci se serverem. Obsahuje službu `SeverConnectionService`, která využívá vlákno `ServerConnectionThread`, broadcast receiver `NetworkChangeReceiver` využívající třídu `Connectivity`, třídu `MessageBuilder` a nakonec výčtové typy: `CommunicationMessage`,`Success` a `Error`.

`SeverConnectionService` je vázaná služba, která pro komunikace se serverem využívá `ServerConnectionThread`. Služba poskytuje ostatním metody pro registraci uživatele, přihlášení a odhlášení uživatele nebo odeslání zprávy na server. Mimo to služba registruje broadcast receiver poslouchající záměry (*Intent*) [14] pro opětovné připojení, který znovu navazuje spojení se serverem poslednímu přihlášenému uživateli.

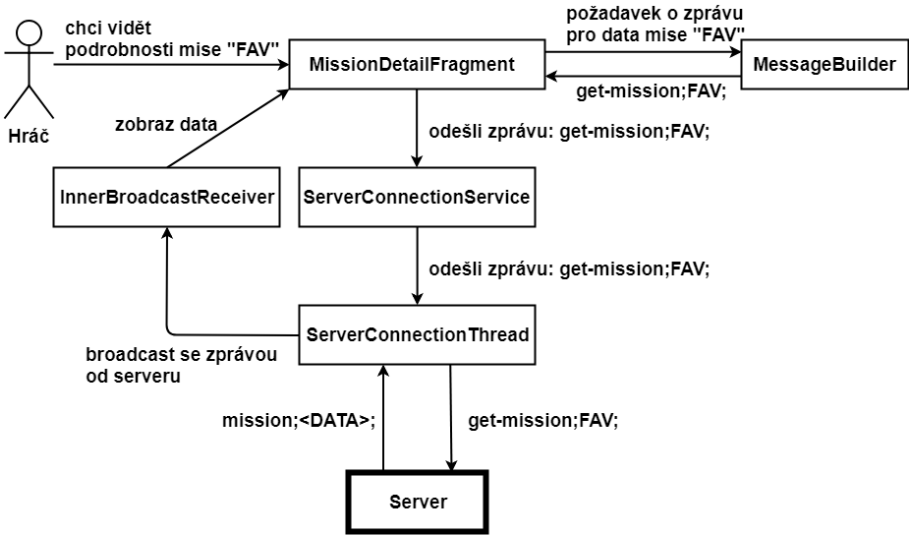
`ServerConnectionThread` je třída vlákna, které komunikuje se serverem. Třída má dva konstruktory, a to konstruktor pro navázání jednorázového připojení k serveru za účelem registrování nového uživatele a konstruktor pro otevření stálého komunikačního kanálu se serverem. Registrace, přihlášení uživatele a celková komunikace mezi klientem a serverem je více popsána v kapitole 5.3. Vlákno poskytuje metody pro odesílání zpráv a odhlášení uživatele a také zpracovává příchozí zprávy od serveru, které přeposílá dál

pomocí broadcastu – viz obrázek 5.5. Více informací o vnitřní komunikace aplikace naleznete v kapitole 5.1.4.

`NetworkChangeReceiver` reaguje na událost změny stavu připojení k síti internetu. Nastane-li změna stavu připojení, receiver zjistí pomocí třídy `Connectivity` o jakou změnu se jedná. Pokud bylo spojení se sítí ztraceno, odešle receiver broadcast o ztrátě spojení. V opačném případě receiver odešle broadcast pro opětovné navázání spojení. Receiver také umožňuje všem ostatním třídám zjistit, jestli je zařízení právě připojeno k internetu či ne.

`MessageBuilder` je třída, která vytváří konkrétní zprávy ve formátu komunikačního protokolu. Výsledné zprávy obsahují hlavičku dle typu zprávy, všechny parametry dané zprávy ve správném pořadí a znak středníku jako oddělovač prvků zprávy.

Výčtový typ `CommunicationMessage` obsahuje hlavičky všech typů zpráv komunikačního protokolu, ke kterým se přistupuje metodou `getMessage()`. Výčtové typy `Error` a `Success` obsahují veškeré chyby a zdary, které mohou být odeslány komunikačním kanálem. Obsahují dvě hodnoty a to jejich kódové označení a ID řetězce odkazující do zdrojů aplikace, ve kterém je obsažena jejich hláška.



Obrázek 5.5: Schéma procesu získání dat o misi od serveru

`model.database`

Balík `model.database` obsahuje třídy, které pracují s databází aplikace. Dělí se dále na dva balíky: `domains` a `managers` a třídy: `DatabaseContract` a `DatabaseHelper`.

V balíku `domains` se nachází třídy, které reprezentují jednotlivé tabulky databáze. Instance těchto tříd mapují jeden záznam tabulky do objektu, kde

hodnoty jednotlivých sloupců tabulky jsou ukládány do atributů tříd. Třídy navíc poskytují getry a setry těchto atributů. V tomto balíku je také uložen výčtový typ `MissionType`, ve kterém jsou uloženy typy misí. Jednotlivé položky typů misí obsahují jak jejich kódové označení, tak řetězec názvu, který je použitý v uživatelském rozhraní.

Balíku `managers` obsahuje manažery, kteří operují nad tabulkami databáze. Manažeři poskytují CRUD operace a případně i dodatečnou logiku aplikace nad daty z databáze. Jedinou výjimkou je třída `UserManager`, která operuje nad daty uživatele, které jsou uloženy v *SharedPreferences* [20].

`DatabaseContract` je třída obsahující konstanty názvů všech tabulek a jejich sloupců v databázi aplikace. `DatabaseHelper` je potomek třídy `SQLiteOpenHelper`, která poskytuje přístup k databázi. Zároveň obsahuje konstanty s SQL příkazy pro vytvoření všech tabulek databáze a metody pro vytvoření celé databáze.

`model.location`

Tento balík obsahuje pouze dvě třídy: službu `LocationUpdateService` a třídu `LocationUtils`, které slouží k práci s polohou zařízení.

`LocationUpdateService` je vázaná služba na popředí – viz kapitola 4.5, která zjišťuje a pracuje s polohou zařízení. Služba k získávání polohy využívá třídu `FusedLocationProviderClient` [10], kterému služba vytvoří a přiřadí požadavek, který provider provádí ve smyčce, společně s úkolem, který se provede po jeho vykonání. Přijde-li od provideru nová informace o pozici zařízení, služba odešle broadcast s touto polohou. Dále také zkontroluje, zda má hráč aktivní misi a pokud ano, zkontroluje aktuální vzdálenost pomocí třídy `DistanceHelper` mezi pozicí hráče a body aktivní mise. V případě že tato vzdálenost je menší než 25 metrů (definováno konstantou `MIN_DISTANCE`), služba zkontroluje připojení se serverem a odešle požadavek pro vyhodnocení bodu. Informaci o tom, jestli má uživatel aktivní misi, získává služba pomocí implementování rozhraní `SharedPreferences.OnSharedPreferenceChangeListener`, neboť se tato informace ukládá do *SharedPreferences*. Služba také registruje broadcast receiver pro zjišťování výsledku validaci bodu nebo pro splnění aktivní mise. Pokud výsledek validace skončí pozitivně, služba vyvolá vibraci zařízení a uloží splnění bodu nebo mise do databáze. Služba navíc poskytuje metody pro zobrazení a skrytí notifikace.

`LocationUtils` je třída, která ukládá a poskytuje informaci o stavu služby `LocationUpdateService`. Tento stav třída ukládá do *SharedPreferences* a říká, jestli služba právě snímá nebo nesnímá polohu zařízení. Služba

totiž může být spuštěna a připojena k součástem aplikace, avšak nemusí snímat polohu. Snímání se musí u služby explicitně spustit pomocí metody `requestLocationUpdates()` nebo případně vypnout pomocí metody `removeLocationRequest()`.

`model.secure`

Balík `model.secure` slouží k zabezpečení uživatelského hesla a obsahuje rozhraní `PasswordHash` a třídu `SHA256PasswordHash`. Rozhraní `PasswordHash` definuje metodu `getHashedPassword()`, která má za úkol převést řetězec s heslem uživatele na řetězec s daným heslem v zašifrované formě. Třída `SHA256PasswordHash` toto rozhraní implementuje a heslo šifruje pomocí algoritmu *SHA-256* [19].

5.1.4 Vnitřní komunikace aplikace

Komunikace v rámci aplikace klienta je realizována více způsoby. Hlavní z nich je komunikace pomocí broadcastů a broadcast receiverů. Mezi další způsoby patří: využití proměnné v *SharedPreferences* a posluchačů událostí změn této proměnné, předávání dat pomocí třídy `Bundle` nebo využití statické metody.

Komunikace pomocí broadcastů a broadcast receiverů je v aplikaci využita hned v několika situacích. Převážně se využívá k vyvolání reakce na odpověď od serveru. Komunikační služba rozpozná zprávu od serveru, založí nový záměr (*Intent*), kterému přiřadí akci, podle které se záměry filtrují, spolu s daty zprávy a odešle ho broadcastem po aplikaci. Součásti aplikace registruji svou instanci abstraktní třídy *InnerBroadcastReceiver*, která tyto broadcasty poslouchá a v případě že se jich daný broadcast týká, zpracují ho. K rozpoznání toho jestli se broadcast týká dané součásti receiver zjistí tím, že se podívá do přibalených dat záměru. Dalším případem použití tohoto typu komunikace je při předávání informace o poloze zařízení od služby `LocationUpdateService` do fragmentu `GameMapFragment`, který tuto informaci zobrazuje na herní mapě. Poslední využití této komunikace je v aktivitě `MainActivity`, která registruje instanci třídy `NetworkChangeReceiver` – více popsána v kapitole 5.1.3 v sekci `model.connection`.

Využití posluchače události změny proměnné v *SharedPreferences* je použit ve fragmentu `GameMapFragment` a ve službě `LocationUpdatesService`. Do proměnné v *SharedPreferences* se ukládá ID aktivní mise hráče nebo číslo -1, pokud hráč aktivní misi nemá. Pokud nastane změna v této proměnné, vyvolá to v obou součástech načtení aktivní mise. V případě že se do proměnné právě uložilo číslo -1, součásti změnu ignorují.

Statické metody jsou v aplikaci využity převážně u tříd `DistanceHelper` a `VibrateHelper`, které nabízí určitou funkcionalitu napříč aplikací svými veřejnými statickými metodami – např. spočtení vzdálenosti mezi dvěma body nebo vyvolání vibrace zařízení. Mimo to jsou statické metody využity pro distribuci instancí služeb do fragmentů aplikace. Aktivita `MainActivity` si připojí služby, uloží je do svých statických proměnných a pomocí nich nabízí tyto instance statickými metodami fragmentům či jiným částem aplikace. Tento postup má výhodu v tom, že jsou služby připojeny pouze na jednom místě.

5.1.5 Ukládání dat

Ukládání dat v rámci aplikace klienta je realizováno pomocí SQLite databáze – viz kapitola 4.10 a tzv. *SharedPreferences* [20]. Databáze se využívá pro ukládání dat hry (mise, body, úrovně) a pro ukládání systémových zpráv od serveru. Do *SharedPreferences* se ukládají data o uživateli, data pro chod aplikace apod.

Připojení do databáze aplikaci poskytuje třída `DatabaseHelper` – popsaná v kapitole 5.1.3. Tu využívají třídy balíku `model.database.managers`, které provádí CRUD operace nad databází. Manažeři od třídy získají připojení k databázi a to buď v režimu *readable* (pouze pro čtení) nebo v režimu *writable*, který dovolí s databází jakýmkoliv způsobem manipulovat. Připojení poskytuje manažerovi metody pro vkládání, aktualizaci a mazání záznamů databáze nebo metody pro spouštění SQL příkazů: `execSQL` pro spuštění příkazu bez návratové hodnoty a `rawQuery()`, který spustí příkaz a zpřístupní jeho výsledky pomocí objektu `Cursor`. Tyto výsledky manažeři uloží do objektů tříd z balíku `model.database.domains`, se kterými se dále pracuje v aplikaci.

SharedPreferences jsou využity v aplikaci využity pro ukládání dat o uživateli, dat pro předvyplnění přihlašovacího formuláře, ID aktivní mise hráče nebo také ukládání stavu, který říká, jestli je snímání polohy ve službě `LocationUpdateService` aktivní. Správu dat o uživateli má na starosti třída `UserManager`. Manažer zpracovává zprávu od serveru s uživatelovými daty a ukládá je do preferencí nebo nabízí metody pro změnu hodnot zkušeností nebo bodů hráče. Data pro předvyplnění přihlašovacího formuláře spravuje aktivita `LoginActivity`, která po úspěšném přihlášení hráče uloží data formuláře do preferencí. Zbylé případy užití *SharedPreferences* jsou popsány v sekci `model.location` kapitoly 5.1.3 nebo v kapitole 5.1.4.

5.2 Server

V této podkapitole bude popsána struktura projektu aplikace, architektura aplikace, jednotlivé balíky třídy spolu s jejich nejvýznamnějšími třídami. Poté bude popsána realizace ukládání dat aplikace a její logování.

5.2.1 Struktura projektu

Projekt aplikace obsahuje strukturu předem definovanou vývojovým prostředím IntelliJ IDEA. Popis projektu se bude týkat pouze obsahu složky `src`, ve které jsou uložený všechny zdrojové soubory aplikace. Popis obsahu této složky viz tabulka 5.2.

Složka	**Obsah**
`\app`	soubory se zdrojovým kódem, které jsou dále členěny do balíků tříd
`\db`	soubor SQLite databáze
`\layout`	FXML soubory s definicemi obrazovek aplikace
`\lib`	knihovnu pro práci s SQLite databází v Javě
`\log`	logy aplikace
`\resources`	ikonu aplikace

Tabulka 5.2: Struktura projektu aplikace serveru

5.2.2 Architektura

Aplikace serveru rovněž jako klientská aplikace využívá architekturu MVC, která je více popsána v kapitole 5.1.2. Model reprezentují manažeři, kteří obsahují operace nad daty a databází, a třídy vláken obsahující komunikační logiku. Kontrolery jsou uloženy v balíku `controllers`. Tyto třídy jsou potomky abstraktní třídy `Controller`, která definuje jejich základní operace. View aplikace jsou uloženy v FXML souborech, které definují obsah obrazovek. Tyto soubory zpracovávají kontrolery pro grafické rozhraní (GUI) pomocí třídy `FXMLLoader`.

Abstraktní třída `Controller` definuje základní operace kontrolerů, díky tomu je možné aplikaci rozšířit o další typy uživatelského rozhraní – např.

webové rozhraní. Aplikace nyní plně podporuje GUI pomocí *JavaFX* a také podporuje její základní funkcionalitu v terminálu operačního systému.

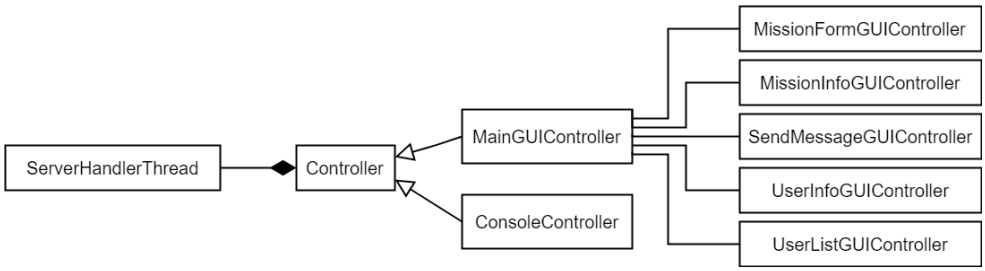
5.2.3 Balíky tříd

Balíky třídy (*packages*) vznikly v projektu k seskupení tříd s podobným významem, funkcionalitou nebo spadaly do stejné oblasti v aplikaci k zpřehlednění výsledného projektu. Balíky a některé třídy mimo ně jsou umístěny ve složce `\src` uvnitř balíku `app`. Dále v této kapitole jsou popsány jednotlivé balíky spolu s jejich nejvýznamnějšími třídami.

`controllers`

Balík `controllers` obsahuje třídy kontrolerů aplikace. Kontrolery slouží k předávání dat mezi modelem (manažery a komunikačními vlákny) a uživatelským rozhraním. Mezi nejvýznamnější třídy zde patří `Controller` a `MainGUIController`, které jsou popsání níže. Ostatní třídy jsou kontrolery pro dílčí obrazovky aplikace s výjimkou třídy `ConsoleController`, která slouží jako hlavní kontroler pro spuštění aplikaci v konzoli operačního systému.

`Controller` je abstraktní třída definující hlavní kontrolery aplikace. Jejími potomky jsou třídy `MainGUIController` a `ConsoleController` – viz obrázek 5.6. Třída definuje metody, které kontrolery musí implementovat nebo je od ní dědí. Metody, které musí potomci třídy implementovat jsou: `registerClient()`, `unregisterClient()` a `clientIpToUser()`. Metody `registerClient()` a `unregisterClient()` mají reagovat na nově přihlášeného nebo na odhlášení uživatele (notifikovat v UI, zapsat do logu atd.). Metoda `clientIpToUser()` má za úkol převést v uživatelském rozhraní IP adresu přihlášeného uživatele na jeho e-mailovou adresu. Příkladem zděděných metod jsou `startServer()` a `endServer`. Metoda `startServer()` vytváří a zapíná hlavní komunikační vlákno serveru `ServerHandlerThread` a metoda `endServer` ho ukončuje.



Obrázek 5.6: Diagram kontrolerů aplikace

`MainGUIController` je kontrolerem hlavní obrazovky GUI aplikace. Kontroler obsluhuje jednotlivé prvky obrazovky (menu, seznamy a další), mimo to také spouští ostatní kontrolery–obrazovky aplikace. Ostatní kontrolery GUI s jejich funkcí naleznete v tabulce 5.3.

Kontroler (obrazovka)	**Funkce**
`MissionFormGUIController`	zobrazuje formulář pro založení nebo úpravu mise
`MissionInfoGUIController`	zobrazuje podrobnosti o misi a dovoluje administrátorovi vyloučit hráče z mise, pokud jí hráč nemá dosud splněnou
`SendMessageGUIController`	zobrazuje formulář pro odeslání systémové zprávy klientům
`UserInfoGUIController`	zobrazuje podrobnosti o hráči a umožňuje administrátorovi odeslat danému hráči systémovou zprávu, odhlásit ho ze serveru nebo ho vyloučit z jeho aktivní mise
`UserListGUIController`	zobrazuje tabulku se všemi registrovanými hráči na serveru spolu s jejich stavem (on-line/off-line), datem posledního přihlášení, úrovní hráče a jejich počtem bodů (počet dosažených odměn z misí)

Tabulka 5.3: Výčet GUI kontrolerů a jejich funkcí

`database`

Tento balík obsahuje třídy pracující s databází aplikace. Uvnitř toho balíku jsou další dva: `domains` a `managers`, společně s třídami: `DatabaseConfig`, `DatabaseContract`, `SQLiteJDBCDriverConnection` a souborem pro nastavení názvu a cesty k souboru databáze `config.properties`.

V balíku `domains` se nachází třídy reprezentující jednotlivé tabulky databáze. Instance těchto tříd mapují jeden záznam tabulky do objektu, kde hodnoty jednotlivých sloupců tabulky jsou ukládány do atributů tříd. Třídy navíc poskytují getry a setry těchto atributů spolu s metodou pro výpis

jejich obsahu do řetězce (`toString()`). Dále je zde uložen výčtový typ `MissionType` spolu s rozhraními `MessageDescripition` a od něj oddděné `MessageDescripitionForSpecUser`. Ve výčtovém typu `MissionType` jsou uloženy typy misí. Jednotlivé položky typů misí obsahují jak jejich kódové označení, tak řetězec názvu, který je použitý v uživatelském rozhraní. Rozhraní `MessageDescrpition` definuje metodu `getMessageDesc()`, která má za úkol navracet řetězec s daty ve formátu komunikačního protokolu. Rozhraní od tohoto oddděné `MessageDescrpitionForSpecUser` definuje rovněž `getMessageDesc()`. S tím rozdílem, že metoda má navíc parametr `id`, který identifikuje položku z databáze, jejíchž obsah se má zapsat do výsledného řetězce.

V balíku `managers` jsou uloženy manažeři, kteří operují nad tabulkami databáze. Manažeři poskytují CRUD operace a případně i dodatečnou logiku aplikace nad daty z databáze. Jedinou výjimkou v tomto balíku je třída `ManagerDistributor`, který ve svém konstruktoru vytvoří instance všech manažerů ve správném pořadí, neboť někteří manažeři jsou mezi sebou vázání – např. `MissionManager a PointManager`. Instance poté distribuuje napříč aplikací pomocí statických getrů. Pokud nastane chyba v konstruktoru v jednom s manažerů, konstruktor distributoru vyhodí výjimku `DatabaseConnectionException` notifikující, že nebylo navázáno spojení s databází.

`SQLiteJDBCDriverConnection` je třída poskytující přípojení k SQLite databázi. Obsahuje metodu `connect()`, která z třídy `DatabaseConfig` získá název a cestu k souboru databáze a pokusí se k němu připojit. Tato metoda obsahuje jeden boolean parametr `dropAll`. Pokud je nastaven na `true`, tak se po připojení k databázi v ní vymažou veškerá data a založí se nové prázdné tabulky. Třída dále obsahuje metody pro zjištění stavu připojení k databázi a pro získání objektu `Connection`, který reprezentuje připojení k databázi.

`DatabaseConfig` má za úkol pouze načíst nastavení databáze ze souboru `config.properies`, ve kterém je definována cesta k souboru databáze a jeho název. Tyto dvě načtené položky nastavení dále poskytuje pomocí getrů.

`DatabaseContract` obsahuje konstanty názvů všech tabulek databáze spolu s názvy jejich sloupců. Dále jsou zde uloženy konstanty s celými příkazy pro vytvoření tabulek.

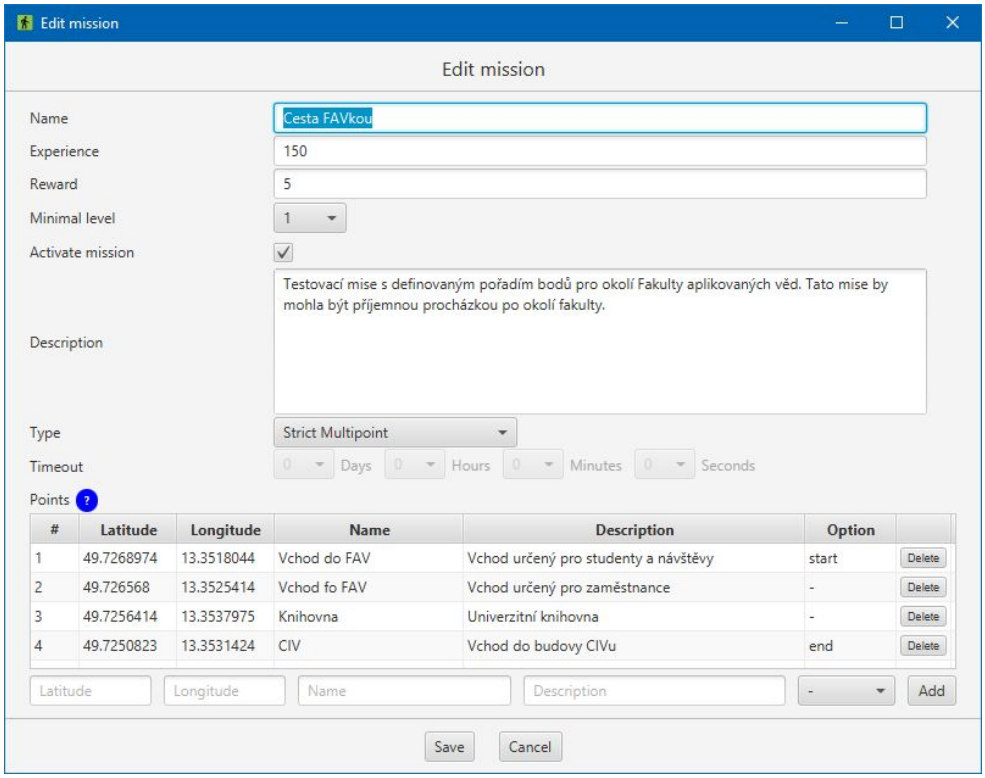
`gui`

Balík `gui` obsahuje třídy spojené s grafickým uživatelským rozhraním, které jsou seskupeny do balíků `properties` a `tablecells`. Mimo ně je zde uložena navíc třída `TextAreaLogHandler` poskytující obsluhu pro zobrazování

logů v prvku `TextArea` na obrazovce. Více o logování v aplikaci naleznete v kapitole 5.2.5.

V balíku `properties` jsou uloženy tzv. *GUIProperty*. Třídy uchovávající data z doménových tříd (třídy z balíku `database.domains`), které využívají prvky grafického rozhraní *JavaFX*. V této aplikaci jsou využity v tabulkách.

Balík `tablecells` obsahuje třídy různých typů buněk tabulky, které neposkytuje JavaFX. Jsou zde buňky, které podporují editaci řetězce, celého čísla a čísla desetinného, dále buňky obsahující tlačítko nebo výběrový seznam. Tyto buňky můžete vidět na obrázku 5.7.



Obrázek 5.7: Formulář pro úpravu mise

`threads`

Tento balík obsahuje všechny třídy vláken, které jsou v aplikaci použity. Nachází se zde jak vlákno obsluhující příkazovou řádku operačního systému `ConsoleHandlerThread`, tak vlákna pro komunikaci: `ServerHandlerThread` a `ClientHandlerThread`.

`ConsoleHandlerThread` poskytuje možnost komunikovat s aplikací skrze příkazovou řádku operačního systému a to i v případě že byla aplikace spuštěna v režimu s grafickým rozhraním. Vlákno má implementovanou pouze

základní funkcionalitu a to přijímání příkazů pro výpis nápovědy (`help`) nebo pro ukončení aplikace (`exit`). Další příkazy by mohly být předmětem rozšíření aplikace.

`ServerHandlerThread` je vlákno aplikace, které má na starosti přijímání nových připojení, odesílání systémových zpráv, a odpojování klientů. Vlákno založí, spustí a poslouchá serverový socket na portu definovaném konstantou `PORT`, která má nastavenou hodnotu 10001. Pokud socket akceptuje nové připojení, vlákno zkontroluje, jestli již neběží klientské vlákno (`ClientHandlerThread`) obsluhující klienta na dané IP adrese. Jestliže takové vlákno existuje, je odpojeno a na místo něj je vytvořeno vlákno nové. Pokud takové vlákno neběží, je rovnou vytvořeno nové a je mu předáno připojení. Poté serverové vlákno informuje kontroler o novém připojení pomocí jeho metody `registerClient()`. Všechna klientská vlákna se zde uchovávají v seznamu (*ArrayList*), díky kterému serverové vlákno může s jednotlivými vlákny operovat: odesílat zprávy klientům, odpojovat klienty nebo získávat informace o klientech.

`ClientHandlerThread` má na starost komunikaci serveru s konkrétním klientem. Vlákno po své inicializaci vyzve klienta k autentizaci, na kterou může klient odpovědět svými autentizačními údaji nebo zprávou o jeho registraci. V případě že se klient registruje, vlákno zkontroluje registrační údaje a odešle klientovi výsledek registrace, poté spojení ukončí. V opačném případě vlákno zkontroluje přihlašovací údaje uživatele a odešle mu výsledek. Pokud autentizace proběhla v pořádku, vlákno spustí instanci třídy `AliveChecker` a poté přejde do nekonečné smyčky, kde obsluhuje příchozí zprávy od klienta. `AliveChecker` je vnitřní třída vlákna, které má na starost kontrolu aktivního spojení mezi klientem a serverem. Tato kontrola funguje tak, že vlákno odesílá speciální zprávu `alive` klientovi každou minutu a očekává od něj jako odpověď zprávu `alive`. Pokud klient neodpoví do stanoveného limitu (jedné minuty) vlákno ukončí dané spojení spolu s celým klientským vláknem. Vlákno obsluhující klienta dále obsahuje metody na zpracování všech typů zpráv – popis zpráv komunikačního protokolu viz kapitola 5.3.3.

Další třídy balíku `app`

V balíku `app` se kromě výše uvedených balíků nachází také několik samostatných tříd. Mezi tyto třídy patří např. `Main`, `MessageBuilder`, `ServerLogger` – viz kapitola 5.2.5, pomocné třídy `DistanceHelper` a `DateHelper`, výčtové typy `CommunicationMessage`, `Error` a `Success` a další.

`Main` je hlavní třída aplikace. Při startu třídy zkontroluje argumenty

z příkazové řádky. Pokud nebyl ve spouštěcím příkazu zadán žádný argument, aplikace je spuštěná ve výchozím režimu a v režimu s grafickým uživatelským rozhraním. Možné parametry jsou: `-help` zobrazující v příkazové řádce nápovědu pro aplikaci a `-nogui` pro spuštění aplikace v režimu příkazové řádky. Poté se spustí inicializace aplikace, kde se nastaví obsluha selhání (třída `FailureHandler` – popsána v kapitole 5.2.5), proběhne připojení k databázi a vytvoří se třída `ManagerDistributor`. Po úspěšné inicializaci aplikace dále pokračuje ve zvoleném režimu.

`MessageBuilder` je třída, která vytváří konkrétní zprávy ve formátu komunikačního protokolu. Výsledné zprávy obsahují hlavičku dle typu zprávy, všechny parametry dané zprávy ve správném pořadí a znak středníku jako oddělovač prvků zprávy.

Pomocné třídy `DistanceHelper` a `DateHelper` poskytují napříč aplikací pomocí statických metod výpočet vzdálenosti mezi dvěma body nebo tzv. „prázdný datum – empty date“. *Prázdný datum* je instance třídy `Date`, ve které je nastaven čas na 00:00:00 dne 1.1.1970, který se v aplikaci využívá místo hodnoty `null`. Navíc třídy `DateHelper` poskytuje konstantní instanci třídy `DateFormat`, která slouží k formátování časových údajů.

Výčtový typ `CommunicationMessage` obsahuje hlavičky všech typů zpráv komunikačního protokolu, ke kterým se přistupuje metodou `getMessage()` a je zcela identický se souborem aplikace klienta. Výčtové typy `Error` a `Success` obsahují veškeré chyby a zdary, které mohou být odeslány komunikačním kanálem. Obsahují dvě hodnoty a to jejich kódové označení a řetězce, ve kterém je obsažena jejich hláška.

5.2.4 Ukládání dat

Veškerá data aplikace, kromě logů – viz kapitola 5.2.5, jsou ukládána do databáze, popis tabulek a model databáze naleznete v kapitole 5.4. Databáze je typu SQLite, viz kapitola 4.10. Tento typ databáze byl zvolen z následujících důvodů: SQLite databázi používá i klientská aplikace, její data jsou uložena v jediném přenositelném souboru a k jejímu využití v aplikaci je potřeba pouze jedna knihovna třetí strany.

Připojení k databázi poskytuje třída `SQLiteJDBCDriverConnection`. To je dále využito v manažerech aplikace, kteří provádí CRUD operace a veškerou logiku na daty. Manažeři ve svých metodách vytváří dotazy do databáze pomocí *prepared statements* [16], které slouží jako ochrana před útoky na databázi. Výsledky dotazů získávají pomocí objektu `ResultSet`, který je alternativou ke třídě `Cursor` na klientské aplikaci a ukládají do objektů tříd z balíku `database.domains`.

5.2.5 Logování

Logování v aplikaci serveru je implementováno ve třídě `ServerLogger`, která ukládá logy do textových souborů do složky `logs`. Tyto soubory obsahují v názvu `kivbp_server_` a datum ve formátu DD-MM-RRRR, tedy každý soubor obsahuje logy jednoho dne. Třída navíc poskytuje metodu pro detailní výpis chyby do logu `printStackTrace()` .

`ServerLogger` k logování využívá třídu Javy `Logger`. Díky této třídě se do loggeru mohou přidávat další obsluhy (*Handler*), nebo spíše proudy, do kterých se logy budou vypisovat. Takto se v aplikaci přidává obsluha výpisu logů do `TextArea`, která je zobrazena v hlavním okně aplikace.

Pro zápis chybových hlášek při selhání aplikace byla implementována třída `FailureHandler`, která se nastavuje v inicializaci aplikace místo výchozí obsluhy neodchycených výjimek, neboli selhání aplikace. Tato třída pouze zapíše chybové hlášky do logů a poté vyvolá výchozí obsluhu selhání.

5.3 Komunikace server-klient

Tato kapitola je zaměřena na popis komunikace mezi aplikacemi serveru a klienta. Nejprve bude popsán komunikační protokol, poté následuje popis případů komunikace mezi aplikacemi a přehled všech zpráv komunikace.

5.3.1 Komunikační protokol

Komunikaci mezi serverem a klientem je skrze textový protokol, kde se jednotlivé zprávy protokolu posílají pomocí segmentů TCP. Více informací o protokolu TCP viz kapitola 4.11.

Protokol, viz struktura zprávy, využívá jako oddělovač hodnot (prvků, parametrů) zprávy znak středníku, kterým se zprávy i ukončují. Prvním prvkem zprávy je vždy její hlavička určující o jakou zprávu se jedná. Poté mohou být přidány jednotlivé hodnoty dané zprávy. Počet hodnot ve zprávě není nijak omezen, neboť je k přenosu využit protokol TCP.

Čtení zprávy začíná přečtením hlavičky a tím i její rozpoznání. Rozpoznaná je dále předána určeným metoda k její zpracování. Ty jako první zprávu rozdělí dle znaku středníku a uloží do pole řetězců. Potě vzniklé pole dále zpracovávají.

Struktura zprávy protokolu:

```
hlavička_zprávy;hodnota_zprávy_1;hodnota_zprávy_2;
```

5.3.2 Příklady komunikace

V této kapitole budou popsány scénáře nejvýznamnějších příkladů komunikace mezi aplikacemi klienta a serveru. Těmito příklady jsou: registrace nového hráče, přihlášení hráče do hry, aktivace mise, validace bodu a zobrazení výsledkové listiny hráčů na klientu.

Registrace nové hráče

Bude-li se nový hráč chtít zaregistrovat na serveru, bude se muset v aplikaci klienta přepnout z přihlašovací obrazovky na obrazovku pro registraci pomocí odkazu *Zaregistrujte se*. V této obrazovce vyplní formulář svými registračními údaji a odešle registrační požadavek na server pomocí tlačítka *Registrovat*. Odeslání požadavku začíná v aktivitě `RegistrationActivity`. Ta po úspěšné validaci údajů zavolá metodu `registerUser()` vázané služby `ServerConnectionService` – viz kapitola 5.1.3: sekce `model.connection`. V této metodě služba vytvoří instanci komunikačního vlákna `ServerConnectionThread` pomocí konstruktoru, který je určený pouze k registraci. Ten inicializuje vlákno a spustí jeho metodu `register`.

Metoda `register` jako první zkontroluje stav připojení k internetu. Pokud kontrola dopadne negativně, odešle broadcast a hráči se zobrazí chybová hláška na obrazovce. V opačném případě se založí nový socket s hodnotami adresy a portu serveru z formuláře a spustí se jeho metoda `connect()`. Jestliže navázání spojení skončí chybou nebo uplyne časový limit, registrace končí a hráči se zobrazí chybová hláška na obrazovce. Proběhne-li připojení k serveru v pořádku, klient čeká na výzvu k autentizace od serveru, na kterou odpovídá zprávou `register` s registračními údaji hráče. Poté klient čeká na výsledek registrace. Přijde-li od serveru zpráva `success` s kódem úspěšné registrace spojení končí a aplikace se přepne zpět na obrazovku pro přihlášení a zobrazí hlášku o úspěšné registraci. V opačném případě rovněž končí spojení se serverem, ale v aplikaci zůstává zobrazena obrazovka registrace společně s chybovou hláškou.

Přihlášení hráče

Přihlášení hráče do hry poskytuje přihlašovací aktivita (`LoginActivity`). Po vyplnění přihlašovacího formuláře údaji hráče a stisknutí tlačítka *Přihlásit* aktivita provede validaci údajů a zkontroluje, jestli hráč povolil na svém zařízení oprávnění pro snímání polohy zařízení v aplikaci a jestli je na zařízení povoleno snímání polohy. Pokud validace skončí chybou, je na obrazovce zobrazena chybová hláška. Pokud nejsou povolena oprávnění nebo není povo-

leno snímání polohy na zařízení, zobrazí se dialogové okno pro jejich povolení. Poté aktivita zavolá metodu `connect()` vázané služby `ServerConnectionService`.

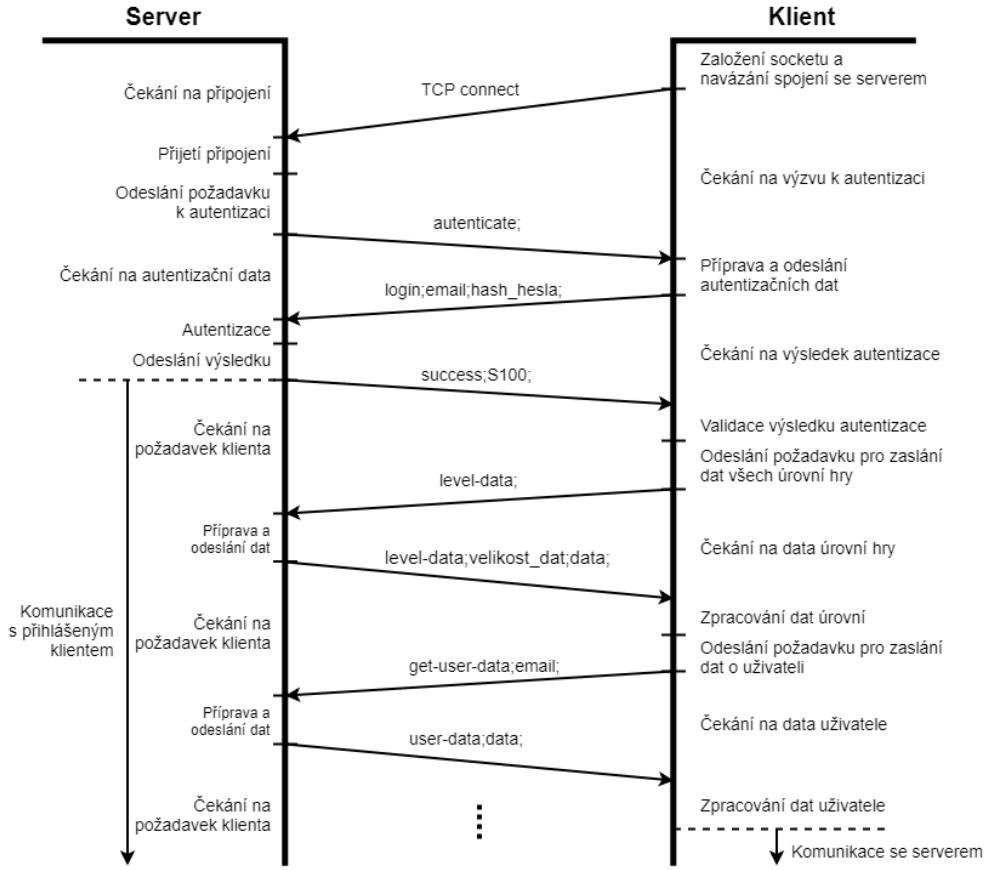
V metodě `connect()` je vytvořena instance vlákna `ServerConnectionThread` a je spuštěna v novém vlákně aplikace. Po inicializaci vlákna proběhne kontrola stavu připojení k internetu. Nemá-li zařízení přístup k internetu, odešle se broadcast s chybovou hláškou, která se zobrazí na obrazovce aktivity a vlákno končí. V opačném případě se vytvoří nový socket s hodnotami adresy a portu serveru z formuláře a spustí se jeho metoda `connect()`. Jestliže navázání spojení skončí chybou nebo uplyne časový limit, vlákno končí a hráči se zobrazí chybová hláška na obrazovce aktivity.

Proběhne-li připojení k serveru v pořádku, klient čeká na výzvu k autentizace od serveru, na kterou odpovídá zprávou `login` s přihlašovacími údaji hráče. Před odesláním zprávy `login` vlákno zašifruje heslo hráče. Po jejím odeslání klient čeká na výsledek přihlášení od serveru. Přijde-li od serveru zpráva `error`, vlákno končí a zobrazí se chybová hláška na obrazovce aktivity. Jestliže proběhne přihlášení v pořádku (přijde od serveru zpráva `success`), vlákno se podívá do hodnot v *SharedPreferences*, aby zjistilo, jestli se přihlašuje již někdy v minulosti přihlašovaný uživatel.

Pokud ano, vlákno zkontroluje data o úrovních hráčů v databázi. V případě, že žádná data v databázi nejsou, odešle požadavek pro zaslání dat o úrovních ve hře (`level-data` a čeká na odpověď od serveru. Poté vlákno odešle další požadavek na server, a to požadavek k zaslání dat o hráči `get-user-data`. Po přijetí odpovědi od serveru (`user-data`) vlákno zkontroluje, jestli obsahuje požadovaná data nebo zprávu `error`. V případě chybové zprávy vlákno skončí a zobrazí se chybová hláška na obrazovce aktivity. V opačném případě se data zpracují třídou `UserManager` a vlákno přechází do nekonečné smyčky, kde probíhá další komunikace se serverem. Znázornění tohoto případu přihlášení hráče s žádostí o zaslání dat o úrovních hráči hry můžete vidět na obrázku 5.8.

Pokud se jedná o první přihlášení v aplikaci nebo o přihlášení jiného uživatele vlákno vymaže veškerá data o předchozím uživateli z aplikace (databázi i preference). Poté vlákno pokračuje jako v případě (viz odstavec výše), kde se provede kontrola dat o úrovních hry a žádost o data hráče.

Jedná-li se o opětovné přihlášení na server, které se vyvolává automaticky po úspěšném obnovení spojení, vlákno odešle navíc broadcast signalizující obnovení spojení se serverem.



Obrázek 5.8: Diagram přihlašování hráče do hry s požadavkem pro zaslání dat úrovní ve hře

Aktivace mise

Pro aktivaci mise musí být hráč přihlášen, tudíž aplikace musí být v aktivitě `MainActivity`. Zde se hráč pomocí nabídky *Mise* v navigačním menu přepne do fragmentu `MissionListFragment`. V metodě `onActivityCreated()` tohoto fragmentu se zkontroluje stav připojení a v případě že je aplikace připojena k serveru, fragment odešle skrze službu `ServerConnectionService` požadavek na server (`list-missions`) k zaslání dat o dostupných misí na serveru.

Po přijetí odpovědi s daty o misi (`mission`), komunikační vlákno `ServerConnectionThread` zprávu s daty přepošle pomocí broadcastu součástem aplikace. Tento broadcast se zprávou zpracuje broadcast receiver, který fragment zaregistroval, a poté aktualizuje seznam misí na obrazovce.

Hráč si vybere misi ze seznamu a „tapne“ na ni. Fragment se tímto změní na `MissionDetailFragment`. V metodě `onCreate` fragmentu se zkontroluje, jestli data o misi jsou uložena v databázi. Pokud ano proběhne ještě kont-

rola, jestli mise nebyla na serveru upravována. Pokud data jsou v databázi a jsou stále aktuální, zobrazí se na obrazovce podrobnosti o misi. Pokud ale data o misi v databázi nejsou nebo již nejsou aktuální, fragment pomocí komunikační služby odešle na server požadavek pro zaslání dat o dané misi. Poté co přijdou žádaná data od serveru jsou fragmentu předány, stejně jako v předchozím případě, pomocí broadcastu a registrovaného broadcast receiveru uvnitř fragmentu, který data zpracuje, uloží do databáze pomocí manažerů a zobrazí je na obrazovce.

Jestliže hráč bude chtít danou misi aktivovat, stiskne tlačítko *Aktivovat*. Obsluha tlačítka, pokud je aplikace připojena k serveru, odešle skrze komunikační službu požadavek na server pro aktivaci dané mise (`start-mission`). V opačném případě zobrazí chybovou hlášku.

Server po přijetí zprávy k aktivaci mise deaktivuje hráči jeho aktivní misi, pokud nějakou aktivní má. Poté zkontroluje jestli daná mise existuje, dále jestli již není hráčem splněna nebo jestli hráč má dostatečnou úroveň pro její aktivaci. Jestliže některá z kontrol skončí negativně, odešle zpět zprávu `error` s konkrétní chybou. V případě že kontroly skončí pozitivně, server odešle zprávu `success`. Příjetím této zprávy se aplikace klienta přepne do fragmentu `GameMapFragment` a mise je hráčem aktivována.

Pokud hráč chce misi deaktivovat, použije stejný postup jako pro aktivaci. S tím rozdílem, že v podrobnostech o misi je tlačítko *Deaktivovat*, které odesílá zprávu k deaktivaci (`reset-mission`).

Validace bodu

Pokud má hráč aktivní misi, služba `LocationUpdatesService`, která má za úkol snímat polohu hráče, ve své metodě `onNewLocation()` kontroluje vzdálenost hráče od jednotlivých bodů.

V případě že se hráč k nějakému z nich přiblíží na vzdálenost 25 metrů nebo blíže, služba zkontroluje stav připojení k serveru. Pokud připojení je ztraceno, zobrazí na obrazovce chybovou hlášku. Pokud je spojení aktivní, odešle na server pomocí komunikační služby požadavek k validaci daného bodu (`point-reached`).

Po přijetí této zprávy server zkontroluje, jestli má hráč nějakou aktivní misi a jestli daná mise není již dokončena. Poté proběhne kontrola definice daného bodu v aktivní misi, pokud bod není v misi definován nebo je mise již dokončena, odešle server zprávu `error`. Dále se kontroluje, jestli mise má startovní nebo koncový bod nebo jestli bod je na řadě v definovaném pořadí mise.

Pokud má mise startovní bod, zkontroluje se jestli je již splněný. Pokud

není, server porovná startovní bod s validovaným. Jsou-li stejné, server odešle zprávu `success` a bod označí jako splněný, jinak server odešle zprávu `error`.

Pokud je startovní bod již dosažen nebo v misi není definován, server kontroluje bod koncový. Je-li validovaný bod totožný s koncovým bodem mise, server zkontroluje, jestli jsou všechny ostatní body mise splněny. Pokud ano, odešle zprávu `success` a bod označí jako splněný, jinak odešle zprávu `error`.

Kontrolu pořadí provádí pouze u misí, které mají definované pořadí bodu (*strict-multipoint*). Server zkontroluje, jestli se validovaný bod se shoduje s bodem, který je na řadě. Pokud ano, odešle zprávu `success` a bod označí jako splněný, jinak odešle zprávu `error`.

Poslední kontrolou u bodu je, jestli daný bod již nebyl hráčem dosažen. V takovém případě server odesílá rovněž zprávu `error`.

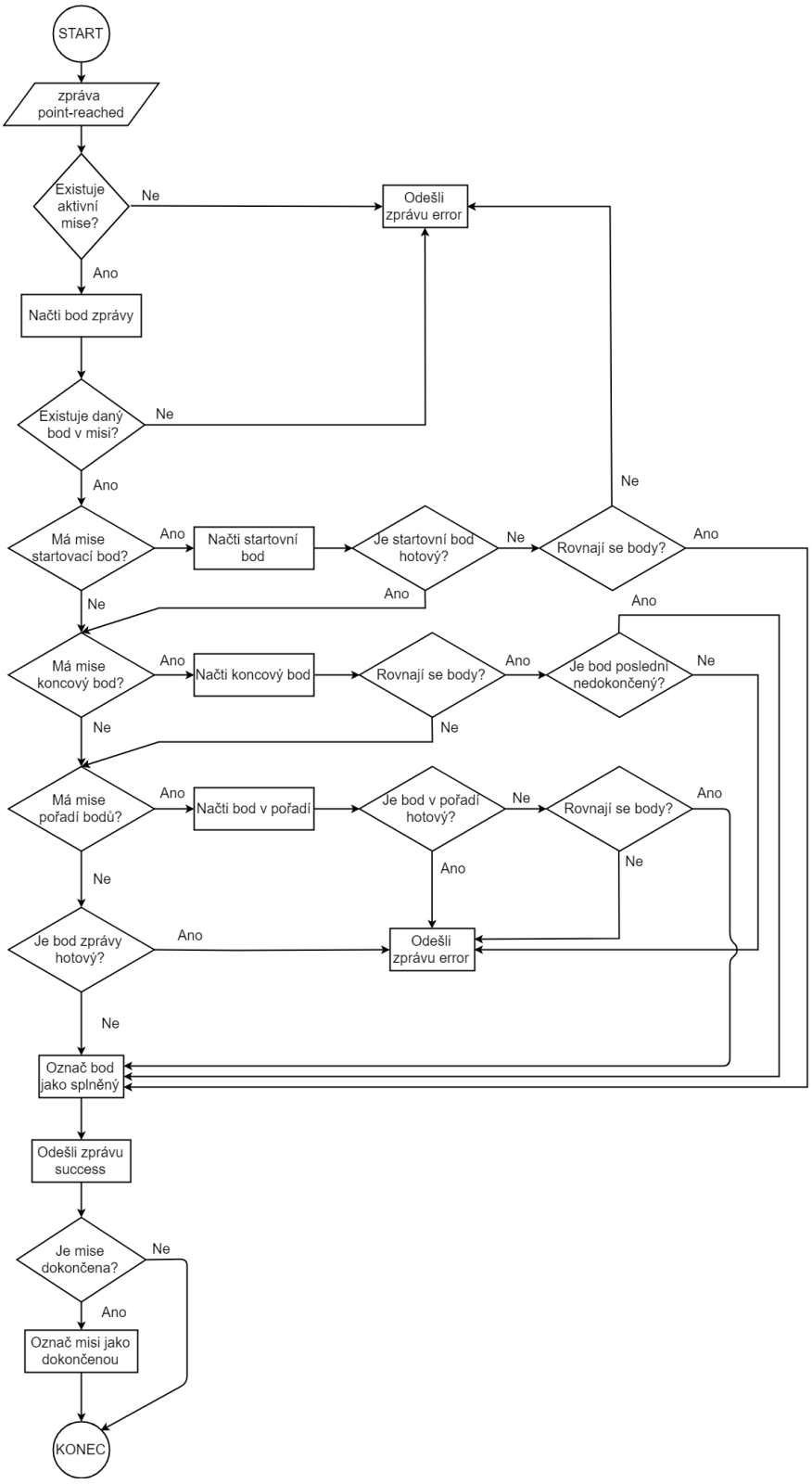
Jestliže kontroly proběhnou v pořádku a bod se označí jako splněný, server navíc zkontroluje, jestli nebyla mise dokončena. Pokud ano, server odešle klientovi zprávu `success` s kódem o splněné mise. Diagram celé validace na serveru naleznete na obrázku 5.9.

Po přijetí výsledku validace na klientu ho komunikační vlákno pomocí broadcastu předává ostatním součástem aplikace. Ty daný výsledek zobrazí na obrazovce hráči buď pomocí hlášky nebo v případě dosažení bodu změní barvu ukazatele daného bodu na herní mapě. Mimo změny barvy ukazatele bodu a zobrazení hlášky, aplikace upozorní hráče o dosažení bodu nebo o dokončení mise také vibrací zařízení.

Zobrazení výsledkové listiny

Zobrazení výsledkové listiny hráčů na klientské aplikaci má na starosti fragment `ScoreFragment`. Ten ve své metodě `onCreateView()` zkontroluje stav připojení k serveru. Je-li spojení aktivní, fragment odešle na sever pomocí komunikační služby požadavek k zaslání dat s výsledky všech hráčů na serveru (`list-score`). Pokud je spojení se serverem ztraceno, zobrazí na obrazovce chybovou hlášku.

Po přijetí požadavku server odešle tutéž zprávu (`list-score`) spolu s daty o výsledcích zpět klientovi. Ten pomocí komunikačního vlákna zprávu s daty předá fragmentu skrze broadcast. Fragment tento broadcast zpracuje registrovaným broadcast receiverem a zobrazí data s výsledky v tabulce na obrazovce.



Obrázek 5.9: Diagram validace bodu na serveru

5.3.3 Přehled zpráv protokolu

Tato kapitola obsahuje popis všech zpráv komunikačního protokolu. U jednotlivých zpráv je uveden jejich předpis, případ užití a případně možné reakce na ně. V předpisu zprávy jsou použity symboly `<>` pro znázornění hodnoty ve zprávě – např. `<e-mail>`. Dále jsou zde také symboly `[]` symbolizující zřetězení daných hodnot za sebe v opakování. Například `[<název>;<zeměpisná_šířka>;<zeměpisná_délka>]` znázorňuje zřetězení hodnot: název bodu, zeměpisná šířka a zeměpisná délka, které se cyklicky opakují podle počtu odesílaných položek, v tomto případě bodů. Symboly jsou použity pouze pro popis, tudíž se v komunikačním protokolu vůbec nevyskytují. Protokol obsahuje pouze hlavičku zprávy a jednotlivé její hodnoty oddělené znakem středníku.

Dále v této kapitole jsou tabulky 5.4 a 5.5 popisující jednotlivé chyby a zdary, které se odesílají komunikačním protokolem. Tabulky obsahují jejich kód a význam.

`alive`

Využívá se při ověřování aktivního spojení mezi serverem a klientem. Server odešle tuto zprávu klientovi a také jí od něj očekává jako odpověď. Pokud klient neodpoví do daného časové limitu, spojení se ukončí.

Předpis:

```
alive;
```

`authenticate`

Odesílá server klientovi jako výzvu k autentizaci hráče. Klient může odpovědět zprávou `login` pro přihlášení hráče nebo zprávou `registration` pro registraci nového uživatele. Bude-li přijata jiná zpráva, server odešle chybu o neočekávané zprávě a odpojí klienta. Zpráva neobsahuje žádné hodnoty.

Předpis:

```
authenticate;
```

`error`

Odesílá server klientovi, nastane-li chybová situace. Zpráva obsahuje jedinou hodnotu, a to kódové označení chyby. Předpis zprávy viz další strana.

Předpis:

```
error;<kód_chyby>;
```

exit

Tuto zprávu může odeslat jak klient, tak server. Pokud je odeslána klientem, znamená to, že se klient odhlašuje ze serveru. Naopak pokud jí odešle sever, klient je ze serveru odpojen („vykopnut“). Zpráva neobsahuje žádné hodnoty.

Předpis:

```
exit;
```

get-mission

Odesílá klient serveru jako žádost o zaslání dat o misi. Mise je ve zprávě identifikována svým názvem, který je předán hodnotou. Server jako odpověď odešle zprávu `mission` obsahující data dané mise nebo zprávu `error` s chybou o neznámé misi.

Předpis:

```
get-mission;<název_mise>;
```

get-user-data

Odesílá klient serveru jako žádost o zaslání dat o hráči. Hráč je ve zprávě identifikován svojí e-mailovou adresou, která je předána hodnotou. Server jako odpověď odešle zprávu `user-data` obsahující data daného uživatele nebo zprávu `error` s chybou o neznámém hráči nebo o neoprávněné žádosti.

Předpis:

```
get-user-data;<e-mail>;
```

level-data

Slouží k odeslání dat o úrovních hry ze serveru na klienta. Zpráva obsahuje počet úrovní hry a všechny jejich číselné označení spolu s potřebnými počty zkušeností k jejich dosažení. Předpis této zprávy naleznete na další straně.

Předpis:

```
level-data;<počet_úrovní>;[<označení_úrovně>;<počet_zkuše-
nosti_k_dosažení_úrovně>];
```

list-missions

Tuto zprávu odesílá jak server klientovi, tak klient serveru. Pokud je odesílána klientem, neobsahuje žádné hodnoty a slouží jako žádost na server k odeslání hlaviček všech misí. Server tuto zprávu využívá jako odpověď na žádost klienta tak, že k ní přidá hodnoty seznamu hlaviček misí. V hodnotách je uložen počet hlaviček misí a všechna jména, počty zkušeností za splnění mise, počty bodů za splnění mise, minimální úroveň pro zpřístupnění mise, data poslední úpravy mise na serveru a typy jednotlivých misí.

Předpis:

klient → server

```
list-missions;
```

server → klient

```
list-missions;<počet_hlaviček>;[<název_mise>;<počet_zkuše-
ností>;<počet_bodů>;<minimální_úroveň_hráče>;<datum_posle-
dní_úpravy>;<typ_mise>];
```

list-score

Zprávu odesílá server i klient. Pokud je odesílána klientem, neobsahuje žádné hodnoty a slouží jako žádost na server k odeslání dat o výsledcích všech hráčů na serveru. Server tuto zprávu využívá jako odpověď na žádost klienta tak, že k ní přidá hodnoty seznamu výsledků. V hodnotách je uložen počet hráčů a všechna jejich jména, e-mailové adresy spolu s jejich nasbíranými body, podle kterých je určeno jejich pořadí ve výsledcích.

Předpis:

klient → server

```
list-score;
```

server → klient

```
list-score;<počet_hráčů>;[<jméno>;<e-mail>;<počet_nasbíra-
ných_bodů>];
```

`login`

Slouží k odeslání autentizačních dat hráče od klienta serveru. Zpráva obsahuje e-mailovou adresu hráče a jeho heslo v zašifrované podobě. Pokud přihlašovací údaje jsou správné server odpoví klientovi zprávou `success`. Jestliže údaje správná nejsou odpoví zprávou `error` s chybou neznámého uživatele nebo nesprávného hesla.

Předpis:

```
login;<e-mail>;<zašifrované_heslo>;
```

`mission`

Odesílá server klientovi jako odpověď na žádost o zaslání dat o misi. Zpráva obsahuje hodnoty pro misi a pro body mise.

Hodnoty mise jsou: název mise, popis mise, počet zkušeností za splnění mise, počet bodů za splnění mise, minimální úroveň pro aktivaci mise, čas poslední úpravy na serveru. Dále obsahuje start aktivace mise hráčem, stavovou proměnnou o dokončení mise a čas dokončení mise hráčem, pokud misi hráč aktivoval. V případě že hráč misi dosud neaktivovat, jsou data nahrazena *prázdným datem* – popsaný v kapitole 5.2.3 v sekci: Další třídy balíku. Poté následuje typ mise, její časový limit pro splnění a pořadí bodů. Jestliže časový limit nebo pořadí bodů nejsou definovány, jsou nahrazeny hodnotou `null`.

Další část zprávy tvoří hodnoty bodů mise, které obsahují: počet bodů spolu s jejich souřadnicemi, jmény a podrobnostmi . Dále jsou zde stavové proměnné o dosažení daných bodů spolu s datem dosažení a stavovými proměnnými o tom, jestli body jsou startem nebo koncem mise. V případě že hráč ještě neaktivoval misi, je místo data dosažení použit rovněž *prázdný datum*.

Předpis:

```
mission;<název_mise>;<podrobnosti_mise>;<počet_zkušeností>;
<počet_bodů>;<minimální_úroveň>;<datum_poslední_úpravy>;
<datum_aktivace>;<stav_dokončení_mise>;<datum_dokončení>;
<typ_mise>;<časový_limit>;<pořadí_bodů>;<počet_bodů_mise>;
```

```
[<zeměpisná_šířka>;<zeměpisná_délka>;<název_bodů>;<podro-
bnosti_bodu>;<stav_dosažení>;<datum_dosažení>;<je_startem
_mise>;<je_koncem_mise>];
```

mission-names

Zprávu odesílá server i klient. Pokud je odesílána klientem, neobsahuje žádné hodnoty a slouží jako žádost na server k odeslání seznamu jmen všech misí. Server tuto zprávu využívá jako odpověď na žádost klienta tak, že k ní přidá hodnoty jmen misí. V hodnotách je uložen počet misí a všechna jejich jména.

Předpis:

klient → server

```
mission-names;
```

server → klient

```
mission-names;<počet_misí>;[<jméno_mise>];
```

mission-progresses

Tuto zprávu odesílá jak server klientovi, tak klient serveru. Pokud je odesílána klientem, obsahuje pouze hodnotu jména mise a slouží jako žádost na server k odeslání dat výsledků všech hráčů, kteří danou misi aktivovali. Server tuto zprávu využívá jako odpověď na žádost klienta tak, že k ní přidá hodnoty výsledků hráčů. V hodnotách je uložen počet hráčů, kteří danou misí aktivovali. Dále je zde počet bodů mise a jednotlivé hodnoty výsledků. V hodnotách výsledků jsou uvedena: jména hráčů, počet nimi dosažených bodů mise a data dokončení mise. Pokud hráč misi dosud nedohrál, je místo data použita hodnota `null`. V případě že daná mise neexistuje, server odešle zprávu `error` s chybou.

Předpis:

klient → server

```
mission-progresses;<jméno_mise>;
```

server → klient

```
mission-progresses;<počet_hráčů>;<počet_bodů_mise>;[<e-mail
_hráče>;<počet_dosažených_bodů>;<datum_dokončení_mise>];
```

new-experience

Slouží k odeslání nově dosaženého počtu zkušenosti ze serveru na klienta. Pokud hráč dokončí misi, je odeslána tato zpráva aktualizovaným stavem zkušeností a úrovně hráče. Zpráva obsahuje hodnoty: aktuální úroveň hráče, počet zkušeností hráče a počet jeho bodů.

Předpis:

```
new-experience;<úroveň>;<počet_zkušeností>;<počet_bodů>;
```

point-reached

Zprávu odesílá klient serveru jako žádost o validaci dosaženého bodu a obsahuje hodnoty souřadnic dosaženého bodu. Server po validaci bodu odešle její výsledek pomocí zprávy `error` nebo `success` v závislosti na tom, jak validace dopadla.

Předpis:

```
point-reached;<zeměpisná_šířka>;<zeměpisná_délka>;
```

registration

Slouží k odeslání registračních údajů nového hráče od klienta serveru. Zpráva obsahuje uživatelské jméno, e-mailovou adresu hráče a jeho heslo v zašifrované podobě. Server jako odpověď odešle výsledek registrace v podobě zprávy `error` nebo `success` a ukončí spojení s klientem.

Předpis:

```
registration;<jméno_užvatele>;<e-mail>;<zašifrované_heslo>;
```

reset-mission

Zprávu odesílá server i klient. Pokud je odesílána klientem, obsahuje hodnotu jména mise a slouží jako žádost na server k deaktivaci mise. Server vymaže údaje o dané aktivní mise hráče a odešle hráči zprávu `success`. V případě že mise nebyla identifikována nebo mise byla již dokončena, server odešle zprávu `error`. Pokud tuto zprávu odešle server klientovy, znamená to, že hráč byl vyloučen z mise a jeho průběh v misi byl vymazán. Předpis této zprávy naleznete na další straně.

Předpis:

klient → server

```
reset-mission;<jméno_mise>;
```

server → klient

```
reset-mission;
```

server-message

Slouží k odeslání systémové zprávy od serveru klientovi. Zpráva obsahuje jako hodnoty titulek zprávy a obsah zprávy.

Předpis:

```
server-message;<titulek>;<obsah>;
```

start-mission

Slouží k aktivaci mise na serveru. Zpráva obsahuje hodnotu názvu mise. Server na ní odpovídá zprávami `success` nebo `error`. Zprávu `success` odešle pod aktivace mise proběhla v pořádku. Zatímco zprávu `error` odešle v případě že, zadaná mise neexistuje, hráč nemá dostatečnou úroveň nebo mise byla již splněna.

Předpis:

```
start-mission;<jméno_mise>;
```

success

Odesílá server klientovi proběhne-li operace či požadavek v pořádku. Zpráva obvykle obsahuje jedinou hodnotu, a to kódové označení zdaru. Ale v případech úspěšné validace dosaženého bodu nebo úspěšné dokončení mise obsahuje zpráva hodnoty navíc – viz předpis zprávy.

Předpis:
standardní užití:

```
success;<kód_zdaru>;
```

úspěšné dosažení bodu

```
success;<kód_zdaru>;<název_mise>;<zeměpisná_šířka_bodu>;
<zeměpisná_délka_bodu>;
```

úspěšné dokončení mise:

```
success;<kód_zdaru>;<název_mise>;<získané_zkušeností>;
```

`user-data`

Slouží jako odpověď na zprávu `get-user-data` a obsahuje data o uživateli: jméno, e-mail, počet nasbíraných zkušeností, nachozenou vzdálenost v aplikaci, datum registrace uživatele, počet nasbíraných bodů, název aktivní mise (nebo `null` v případě že aktivní misi nemá), počet dokončených misí a nakonec názvy spolu s časy splnění všech splněných misí.

Předpis:

```
user-data;<jméno>;<e-mail>;<úroveň>;<zkušenosti>;<nachozená
_vzdálenost>;<datum_startu>;<skóre>;<název_aktivní_mise>;
<počet_dokončených_misí>;[<název_mise>;<datum_dokončení>];
```

V následující tabulce (5.4) je uveden přehled všech chybových zpráv komunikačního protokolu. Jednotlivé řádky této tabulky tvoří kódové označení chyby a její význam.

Kód	**Význam**
`E001`	Neočekávaná zpráva
`E100`	Nezdařené přihlášení
`E101`	Neznámý uživatel
`E102`	Chybné heslo
`E103`	Nedostatečná oprávnění
`E104`	Odpojen ze severu
`E105`	E-mailová adresa již existuje
`E106`	Server neodeslal výzvu k autentizaci
`E200`	Neznámá mise
`E201`	Mise byla již splněna

Kód	Význam
E202	Bod byl již splněn
E203	Bod v misi neexistuje
E204	Bod není startovacím bodem mise
E205	Bod není koncovým bodem mise
E206	Bod nelze splnit, jelikož existuje jiný nesplněný bod
E207	Některý z předešlých bodů v pořadí nebyl dosažen
E208	Nedostatečná úroveň hráče
E300	Spojení ztraceno
E301	Připojení se nezdařilo

Tabulka 5.4: Přehled chybových zpráv

Tabulka 5.5 obsahuje přehled všech zpráv komunikačního protokolu pro úspěšné operace. Jednotlivé řádky této tabulky tvoří kódové označení zdaru a jeho význam.

Kód	Význam
S100	Přihlášení (autentizace) proběhlo úspěšně
S101	Registrace proběhla úspěšně
S200	Mise byla úspěšně aktivována
S201	Mise byla úspěšně deaktivována
S202	Mise byla úspěšně dokončena
S203	Bod byl úspěšně dosažen
S204	Mise byla úspěšně načtena

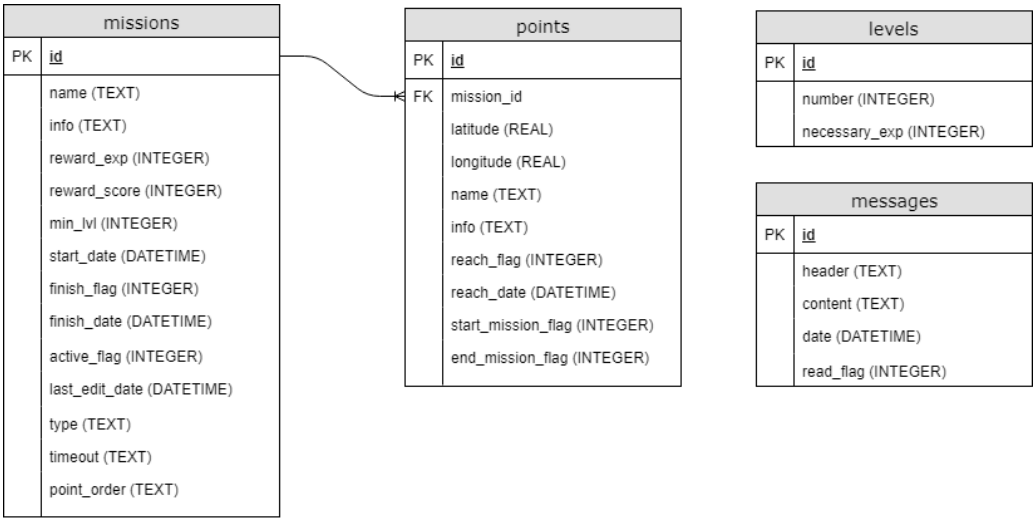
Tabulka 5.5: Přehled zpráv pro úspěšné operace

5.4 Databáze

Tato kapitola je zaměřena na popis databází aplikací klienta a serveru. Budou zde popsány významy databází v aplikacích spolu s jejich ERA modely a přehledy tabulek.

5.4.1 Klient

Aplikace klienta využívá k uchovávání dat hry SQLite databázi. Ta obsahuje tabulky pro ukládání dat o misích a jejich bodů. Dále také obsahuje tabulky pro ukládání dat o úrovních hry a dat systémových zpráv, které nejsou nijak provázány s ostatními tabulkami a slouží pouze jako úložiště těchto dat. Rozhodl jsem využít tyto tabulky, protože manipulace s daty v databázi je pomocí manažerů snadná a všechna data hry jsou uložena na jednom místě. ERA model této databáze můžete vidět na obrázku 5.10.



Obrázek 5.10: ERA model databáze klienta

Přehled tabulek

V této kapitole je uveden přehled všech tabulek databáze aplikace klienta. U jednotlivých tabulek je popsán jejich význam spolu s významem sloupců a jejich případnými podmínkami hodnot. Každá tabulka obsahuje sloupec `id`, ve kterém jsou uloženy primární klíče záznamů tabulky.

Tabulka **`levels`** obsahuje data o úrovních hry. Sloupce této tabulky jsou: `id`, `number` a `nesessary_exp`. Sloupec `number` obsahuje číselné označení úrovně a `nesessary_exp` udává počet zkušeností, které musí hráč nasbírat,

aby dosáhl úrovně vyšší. Oba tyto sloupce obsahují podmínky, že nesmí být do nich uložena hodnota `null` a také že hodnota musí být větší než nula. Navíc hodnota sloupce `number` musí být unikátní.

Tabulka **`messages`** slouží k ukládání dat systémových zpráv od serveru. Sloupce této tabulky jsou: `id`, `header`, `content`, `date`, `read_flag`. Sloupce `header` a `content` obsahují texty zprávy (titulek a obsah), které nesmí obsahovat hodnotu `null`. Sloupec `date` uchovává datum přijetí zprávy a `read_flag` obsahuje pouze hodnotu 0 nebo 1, která říká v jakém stavu daná zpráva je (0 – nepřečteno, 1 – přečteno). Výchozí hodnotou tohoto sloupce je nula.

Tabulka **`missions`** uchovává data misí hry. Obsahuje sloupce: `id`, `name` pro jméno mise a `info` s popisem mise. Hodnoty sloupce pro jména misí musí být unikátní.

Dalšími sloupci jsou `reward_exp` a `reward_score`, které uchovávají odměny za splnění mise: počet zkušeností a počet bodů a sloupec `min_lvl` obsahující číselné označení minimální úrovně hráče, kterou musí hráč dosáhnout předtím, než bude chtít misi aktivovat. Tyto číselné údaje musí být celá čísla větší než nula s výjimkou sloupce pro zkušenosti, ve kterém může být hodnota rovna nule.

Do sloupců `start_date`, `finish_date` a `last_edit_date` se ukládají data: aktivace mise, dokončení mise a poslední úpravy na serveru. Dále sloupce `active_flag`, `finish_flag`, do kterých mohou být uložený pouze hodnoty 0 a 1. Do `active_flag` se ukládá stav o tom, jestli je mise aktivní a do `finish_flag`, jestli byla mise dokončena. Výchozí hodnotou těchto sloupců je nula.

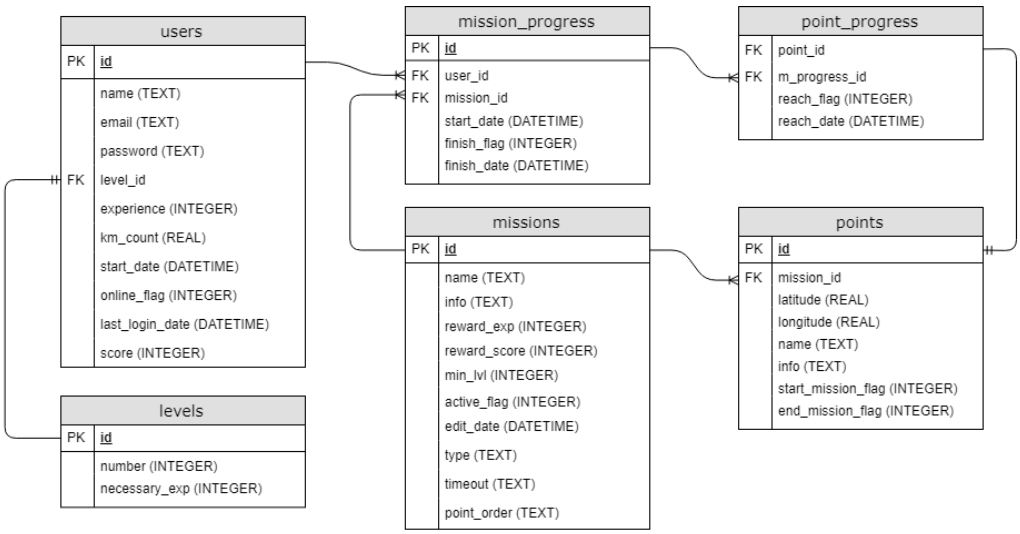
Jako poslední tabulka obsahuje sloupce `type`, `timeout` a `point_order`. Ve sloupci `type` je uložen řetězec určující typ dané mise, který nesmí obsahovat hodnotu `null`. Zbylé sloupce obsahují řetězce s dodatečnými informacemi pro misi v závislosti na jejím typu, případně hodnotu `null`. Má-li mise definované pořadí bodů, tak je toto pořadí uloženo právě ve sloupci `point_order`. Sloupec `timeout` obsahuje časové omezení mise. Mise s časovým omezením v této práci není implementováno a mohla by být předmětem možných rozšíření.

Tabulka **`points`** obsahuje data o bodech mise. Tabulku je tvořena sloupci: `id`, `mission_id` – cizí klíč odkazující na misi daného bodu, `latitude` a `longitude` pro jeho souřadnice, `name` pro jeho název, `info` pro popis (nebo hodnotu `null`) a `reach_date` pro datum splnění. Mimo to tabulka také obsahuje sloupce `reach_flag`, `start_mission_flag` a `end_mission_flag`,

které mohou nabývat pouze hodnot 0 a 1. Ty určují, jestli byl bod splněn nebo jestli je startovním či cílovým bodem mise.

5.4.2 Server

Aplikace server využívá jako datové úložiště SQLite databázi, která je rovněž použita v aplikaci klienta. Databáze obsahuje tabulky pro ukládání dat o hráčích, úrovních hry, herních misí a jejich bodů. Dále také obsahuje tabulky pro ukládání dat o průběhů misí jednotlivých hráčů. ERA model databáze viz obrázek 5.11.



Obrázek 5.11: ERA model databáze serveru

Přehled tabulek

V této kapitole jsou popsány všechny tabulky databáze aplikace serveru. Popis jednotlivých tabulek obsahuje jejich význam, význam jejich sloupců a případné podmínky hodnot sloupců. Každá tabulka obsahuje sloupec `id`, ve kterém jsou uloženy primární klíče záznamů tabulky. Jedinou výjimkou je tabulka `point_progress`, která tento sloupec nemá.

Tabulka **`levels`** obsahuje data úrovní hry. Tabulka je totožná s tabulkou `levels` databáze klienta. Popis této tabulky viz kapitola 5.4.2.

Tabulka **`missions`** slouží k ukládání dat o misích hry. Tabulku tvoří sloupce: `id`, `name` pro jméno mise, které musí být unikátní a `info` pro popis mise.

Tabulka také obsahuje sloupce `reward_exp` a `reward_score` pro počty zkušeností a bodů jako odměny za splnění mise. Dále sloupec `min_lvl`, ve kterém je uloženo označení minimální úrovně, kterou hráč musí mít, aby mohl misi aktivovat. Všechny tyto celočíselné hodnoty musí být větší než nula s výjimkou počtu zkušeností, které mohou být rovny nule.

Poté jsou v tabulce sloupce `active_flag`, `edit_date` a `type`. Sloupec `active_flag` může obsahovat pouze hodnoty 0 a 1, které určují, jestli je mise aktivní na serveru (hráči ji mohou hrát). Sloupec `edit_date` uchovává datum poslední úpravy dat mise a `type` obsahuje řetězec definující typ mise.

Posledními sloupci tabulky jsou `timeout` a `point_order`, které obsahují doplňující data v závislosti na typu mise nebo hodnoty `null`. Má-li mise definované pořadí bodů, tak je toto pořadí uloženo právě ve sloupci `point_order`. Sloupec `timeout` obsahuje časové omezení mise. Mise s časovým omezením v této práci není implementována a mohla by být předmětem možných rozšíření.

Tabulka **`mission_progress`** ukládá průběh plnění misí hráči a je rozkladovou tabulkou mezi `users` a `missions`, která navíc ukládá data aktivace a dokončení mise. Obsahuje sloupce `id` – primární klíč, který je využit v tabulce `point_progress` , `user_id`–cizí klíč do tabulky `users` a `mission_id` – cizí klíč do tabulky `missions`. Dále sloupce `start_date` a `end_date` pro data aktivace a dokončení mise. Posledním sloupcem tabulky je `finish_flag`, který může obsahovat pouze hodnoty 0 a 1 určující, jestli byla mise dokončena (1) či ne (0).

Tabulka **`points`** obsahuje data o bodech mise. Tabulku tvoří sloupce: `id`, `mission_id` – cizí klíč odkazující na misi daného bodu, `latitude` a `longitude` pro jeho souřadnice, `name` pro název a `info` pro popis (nebo hodnotu `null`). Dalšími sloupci této tabulky jsou `start_mission_flag` a `end_mission_flag`, které mohou nabývat pouze hodnot 0 a 1. Ty slouží pro určení, jestli je bod startovním nebo cílovým bodem mise.

Tabulka **`point_progress`** slouží k ukládání průběhů plnění jednotlivých bodů misí hráči. Tato tabulka je jediná tabulka databáze, která neobsahuje sloupec `id`. Jedná se o rozkladovou tabulku mezi `mission_progress` a `points`, kde jsou jednotlivé záznamy tabulky identifikovány dvojicí cizích klíčů. Tabulka navíc obsahuje sloupec `reach_date` pro uložení data splnění bodu a sloupec `reach_flag` obsahující pouze hodnoty 0 a 1, které určují, jestli byl bod splněn (1) nebo ne (0).

Tabulka **users** ukládá data o hráčích. Obsahuje sloupec `id` a `level_id`, který obsahuje cizí klíč do tabulky `levels` určující aktuální úroveň hráče. Dále `name` pro jméno hráče, `email` pro e-mailovou adresu hráče, hodnota toho sloupce musí být unikátní, a `password` pro heslo hráče v zašifrované podobě.

Dalšími sloupci tabulky jsou `experience` a `score`. Ty obsahují celočíselné hodnoty pro počty nasbíraných zkušeností a bodů za splněné mise. Tyto sloupce nesmí obsahovat záporné hodnoty.

Nakonec tabulka obsahuje sloupce `start_date`, `last_login_date`, `online_flag` a `km_count`. Sloupce `start_date` a `last_login_date` obsahují data registrace a posledního přihlášení hráče. Dále do `online_flag` se ukládá hodnota 0 nebo 1 podle toho, jestli hráč je právě přihlášený (1) nebo ne (0). Posledním sloupcem tabulky je `km_count`, který by měl obsahovat reálné číslo s počtem nachozených kilometrů v aplikaci. Tato funkce ale nebyla v této práci implementována, mohla by být však předmětem dalších rozšíření.

6 Testování

V následující kapitole je popsáno testování aplikace. Nejprve jsou popsány zařízení, na kterých byla aplikace testována. Po nich následuje popis jednotlivých testovacích scénářů pro aplikaci spolu s jejich výsledky.

6.1 Testovací zařízení

Testovací zařízení použita pro testování aplikace byl můj osobní stolní počítač, mobilní telefon Asus Zenfone 5 a mobilní telefon Samsung Galaxy J5. Přehled s informacemi o zařízeních naleznete v tabulce 6.1.

K tomu aby server byl na stolním počítači viditelný mimo lokální sít bylo nutné na domácím routeru nastavit směrování portu (*port forwarding* a nastavit pravidla firewallu operačního systému.

Stolní počítač	
Operační system:	Windows 10 Home (64bit)
CPU:	Intel Core i5-4570 (3.20 GHz)
RAM:	8 GB
Rozlišení obrazovky:	1920 x 1080
Java:	jdk v.1.8.0_162
Asus Zenfone 5 A500KL	
Operační system:	Android 5.0
CPU:	Qualcomm Snapdragon 400 (1.2 GHz)
RAM:	2 GB
Rozlišení obrazovky:	1280 x 720 (5")
Samsung Galaxy J5	
Operační system:	Android 6.0
CPU:	Qualcomm Snapdragon 410 (1.2 GHz)
RAM:	2.00 GB
Rozlišení obrazovky:	1280 x 720 (5,2")

Tabulka 6.1: Přehled testovacích zařízení

6.2 Testovací scénáře

V této kapitole budou popsány testovací scénáře spolu s jejich výsledky, kterými byla aplikace testována na testovacích zařízeních, které jsou uvedeny v kapitole 6.1.

6.2.1 Registrace

Scénář:
Nejprve zapneme aplikaci serveru a spustíme server pomocí volby *Start* v menu nabídce aplikace. Poté zapneme aplikaci na mobilním zařízení, kde není zapnuto připojení k internetu (Wi-Fi, mobilní data).

Na přihlašovací obrazovce vybereme možnost *Zaregistrujte se*. Obrazovka by se měla změnit na registrační formulář. Tento formulář vyplníme validními daty a stiskneme tlačítko `registrovat`. Na obrazovce by se měla zobrazit animace načítání a po jejím zmizení by se měla zobrazit chybová hláška o nezdařeném připojení. Na zařízení zapneme připojení k internetu a tlačítko stiskneme znovu. Aplikace by se měla vrátit zpět na obrazovku s přihlášením a měla by se objevit hláška informující o úspěšné registraci.

Vyhodnocení:
Aplikace reagovala dle očekávání a registrace proběhla úspěšně. Po tomto testovacím scénáři byla registrace testována s nevalidními daty formuláře (nevalidní adresa serveru, nevalidní data hráče) a aplikace rovněž reagovala podle očekávání. Nedošlo k ani k pádu aplikace a ani k uložení špatných dat.

6.2.2 Přihlášení

Scénář:
Po úspěšné registraci nového hráče (viz scénář: registrace), se nacházíme v aplikaci na přihlašovací obrazovce a máme aktivní připojení k internetu a aplikace serveru je spuštěná.

Nyní vyplníme přihlašovací formulář údaji, které jsme zadali v registraci, a stiskneme tlačítko *Přihlásit*. Na zařízení nebylo zapnuté sledování polohy, proto se na obrazovce objevilo dialogové okno pro povolení sledování. V případě mobilního zařízení Samsung se jako první objevilo dialogové okno pro povolení oprávnění k sledování polohy, a poté pro povolení sledování polohy. Po povolení oprávnění a sledování polohy se na obrazovce objevila herní mapa. Tedy přihlášení do hry proběhlo v pořádku.

Vyhodnocení:
Aplikace reagovala, tak jak by měla a přihlášení proběhlo v pořádku. Dalšími kroky testování přihlašování do hry byly: Opětovné přihlášení pro ověření, jestli se objeví znovu dialogová okna (když už je vše pro běh aplikace povoleno). Dále použití nevalidních přihlašovacích údajů, přihlašování k vypnutému serveru, pokus o přihlášení při vypnutém připojení k internetu a nepovolení práv či sledování polohy v dialogových oknech. V těchto scénářích aplikace rovněž reagovala dle definovaného chování.

6.2.3 Aktivace mise

Scénář:
Aplikace je ve stavu, kde hráč je přihlášen do hry a nachází se na obrazovce s herní mapou. Zařízení je připojeno k internetu a k serveru.

Přepneme se do obrazovky se seznamem herních misí pomocí položky *Mise* v navigačním menu. Zde se nám zobrazil seznam všech misí na serveru. Vybereme si misi, která vyžaduje, jako minimální úroveň hráče, úroveň 1 a klikneme na její položku v seznamu. Na obrazovce se nám zobrazí podrobnosti o dané misi spolu s tlačítky *Aktivovat* a *Výsledky*. Stiskneme tlačítko *Aktivovat*, kterým aktivujeme misi. Po úspěšné aktivaci mise se hráči zobrazí obrazovka s herní mapou, ne které jsou vykresleny body mise. Vrátíme se do seznamu misí, kde se námi aktivovaná mise barevně odlišuje od ostatních. Klikneme na ní a v zobrazených podrobnostech se nyní zobrazuje i datum aktivace mise, k názvu mise je přidáno slovo „AKTIVNÍ“ a tlačítko *Aktivovat* se změnilo na *Deaktivovat*.

Vyhodnocení:
Aplikace reagovala dle definovaného chování a aktivace mise proběhla v pořádku. V dalších testech aktivace mise bylo testováno: aktivace mise s definovanou vyšší minimální úrovní hráče, zobrazení seznamu misi při výpadku spojení se serverem, aktivace mise ve výpadku spojení se serverem a pokus o aktivaci již splněné mise. V tomto dodatečném testování aplikace také reagovala dle definovaného chování.

6.2.4 Validace bodu

Scénář:
Aplikace je ve stavu, kde přihlášený hráč aktivoval misi a zařízení má aktivní připojení k internetu a serveru. Na obrazovce je zobrazena herní mapa s body mise. Body mise nemají definované pořadí.

Klikneme na libovolný ukazatel bodu mise na mapě. Nad ukazatelem se nám ukázali podrobnosti o daném bodu. Nyní dojdeme k danému bodu, zařízení během cesty k bodu nemusí mít stále připojení k internetu (k serveru), to je důležité až pro jeho validaci.

Poté, co dojdeme k danému bodu, zařízení zavibruje, ukazatel bodu zmodrá a zobrazí se hláška o splnění bodu. To znamená, že bod byl splněn a hráč může pokračovat k dalšímu bodu mise. V případě že hráč dojde k danému bodu a zařízení nemá připojení k internetu, zobrazí se uživateli chybová hláška, že bod nejde splnit, protože aplikace není připojená k serveru. Po opětovném navázání spojení k serveru, se bod splní. Takto splníme všechny body mise a u posledního splněného bodu se zobrazí navíc hláška o splnění mise a ukazatele bodů zmizí z herní mapy.

Pokud poté přejdeme do nabídky misí, v seznamu se u dokončené mise zobrazí „Splněno“. A v podrobnostech této mise se objeví datum dokončení mise,k názvu mise se přidá „SPLNĚNA“, ukazatele na mapě jsou všechny modré a tlačítko *Deaktivovat* se již nezobrazuje. Navíc v podrobnostech o hráči, které se zobrazí po kliknutí na položku *Zobrazit profil* navigačního menu, se aktualizoval stav počtu zkušeností a bodů hráče a případně se zvýšila úroveň hráče.

Vyhodnocení:
Validace bodu a plnění mise probíhala v aplikaci v pořádku tak, jak bylo definováno. Mimo tento testovací scénář byla testována navíc validace bodů mise s definovaným pořadím, a to jak dodržení daného pořadí, tak jeho porušení (neúspěšné validace).

Dále bylo testováno odhlášení ze hry během plnění mise a následné pokračování v misi po opětovném přihlášením. Aplikace po opětovném přihlášení bez problému pokračovala v rozehrané misi.

Testována byla také deaktivace napůl splněné mise a opětovaná aktivace dané mise – vynulování průběhu mise. Mise byla úspěšně vynulována a mohla být hrána od začátku.

Dalším scénářem byla validace bodu, když je aplikace spuštěna na pozadí zařízení nebo když hráč nachází v jiné obrazovce aplikace než v té s herní mapou. A nakonec vyloučení hráče z mise administrátorem na serveru.

I v těchto testovacích scénářích aplikace reagovala korektně a nedošlo k žádným jejím pádům nebo ke ztrátě dat.

6.2.5 Zobrazení výsledků

Scénář:
Aplikace je ve stavu, kde přihlášenému hráči je zobrazena herní mapa a zařízení má aktivní připojení k internetu a serveru.

Přepneme se do obrazovky s výsledkovou listinou hráčů pomocí položky *Výsledky* v navigačním menu. Zde se nám zobrazila tabulka s výsledky všech hráčů registrovaných na serveru spolu s tlačítky *Aktualizovat* a *Skóre misí*.

Pro pohled na výsledky konkrétní mise stiskneme tlačítko *Skóre misí*. Na obrazovce se zobrazí seznam misí, které jsou na serveru. Kliknutím na položku seznamu se na obrazovce zobrazí tabulka s výsledky hráčů v dané misi.

Vyhodnocení:
Zobrazení výsledkové listiny hráčů na serveru a zobrazení výsledků u konkrétní mise proběhlo v aplikaci v pořádku. Mimo tento scénář byly také testovány scénáře, kde aplikace nebyla připojena k internetu nebo kde u konkrétní mise nebyl uveden žádný hráč, neboť danou misi žádný hráč dosud neaktivoval. I v těchto testovacích scénářích aplikace reagovala dle stanoveného chování.

6.2.6 Založení mise

Scénář:
Aplikace serveru je spuštěna, server hry je off-line a na obrazovce je zobrazena hlavní obrazovka aplikace.

Kliknutím na položku *Create mission* v nabídce menu *Mission* se zobrazí obrazovka s formulářem pro vytvoření nové mise. Vyplníme formulář daty o misi a přidáme ji dva herní body. Poté misi uložíme stisknutím tlačítka *Save*. Poté se zobrazí dialogové okno s hláškou o úspěšném založení mise, obrazovka s formulářem zmizí a v seznamu misí na hlavní obrazovce se zobrazí nová položka se jménem založené mise.

Vyhodnocení:
Založení mise na serveru proběhlo v pořádku a aplikace reagovala tak, jak bylo definováno. Mimo tento testovací scénář bylo také testováno založení mise, když je server on-line. Aplikace reagovala správně a nedošlo k žádným jejím pádům či ztrátě dat mise.

Dále bylo také testováno vkládání nevalidních hodnot do formuláře nebo úpravy již vložených herních bodů do mise. Při testech se neprojevilo žádné nevalidní chování aplikace a nedošlo ke ztrátě dat nebo pádům aplikace.

6.2.7 Odeslání systémové zprávy

Scénář:
Aplikace serveru je spuštěna, server hry je on-line a na obrazovce je zobrazena hlavní obrazovka aplikace. Na serveru jsou on-line hráči.

Kliknutím na položku *Send message to all* v nabídce menu *User* se zobrazí obrazovka s formulářem pro odeslání systémové zprávy všem online hráčům. Vyplníme formulář zprávou a stiskneme tlačítko *Send*. Po odeslání zprávy se zobrazí dialogové okno s hláškou o úspěšném odeslání zprávy.

Po odeslání zprávy se na klientech v panelu aplikace na ikoně obálky objevil odznak s číslem notifikující přijetí nové systémové zprávy. Klikneme na ikonu obálky a na obrazovce se zobrazí seznam se systémovými zprávami. Kliknutím na položku seznamu se na obrazovce zobrazí obsah zprávy.

Vyhodnocení:
Odeslání zprávy v aplikaci serveru, tak její přijetí na klientské aplikaci proběhlo v pořádku a aplikace reagovali tak, jak bylo definováno. Na serveru bylo navíc testováno: nevalidně vyplněný formulář zprávy, pokus odeslat zprávu, když je server off-line nebo pokus odeslat zprávu, když na serveru není nikdo online. I v těchto scénářích aplikace reagovala tak, jak bylo definováno.

6.2.8 Odpojení hráče serverem

Scénář:
Aplikace serveru je spuštěna, server hry je on-line a na obrazovce je zobrazena hlavní obrazovka aplikace. Na serveru jsou on-line hráči.

Kliknutím pravým tlačítkem na položku seznamu on-line hráčů zobrazíme kontextové menu obsahující operace nad hráčem, kde vybereme operaci *Kick player*. Na obrazovce se zobrazí dialogové okno pro potvrzení operace, kterou potvrdíme tlačítkem *OK*. Pote je hráč ze seznamu online hráčů odstraněn a hráč je odpojen.

Po odpojení klienta na serveru se na klientské aplikaci hráče objeví přihlašovací obrazovka s hláškou „Server vás odpojil“.

Vyhodnocení:
Obě aplikace v tomto testovacím scénáři reagovali dle definovaného chování.

7 Návrhy pro rozšíření aplikace

Tato kapitola je zaměřena na popis možných rozšíření aplikací serveru a klienta. Možných rozšíření u jednotlivých aplikací obsahuje nespočet možností, a to od drobných úprav uživatelského rozhraní (vylepšení grafiky, přidání animací apod.) po přidání celých nových funkcionalit – např: nové typy misí, ukládání splněných bodů do zásobníku pro pozdější validaci na serveru nebo třeba validace rychlosti pohybu hráče. Návrhy rozšíření, které bych zde chtěl uvést jsou: nové typy misí, ukládání splněných bodů pro pozdější validaci, vylepšení komunikace nebo vylepšení bezpečnosti.

7.1 Nové typy misí

Jedno z možných rozšíření aplikace je přidání nových typů mise. Možností v tomto směru je mnoho, já bych zde rád uvedl příklady: mise s časovým omezením, mise se skrytými body a mise s body obsahujícími QR kódy.

Mise s časovým omezením je z části již v aplikaci připravena. Databáze aplikací a formulář pro vytvoření mise již obsahuje aplikační logiku pro tento typ mise. Jediné co v aplikaci chybí, je logika validace této mise a prvky UI v klientské aplikaci (zobrazení odpočtu apod.). Jedná se o typ mise, ve kterém se od data aktivace mise odpočítává časový limit, do kterého musí hráč misi splnit. Tento typ by se dal kombinovat i s ostatními typy mise – například s misí s definovaným pořadím bodů.

Dalším možným typem mise by mohla být mise se skrytými body. Cílem této mise by bylo odhalit pozice všech skrytých bodů v určité oblasti mapy. Příklad mise: V oblasti na mapě (Náměstí republiky v Plzni) nalezněte tři skryté body (Katedrála, Mariánský sloup, budova Magistrátu). Tento typ mise by se dal dále rozšířit o časový limit.

Posledním možných typem mise by mohla být mise s body obsahující QR kódy. Mise by měla stejná pravidla jako typ *multipoint* nebo *strict-multipoint* s tím rozdílem, že by nestačilo dojít k danému bodu. Na místě herního bodu by byl fyzicky umístěn štítek s QR kódem, který by musel na daném místě hráč načíst pomocí aplikace. Ta by například pro validaci mohla použít obsah QR kódu spolu s jeho aktuální polohou.

7.2 Ukládání splněných bodů do zásobníku

Dalším možným vylepšením aplikace je přidání ukládání splněných bodu do zásobníku v aplikaci klienta. Plnění bodu by fungovalo obdobně jako tomu nyní, jediné co by se změnilo by bylo to,že by se přidal mezistav validace bodu.

Hráč by fyzicky došel k bodu, ale nebyl by přihlášen k serveru ani k internetu. Aplikace by místo přímého odesílání bodu k validaci na serveru uložila daný bod do svého zásobníku a změnila by barvu ukazatele bodu například na žlutou. To by hráče upozornilo, že je bod uložen k pozdější validaci.

Poté, co by se hráč připojil k internetu a k serveru, by se odeslaly body uložené v zásobníku k jejich validaci na server. Na základě výsledku dané validace by se body splnily a případně i celá mise.

7.3 Vylepšení a zabezpečení komunikačního protokolu

Posledním rozšířením aplikace, které bych zde chtěl uvést, je vylepšení komunikačního protokolu a zabezpečení posílaných dat. Vylepšení protokolu by se mohlo docílit tak, že by se změnila struktura jeho zpráv. Zprávy by mohly obsahovat řetězec se zápisem JSON objektu, ve kterém by byla uložena hlavička identifikující typ zprávy spolu s daty zprávy. Použitím JSON objektu by se snáz parsovala data ze zprávy. Mimo to by se také snáz odesílala pole dat.

Případné další vylepšení protokolu by mohlo být šifrování zpráv, které by zvýšilo úroveň jeho zabezpečení. Toto šifrování by mohlo být navíc symetrické, ve kterém by se data šifrovala a dešifrovala pomocí hashe hesla uživatele.

8 Závěr

Cílem této bakalářské práce bylo navrhnout a vytvořit aplikaci mobilní hry pro platformu Android, která bude využívat GPS navigaci zařízení telefonu.

Nejprve byl byl proveden rozbor mobilních her a poznávacích aplikací, které využívají sledování polohy zařízení. Z tohoto rozboru a analýzy zadání byly navrženy scénáře hry, její vlastnosti a komunikace v rámci aplikace. Poté byly uvedeny technologie využité v aplikaci.

V další části práce byla popsána implementace aplikací klienta a serveru. Popisy implementací obsahují struktury projektů aplikací, použité architektury, balíky tříd s jejich nejvýznamnějšími třídami, ukládání dat v aplikacích a komunikace mezi nimi. Dále zde byl vysvětlen komunikační protokol využívaný v komunikaci mezi serverem a klientem spolu s příklady komunikací a přehledem zpráv protokolu. A na konci této části byly uvedeny použité databáze aplikací.

Ve zbylých kapitolách byly vypsány testovací scénáře spolu s jejich výsledky a možné budoucí rozšíření práce.

Výsledná aplikace by mohla být využita nejen jako hra pro volný čas, ale i jako poznávací aplikace, kde by se jednotlivé herní body misí nacházely například v blízkosti památek a zajímavých lokalit. Tyto body by rovněž mohly ve svých podrobnostech zobrazovat zajímavosti o daném místu.

Přehled zkratek

Zkratka	Význam
ANR dialog	Application Not Responding dialog – Dialogové okno, které se ptá uživatele, jestli se má ukončit neodpovídající aplikace nebo jestli se má počkat.
API	Application Programming Interface – Jde o sbírku procedur, funkcí, tříd či protokolů nějaké knihovny, jiného programu nebo jádra operačního systému, které může programátor využívat [12].
AR	Augmented reality – Rozšířená realita je vylepšená verze reality, kde přímé nebo nepřímé pohledy na prostředí fyzického prostředí v reálném světě jsou rozšířeny o obrazy generované počítačem [2].
CRUD	C–Create, R–Read, U–Update, D–Delete – Základní operace nad úložištěm dat
ERA model	E–Entita, R–Relace, A–Atribut Je model popisující relace entit včetně popisu jejich atributů. Model je používán k popisu databází.
GPS	Global Positioning System – Globální polohový systém, je vojenský globální družicový polohový systém provozovaný Ministerstvem obrany Spojených států amerických, s jehož pomocí je možno určit geografickou polohu přijímače nacházejícího se kdekoliv na Zemi nebo nad Zemí [13].
GUI	Graphical User Interface – Grafické uživatelské rozhraní.
IDE	Integrated Development Environment – Vývojové prostředí.
JSON	JavaScript Object Notation – JavaScriptový objektový zápis
QR kód	Quick Response kód – kód sloužící k zakódování dat do čtvercové mřížky
TCP	Transmission Control Protocol.

Literatura

[1] *Android Application Development* [online]. Tutorials Point Pvt. Ltd., 2014. [cit. 2018/06/09]. Dostupné z: https://www.tutorialspoint.com/android/android_tutorial.pdf.

[2] *The Ultimate Guide to Augmented Reality (AR) Technology* [online]. Reality Technologies, 2016. [cit. 2018/06/09]. Dostupné z: http://www.realitytechnologies.com/augmented-reality.

[3] Developers, A. *Distribution dashboard* [online]. Android Developers, 2018. [cit. 2018/06/10]. Dostupné z: https://developer.android.com/about/dashboards.

[4] Developers, A. *Vyvíjíme pro Android: Fragmenty a SQLite databáze* [online]. Zdroják.cz, 2018. [cit. 2018/06/10]. Dostupné z: https://www.zdrojak.cz/clanky/vyvijime-pro-android-fragmenty-a-sqlite-databaze.

[5] Developers, A. *Get the last known locationI* [online]. Android developers, 2018. [cit. 2018/06/11]. Dostupné z: https://developer.android.com/training/location/retrieve-current#java.

[6] Developers, A. *App Manifest Overview* [online]. Android Developers, 2018. [cit. 2018/06/11]. Dostupné z: https://developer.android.com/guide/topics/manifest/manifest-intro.

[7] Developers, A. *Permissions Overview* [online]. Android Developers, 2018. [cit. 2018/06/11]. Dostupné z: https://developer.android.com/guide/topics/permissions/overview.

[8] Developers, A. *Services overview* [online]. Android Developers, 2018. [cit. 2018/06/11]. Dostupné z: https://developer.android.com/guide/components/services.

[9] Developers, A. *Sending operations to multiple threads* [online]. Android developers, 2018. [cit. 2018/06/11]. Dostupné z: https://developer.android.com/training/multiple-threads.

[10] Developers, G. *Fused Location Provider API* [online]. Google developers, 2018. [cit. 2018/06/11]. Dostupné z: https://developers.google.com/location-context/fused-location-provider.

[11] Developers, G. *Maps SDK for Android* [online]. Google developers, 2018. [cit. 2018/06/11]. Dostupné z: https://developers.google.com/maps/documentation/android-sdk/intro.

[12] *Applicaiton Program Interface* [online]. Free On-line Dictionary of Computing, 1995. [cit. 2018/06/09]. Dostupné z: http://foldoc.org/Application+Program+Interface.

[13] *Global Positioning System Wide Area Augmentation System (WAAS) performance standard* [online]. Federal Aviation Administration, 2008. [cit. 2018/06/09]. Dostupné z: https://www.gps.gov/technical/ps/2008-WAAS-performance-standard.pdf.

[14] Grant, A. *Android 4 – Průvodce programováním mobilních aplikací*. Nakladatelství Computer Press ve společnosti Albatros Media a.s., 2013. ISBN 978-80-251-3782-6.

[15] *Mobile Operating System Market Share Worldwide* [online]. StatCounter, 2018. [cit. 2018/06/10]. Dostupné z: http://gs.statcounter.com/os-market-share/mobile/worldwide.

[16] Oracle. *Using Prepared Statements* [online]. Oracle, 2017. [cit. 2018/06/19]. Dostupné z: https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html.

[17] Parry, C. *Geocaching* [online]. BBC Oxford, 2009. [cit. 2018/06/09]. Dostupné z: http://www.bbc.co.uk/oxford/content/articles/2009/04/29/geocaching_feature.shtml.

[18] Team, W. *What is bluetooth?* [online]. BBC.co.uk, 2012. [cit. 2018/06/11]. Dostupné z: http://www.bbc.co.uk/webwise/guides/about-bluetooth.

[19] Veness, C. *SHA-256 Cryptographic Hash Algorithm* [online]. Movable Type Scripts, 2017. [cit. 2018/06/17]. Dostupné z: https://www.movable-type.co.uk/scripts/sha256.html.

[20] Ľuboslav, L. *Mistrovství – Android*. Nakladatelství Computer Press ve společnosti Albatros Media a.s., 2017. ISBN 978-80-251-4875-4.

Přílohy

Příloha A: Instalační příručka

Příloha B: Uživatelská příručka

Příloha C: Obsah CD

A Instalační příručka

V této příručce jsou popsány postupy instalací aplikací serveru a klienta.

A.1 Instalace aplikace serveru

Pro instalaci a spuštění aplikace serveru je důležité mít na svém počítači nainstalovanou a nastavenou Javu verze 8 (aplikace byla vyvíjena v Javě verze 1.8.0_162), kterou můžete stáhnout na adrese: `http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`.

Na CD, které bylo v příloze práce, nalezneme a otevřeme adresář `instalace`. Z tohoto adresáře zkopírujeme adresář `server` na disk našeho počítače.

Nyní otevřeme zkopírovaný adresář na disku. V adresáři se nachází podadresář `src` se zdrojovými kódy aplikace, dávkový soubor `cmp.bat` a textový soubor `readme.txt` obsahující nápovědu k instalaci a spuštění aplikace. Dvojklikem spustíme dávkový soubor `cmp.bat` a tím spustíme kompilaci kódu. Po spuštění kompilace se na obrazovce na chvíli objeví příkazová řádka s probíhající kompilací kódu a poté zmizí.

Po úspěšné kompilaci se v adresáři vytvořil nový podadresář `bin` s přeloženými zdrojovými kódy a dalšími soubory aplikace. Dále se zde vytvořily textový soubor `sources.txt`, který byl využit při kompilaci, a dávkový soubor `run.bat`, který slouží ke spuštění aplikace.

Pokud nemáte zařízení s veřejnou IP adresou a chcete, aby byl server přístupný i pro klienty, kteří nejsou v lokální síti, musíte na svém routeru nastavit směrování portů (*port forwarding*). Postup, jak směrování portů nastavit, najdete v manuálu svého routeru.

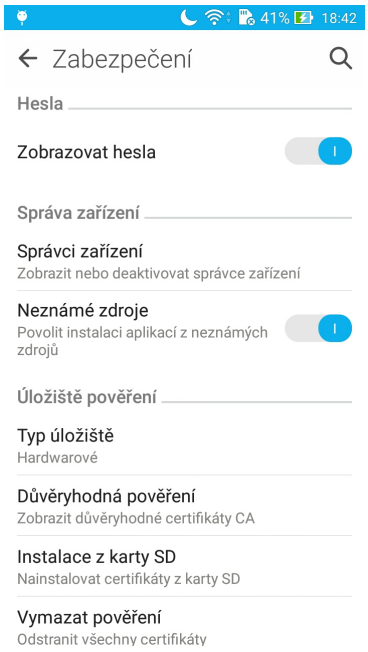
A.2 Instalace aplikace klienta

K úspěšné instalaci a spuštění aplikace klienta je zapotřebí mít mobilní zařízení s operačním systémem Android 4.4 (KitKat) nebo novější. Vývoj společně s testováním aplikace probíhal na OS Android verze 5.0 a 6.0.

Před startem instalace do telefonu musíme v nastavení zařízení povolit instalaci z neznámých zdrojů. Otevřeme si nastavení telefonu a najdeme v seznamu položku *Zabezpečení*. V této části nastavení najdeme položku *Neznámé zdroje* a povolíme ji pomocí přepínacího tlačítka – viz obrázek A.1(a).

Po přepnutí přepínače se objeví dialogové okno pro potvrzení operace, ve kterém jí potvrdíme možností *OK*.

Poté připojíme zařízení k počítači pomocí USB kabelu a nakopírujeme do něj (například do složky `Download`) soubor `zieglerz_bp.apk`, který je uložen na přiloženém CD práce ve složce **instalace/server**. Zařízení odpojíme od počítače a pomocí aplikace *Správce souborů* najdeme nakopírovaný soubor a otevřeme jej. Otevřením souboru se objeví průvodce instalací aplikací od operačního systému, kde postupujeme dále podle pokynů průvodce. Po úspěšné instalaci můžeme nakopírovaný soubor ze zařízení odstranit a aplikaci spustit pomocí její ikony na ploše – viz obrázek A.1(b).



(a) Povolení instalace z neznámých zdrojů v nastavení zařízení

(b) Ikona aplikace klienta na ploše zařízení

Obrázek A.1: Snímky obrazovky zařízení

B Uživatelská příručka

V této příručce je vysvětleno, jak ovládat aplikace serveru a klienta.

B.1 Ovládání aplikace serveru

Tato kapitola uživatelské příručky popisuje, jak spustit a ovládat aplikaci serveru.

B.1.1 Spuštění aplikace

Po úspěšné instalaci aplikace (viz příručka A) se v adresáři aplikace vytvořil dávkový soubor `run.bat`, který slouží ke spuštění aplikace ve výchozím nastavení – s uživatelským rozhraním. Soubor spustíme příkazem:

```
run.bat
```

Chcete-li aplikaci spustit z příkazové řádky operačního systému, bez použití dávkového souboru, otevřete si příkazovou řádku v podadresáři `bin`, kde se nachází přeložené zdrojové kódy aplikace. Poté do příkazové řádky napište následující příkaz:

```
java -cp "lib\sqlite-jdbc-3.21.0.jar;" app.Main
```

Tento příkaz spustí aplikaci ve výchozím nastavením, tedy s uživatelským rozhraním, stejně jako dávkový soubor `run.bat`. Kromě toho lze za příkaz ještě napsat parametry `-nogui` nebo `-help`.

Parametr `-nogui` spustí aplikaci v režimu konzole. V tomto režimu není umožněno vytvářet a upravovat mise, vylučovat a odpojovat hráče nebo posílat systémové zprávy. Server pouze obsluhuje přihlášené hráče.

V konzolovém režimu aplikace lze do konzole zadat příkazy `help` pro zobrazení nápovědy a `exit` pro odpojení všech přihlášených hráčů a ukončení aplikace.

```
java -cp "lib\sqlite-jdbc-3.21.0.jar;" app.Main -nogui
```

Parametr `-help` nespustí aplikaci, ale pouze vypíše do konzole nápovědu ke spuštění aplikace.

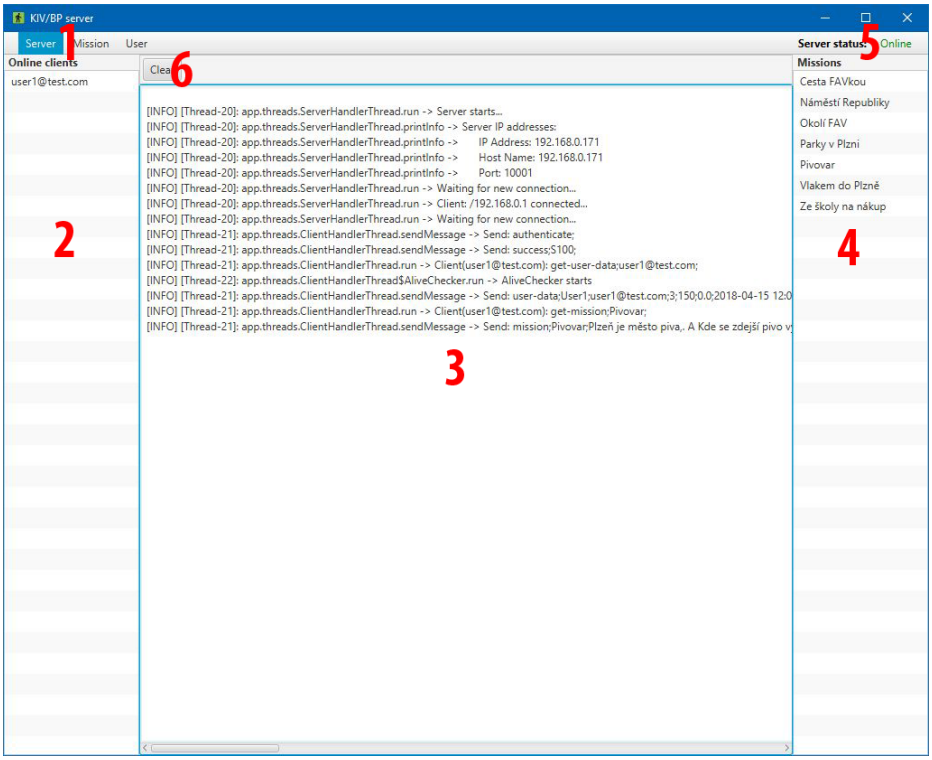
```
java -cp "lib\sqlite-jdbc-3.21.0.jar;" app.Main -help
```

B.1.2 Hlavní obrazovka

Hlavní obrazovka aplikace se skládá z prvků, které jsou vyjmenovány a popsány v tabulce B.1. Obrazovku můžete vidět na obrázku B.10, na kterém jsou jednotlivé prvky označeny číslicemi odkazujícími do tabulky.

#	**Prvek**	**Funkce**
1	Nabídka menu	Jednotlivé položky menu nabídky viz podkapitola B.1.2.
2	Seznam hráčů	Seznam všech on-line hráčů na serveru.
3	Výpis logů	textová oblast, ve které se zobrazují logy serveru.
4	Seznam misí	Seznam všech vytvořených misí na serveru.
5	Status serveru	Ukazatel stavu serveru (*Online*/*Offline*).
6	Tlačítko *Clear*	Tlačítko pro smazaní logů z textové oblasti.

Tabulka B.1: Prvky hlavní obrazovky



Obrázek B.1: Hlavní obrazovka aplikace

Nabídka menu

Nabídka menu, kterou můžete vidět na obrázku B.1 pod číslicí 1, obsahuje položky: *Server*, *Mission* a *User*.

Položka *Server* obsahuje možnosti *Start* pro zapnutí serveru, *Stop* pro vypnutí serveru, *About* pro zobrazení informací o aplikaci a možnost *Close* pro vypnutí serveru i celé aplikace.

Položka *Mission* v sobě obsahuje pouze jednu možnost a to *Create mission*, která zobrazí novou obrazovku s formulářem pro vytvoření nové mise.

Položka *User* má dvě možnosti, a to *Show list*, která zobrazí v novém okně tabulku se všemi registrovanými uživateli na serveru, a možnost *Send message to all*, která zobrazí v novém okně formulář pro odeslání systémové zprávy všem uživatelům.

Kontextové menu seznamu hráčů

Kliknutím pravým tlačítkem na položku seznamu on-line hráčů se objeví kontextová nabídka pro daného hráče. Tato nabídka obsahuje možnosti: *Show detail* pro zobrazení podrobností o hráči v novém okně, *Send message* pro odeslání systémové zprávy pouze tomuto hráči a *Kick player* pro odpojení hráče ze serveru.

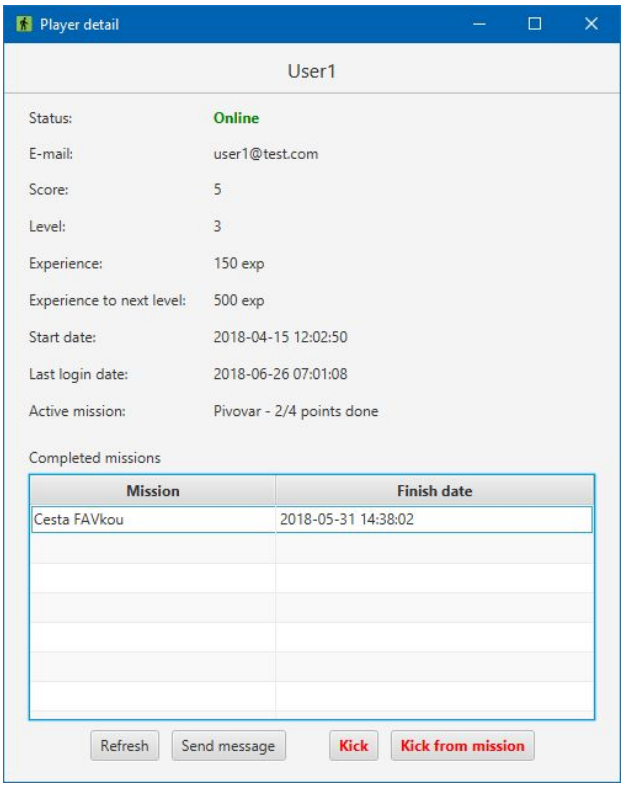
Kontextové menu seznamu misí

Kliknutím pravým tlačítkem na položku seznamu misí na serveru se objeví kontextová nabídka pro danou misi. Tato nabídka nabízí uživateli: *Info* pro zobrazení podrobností o misi v novém okně, *Edit* pro zobrazení formuláře pro úpravu mise a *Delete* pro odstranění dané mise.

B.1.3 Detail uživatele

Pro zobrazení podrobností o přihlášeném uživateli stiskneme dvojklikem na položku hráče v seznamu přihlášených hráčů nebo na položku seznamu klikneme pravým tlačítkem a v kontextové nabídce vybereme možnost *Show detail*.

V novém okně aplikace se zobrazí podrobnosti o přihlášeném hráči spolu s tabulkou všech misí, které hráč dokončil, a tlačítky *Refresh* k aktualizaci dat, *Send message* pro odeslání systémové hráči hráči, *Kick* k odpojení hráče a *Kick from mission* pro vyloučení hráče z aktivní mise.



Obrázek B.2: Podrobnosti o hráči

B.1.4 Vytvoření a úprava mise

Vytvoření nové mise na serveru je provedeno pomocí formuláře (viz obrázek B.3), který zobrazíme pomocí možnosti *Create mission* v položce nabídky menu *Mission* nebo dvojitým kliknutím do seznamu misí na místo, kde není žádná položka.

Formulář se skládá z položek *Name* pro unikátní název mise, *Experience* pro počet zkušeností pro hráče za splnění mise, který musí být celé kladné číslo nebo nula, a *Reward* pro počet bodů pro hráče za splnění mise, který musí být celé kladné číslo větší než nula.

Dalšími položkami jsou: *Minimal level* pro výběr minimální úrovně hráče potřebné k aktivaci mise, *Activate mission* určující, jestli mise je viditelná pro hráče či ne, *Description* pro popis mise a *Type* pro výběr typu mise.

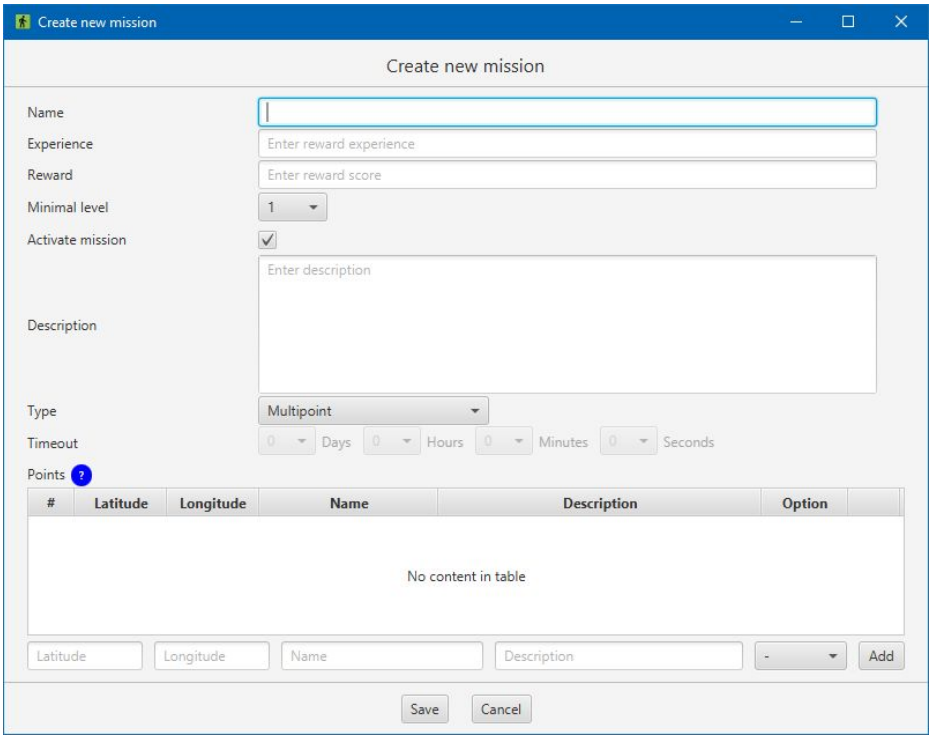
Možné typy misí jsou *Multipoint* a *Strict Multipoint*. *Multipoint* je typ mise, kde není definováno pořadí bodů mise. Naopak u *Strict Multipoint* pořadí bodů mise definováno je.

Poslední součástí formuláře je tabulka s formulářem pro body mise. Ty obsahují položky pro souřadnice bodu, jméno, popis a nastavení bodu. V nastavení bodu se určuje, jestli je bud startovním (*start*), koncovým (*end*) nebo

ani jedním z nich (-). Jedinou nepovinnou položkou formuláře pro body mise je *Description* pro popis bodu. Přidáváním bodů do mise (do tabulky) se provádí pomocí tlačítka *Add*. Pro případné úpravy hodnot již přidaného bodu do tabulky stačí kliknout dvojklikem na hodnotu, kterou chceme upravit. Pro smazání celého bodu z tabulky stiskneme tlačítko *Delete* daného bodu. Další podmínkou pro body mise je, že mezi přidanými body mise musí být vzdálenost větší než 25 metrů.

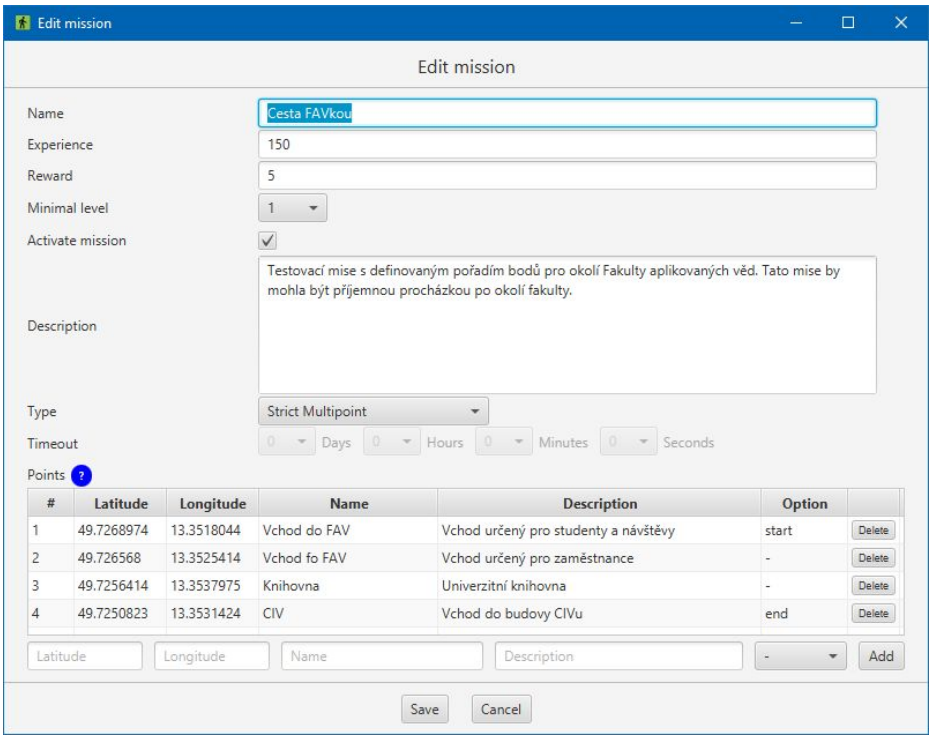
Ve formuláři je položka pro definici časového limit *Timeout*, která je deaktivována a bude zpřístupněna v budoucích verzích aplikace.

Dalšími prvky obrazovky jsou tlačítka *Save* pro uložení bodu a *Cancel* pro stornování vytváření nové mise. Stisknutím tlačítka *Save* se provede validace a pokud proběhne v pořádku, obrazovka zmizí a objeví se dialogové okno s informativní hláškou o úspěšném vytvoření mise. V opačném případě se zobrazí dialogové okno s chybovou hláškou.



Obrázek B.3: Formulář pro vytvoření nové mise

Pokud chceme vytvořenou misi upravit, klikneme na položku dané mise v seznamu misí pravým tlačítkem a v kontextové nabídce vybereme možnost *Edit*. Ta nám otevře formulář pro úpravu mise, který je strukturou totožný s formulářem pro vytvoření nové mise, a vyplní ho daty mise.

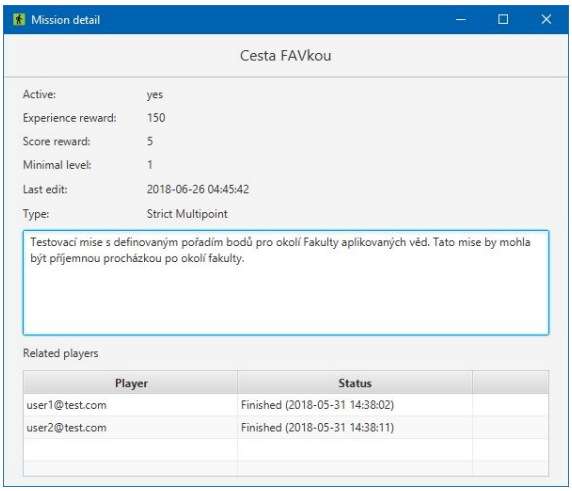


Obrázek B.4: Formulář pro úpravu mise

B.1.5 Detail mise

Pro zobrazení detailu mise (viz obrázek B.5) stačí kliknout dvojklikem na položku mise v seznamu misí nebo vybrání položky *Info* v kontextovém menu u dané položky seznamu misí.

V novém okně aplikace se zobrazí podrobnosti o misi a tabulka s hráči, kteří misi již dohráli nebo ji právě hrají. Pokud misi již hráč dohrál, je u něj ve sloupci *Status* uveden datum dokončení. Jestliže misi hráč právě hraje, je ve sloupci uveden počet splněných bodů z celkového počtu. Kliknutím na řádek tabulky pravým tlačítkem se objeví kontextové menu s jedinou položkou *Kick from mission*, která slouží k vyloučení hráče z mise. Tato operace lze provést pouze u hráčů, kteří mají misi právě aktivní.



Obrázek B.5: Detail mise

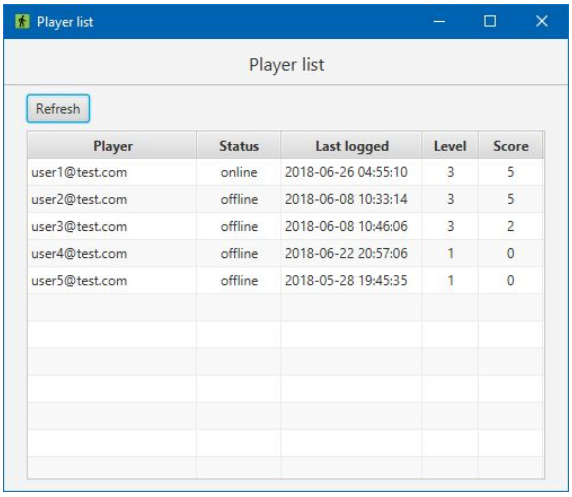
B.1.6 Seznam hráčů a jejich odstranění

Výběrem možnosti *Show list* v položce *User* menu nabídky hlavní obrazovky se zobrazí v novém okně tabulka se všemi registrovanými uživateli na serveru – viz obrázek B.6.

Tabulka obsahuje sloupce: *Player* s e-mailovou adresou uživatele, jestli je on-line nebo off-line, datum posledního přihlášení, aktuální úroveň a počet bodů za splněné mise.

Kliknutím na řádek tabulky pravým tlačítkem se objeví kontextové menu s jedinou položkou *Delete*, která slouží k odstranění hráče ze serveru.

Navíc je v okně seznamu hráčů zobrazeno tlačítko *Refresh* pro aktualizaci dat v tabulce.



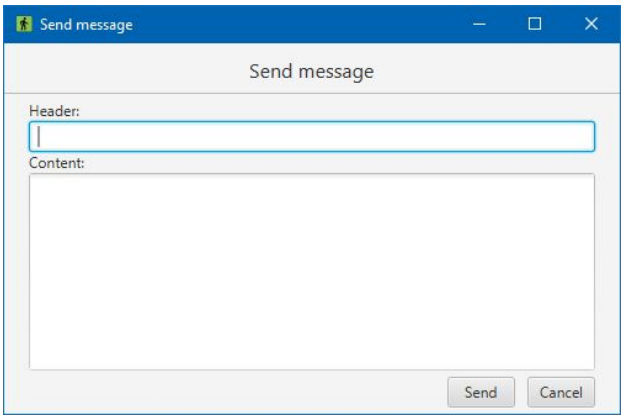
Obrázek B.6: Seznam registrovaných hráčů

B.1.7 Odeslání systémové zprávy

Pro odeslání systémové zprávy všem přihlášeným hráčům vybereme možnost *Send message to all* z položky *User* v navigačním menu hlavního okna aplikace. V novém okně aplikace se zobrazí formulář pro odeslání zprávy, kterou můžete vidět v obrázku B.7.

Formulář obsahuje pouze položky pro titulek a obsah zprávy a tlačítka *Send* k odeslání zprávy a *Cancel* pro zrušení operace.

Mimo odeslání systémové zprávy všem přihlášeným uživatelům, lze také odeslat zprávu jednotlivci. Odeslání systémové zprávy pouze jednomu hráči se skrze kontextovou nabídku v seznamu přihlášených hráčů a v obrazovce detailu hráče – viz kapitola B.1.3.



Obrázek B.7: Formulář pro odeslání systémové zprávy

B.2 Ovládání aplikace klienta

V této kapitole uživatelské příručky je uvedeno jak spustit a ovládat aplikaci klienta.

B.2.1 Spuštění aplikace

Po úspěšné instalaci aplikace do mobilního zařízení, která je více popsána v příručce A, se na ploše zařízení objevila nová ikona aplikace – viz obrázek A.1(b). Klepnutím na tuto ikonu se aplikace klienta spustí.

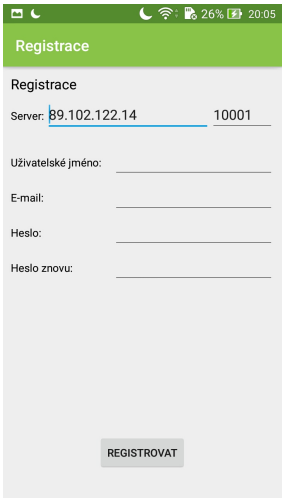
B.2.2 Registrace

Pro registraci nového uživatele do hry zvolíme možnost *Zaregistrujte se* na přihlašovací obrazovce – viz obrázek B.9(a). Tím se na obrazovce zobrazí registrační formulář, který můžete vidět na obrázku B.8(a), obsahující položky: *Server*, *Uživatelské jméno*, *Heslo* a *Heslo znovu* a tlačítko *Registrovat*.

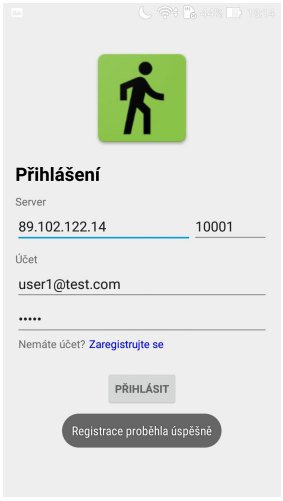
Položka *Server* obsahuje dvě textová pole. První pole (vlevo) je pro IP adresu serveru (IPv4), případně pro jeho *hostname*, a druhý (vpravo) je pro číslo portu, na kterém server poslouchá. Pokud již byla v aplikaci provedena úspěšná registrace nebo úspěšné přihlášení, tyto dvě položky budou již předvyplněny použitým serverem.

Zbylé položky se týkají nového hráče. Hráč musí vyplnit své uživatelské jméno, e-mailovou adresu, která musí být unikátní na serveru a heslo. Položka *Heslo znovu* slouží pouze pro kontrolu zadaného hesla.

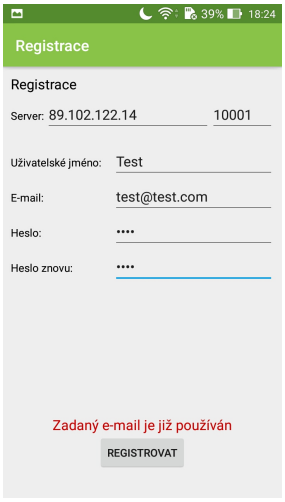
Po vyplnění formuláře stiskneme tlačítko *Registrovat*, tím se spustí validace údajů a registrace. Po úspěšné registraci se aplikace vrátí na přihlášení hráče a zobrazí hlášku o úspěšné registraci – viz obrázek B.8(b). Pokud validace nebo registrace skončí chybou, zobrazí se pod formulářem chybová hláška – viz obrázek B.8(c).



(a) Formulář registrace



(b) Hláška o úspěšné registraci



(c) Chybová hláška

Obrázek B.8: Registrace v aplikaci

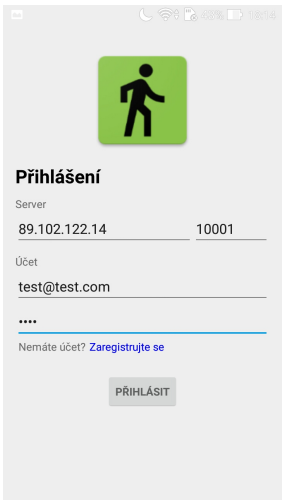
B.2.3 Přihlášení

Po spuštění aplikace se zobrazí úvodní obrazovka aplikace obsahující přihlašovací formulář spolu s volbou pro registraci *Zaregistrujte se* a tlačítkem *Přihlásit* – viz obrázek B.9(a). Přihlašovací formulář obsahuje položky: *Server* a *Účet*.

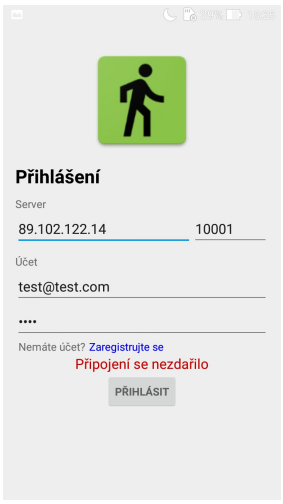
Položka *Server* obsahuje dvě textová pole. První pole (vlevo) je pro IP adresu serveru (IPv4), případně pro jeho *hostname*, a druhý (vpravo) je pro číslo portu, na kterém server poslouchá. Pokud již byla v aplikaci provedena úspěšná registrace nebo úspěšné přihlášení, tyto dvě položky budou již předvyplněny použitým serverem.

Položka *Účet* obsahuje také dvě textová pole. Pole pro hráčovo e-mailovou adresu (nahoře), kterou uvedl v registraci, a pole pro jeho heslo (dole).

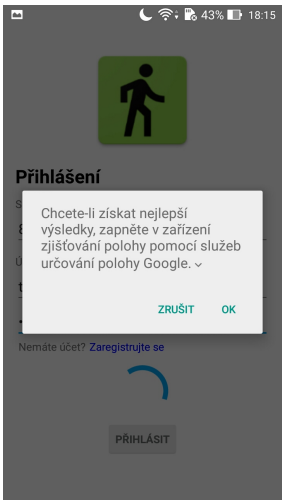
Po vyplnění formuláře stiskneme tlačítko *Přihlásit*, tím se spustí validace údajů, kontrola oprávnění aplikace a přihlášení. Pokud na mobilním zařízení nejsou povolena oprávnění pro sledování polohy nebo není povolené zjišťování polohy, objeví se na obrazovce dialogové okno pro jejich povolení. Pokud přihlášení je úspěšné, zobrazí se hlavní obrazovka hry s herní mapou – viz obrázek B.10. Jestliže validace nebo přihlášení skončí chybou, zobrazí se pod formulářem chybová hláška – viz obrázek B.9(b).



(a) Uvítací obrazovka



(b) Zobrazená chybová hláška



(c) Dialogové okno pro povolení zjišťování polohy

Obrázek B.9: Přihlášení do aplikace

B.2.4 Hlavní obrazovka

Hlavní obrazovka aplikace se skládá z herní mapy a navigačního panelu. Prvky těchto dvou částí obrazovky můžete vidět na obrázku B.10, na kterém jsou označeny číslicemi a dále popsány v tabulce B.2.

#	**Prvek**	**Funkce**
1	Poloha hráče	Zobrazení polohy hráče na mapě.
2	Splněný bod	Znázornění splněného bodu mise na herní mapě. Po kliknutí na něj se objeví podrobnosti o bodu.
3	Nesplněný bod	Zobrazení nesplněného bodu mise na herní mapě. Kliknutím na něj se objeví podrobnosti o bodu.
4	Navigační menu	Kliknutím na ikonu se objeví navigační menu.
5	Systém. zprávy	Zobrazí seznam systémových zpráv a také slouží k notifikaci uživatele o nově příchozí zprávě.
6	Nastavení	Po kliknutí zobrazí nastavení aplikace.
7	Aktuální pozice	Zobrazí polohu hráče v centru mapy.
8	Zoom mapy	Přiblíží nebo oddálí mapa.

Tabulka B.2: Prvky hlavní obrazovky

Obrázek B.10: Hlavní obrazovka aplikace

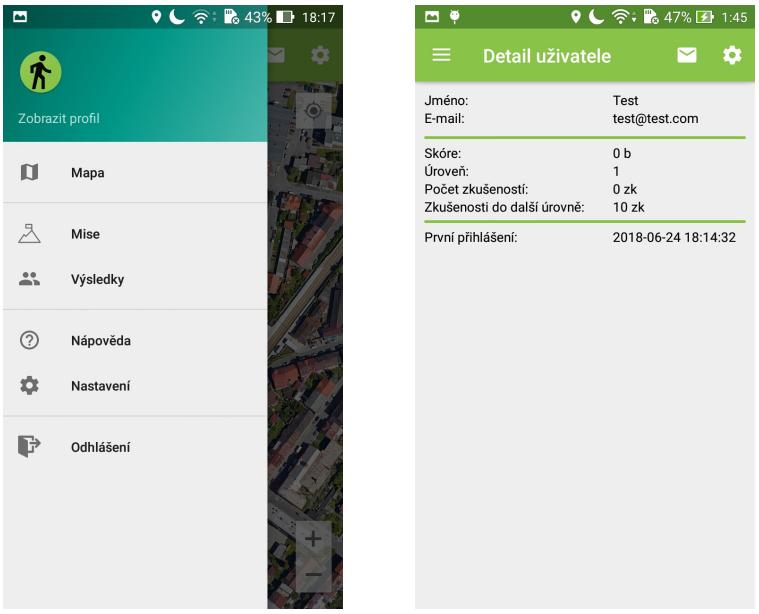
B.2.5 Navigační menu

Navigační menu aplikace, které můžete vidět na obrázku B.11(a) se skládá z hlavičky menu a seznamu odkazů na obrazovky aplikace.

V hlavičce menu je zobrazena ikona aplikace a volba *Zobrazit profil*, která odkazuje na obrazovku o podrobnostech přihlášeného hráče – viz obrázek B.11(a). Položky seznamu spolu s jejích funkcemi naleznete v tabulce B.3.

Prvek	Funkce
Mapa	Zobrazení herní mapy.
Mise	Zobrazení nabídky misí hry.
Výsledky	Zobrazení výsledkové listiny.
Nápověda	Zobrazení nápovědy.
Nastavení	Zobrazení nastavení aplikace.
Odhlášení	Odhlášení hráče ze hry.

Tabulka B.3: Prvky navigačního menu



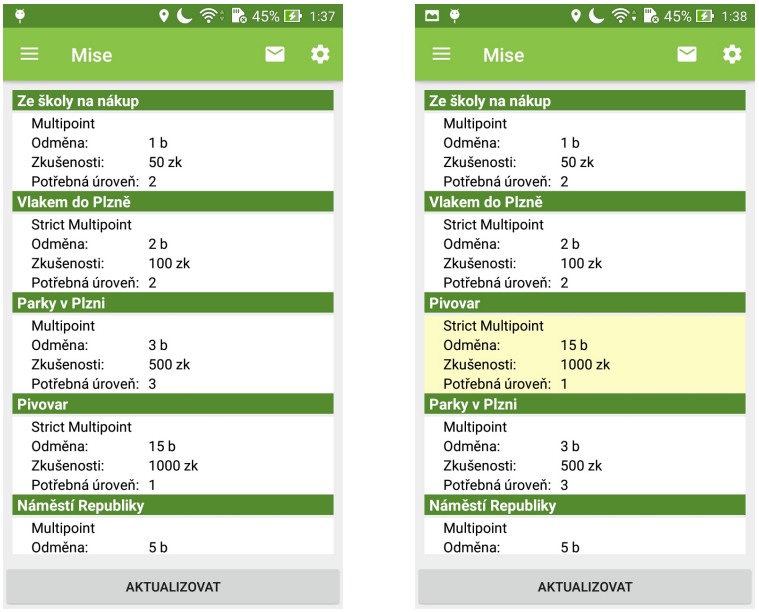
(a) Navigační menu (b) Podrobnosti o hráči

Obrázek B.11: Obrazovky nabídky misí

B.2.6 Aktivace a deaktivace mise

V navigačním menu se pomocí položky *Mise* přepneme do nabídky herních misí – viz obrázek B.12(a). V té je zobrazen seznam misí s jejich stručným popisem a tlačítko *Aktualizovat* pro aktualizaci nabídky.

Položka v seznamu obsahuje název mise, její typ, počet bodů jako odměnu za splnění, počet zkušeností za splnění a potřebná minimální úroveň hráče pro její aktivaci. Položka může být navíc pozadím barevně odlišena od ostatních, v případě že je daná mise právě aktivní – viz obrázek B.12(b). Pokud je mise splněná, je u položky navíc popisek *Splněna*.



(a) Nabídka misí (b) Aktivní mise v nabídce

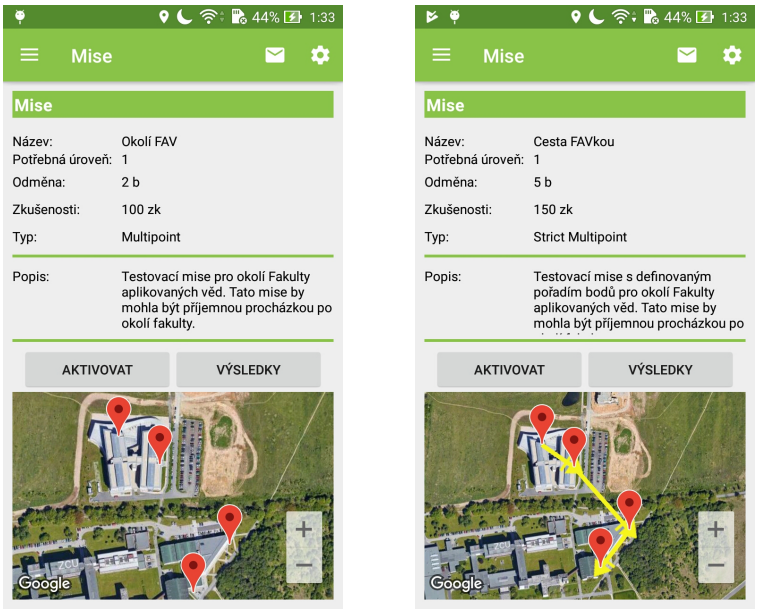
Obrázek B.12: Obrazovky nabídky misí

Kliknutím na položku seznamu se zobrazí podrobnosti mise – viz obrázek B.13(a). Ty oproti položce z nabídky misí navíc obsahují popis dané mise. Také jsou zde na mapě zobrazeny herní body mise. Pokud se jedná o misi typu *Strict Multipoint*, jsou body navíc spojeny šipkami pro znázornění jejich pořadí – viz obrázek B.13(b).

Mimo to jsou zde také zobrazena tlačítka *Aktivovat* a *Výsledky*. Pomocí tlačítka *Aktivovat* lze danou misi aktivovat a pomocí tlačítka *Výsledky* lze zobrazit výsledky všech hráčů, kteří mají misi již dokončenou nebo právě aktivní.

Po aktivaci mise se obrazovka přepne do hlavní obrazovky aplikace s mapou, ve které se zobrazí ukazatele reprezentující herní body mise. Dále v po-

drobnostech aktivní mise se na místo tlačítka *Aktivovat* zobrazí *Deaktivovat*, které slouží k deaktivaci (resetování) mise. Dále se do podrobností přidá položka s datem aktivace. V případě že je mise splněna, se zobrazí pouze tlačítko *Výsledky* a v podrobnostech jsou navíc uvedena data aktivace a dokončení mise.



(a) Detail mise typu Multipoint

(b) Detail mise typu Strict Multipoint

Obrázek B.13: Obrazovky detailu mise

B.2.7 Hraní mise

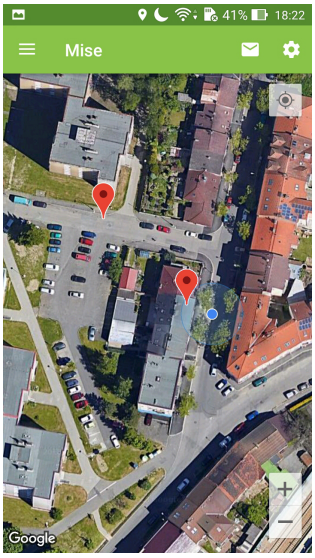
Po aktivaci mise se na mapě hlavní obrazovky aplikace zobrazují ukazatele reprezentující body mise – viz obrázek B.14(a), které je potřeba splnit. Kliknutím na ukazatele zobrazíme podrobnosti o daném bodu – viz obrázek B.14(d).

Body splníme tak, že se k nim přiblížíme na vzdálenost 25 metrů nebo blíž. Pokud se pohybujeme po mapě, tak aplikace nemusí mít připojení k serveru, ale v případě že jsme v blízkosti bodů, připojení potřebujeme, neboť validace splnění bodu se provádí na serveru aplikace. Pokud budeme chtít validovat bod bez připojení k serveru, zobrazí se chybová hláška – viz obrázek B.14(b).

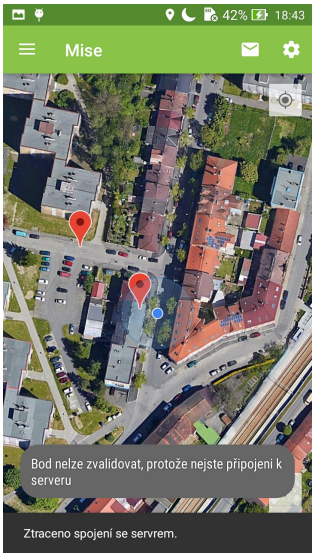
Jestliže splníme bod, jeho ukazatel na mapě změní svou barvu z červené na modrou, což indikuje, že bod byl splněn. Zařízení po splnění bodu také

zavibruje a zobrazí se hláška o splnění bodu – viz obrázek B.14(c). V případě, že validace bodu na serveru skončí chybou, na obrazovce se zobrazí chybová hláška.

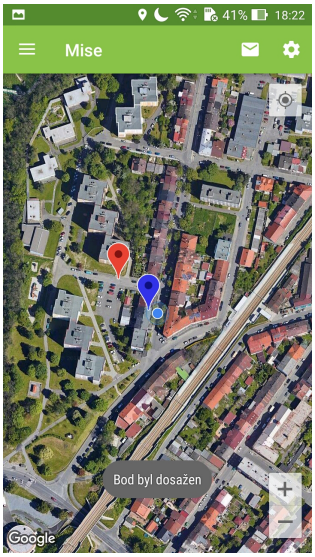
Splníme-li poslední bod mise, mise je dokončena a obdrželi jsme odměnu za její splnění ve formě zkušeností a bodů, kterými se určuje pořadí ve výsledkové listině hráčů.



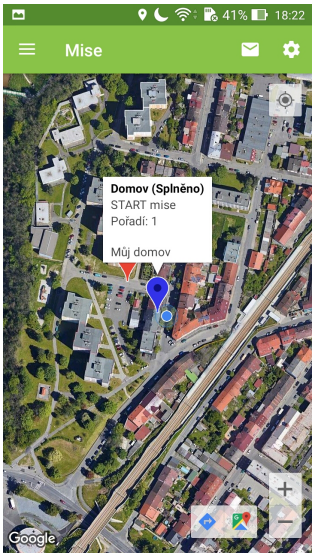
(a) Mapa s ukazateli bodů mise



(b) Chybná validace kvůli ztracenému spojení se serverem



(c) Aktivní mise v nabídce



(d) Podrobnosti o bodu

Obrázek B.14: Obrazovky z plnění mise

B.2.8 Zobrazení výsledků

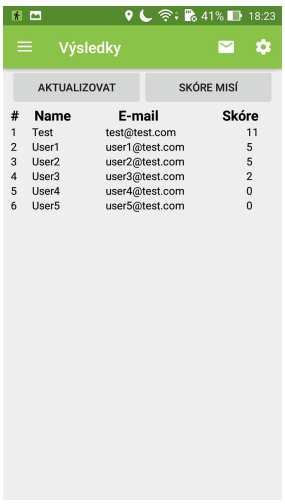
V navigačním menu vybereme položku *Výsledky*. Tím se nám na obrazovce objeví tabulka s výsledky všech hráčů na serveru spolu s tlačítky *Aktualizovat* pro aktualizaci tabulky a *Skóre misí* pro zobrazení výsledků podle mise – viz obrázek B.15(a).

V tabulce jsou uvedeni všichni registrovaní hráči na serveru a jsou seřazeni podle počtu nasbíraných bodů z dokončených misí. Tabulka obsahuje uživatelská jména, e-mailové adresy a počty nasbíraných bodů jednotlivých hráčů.

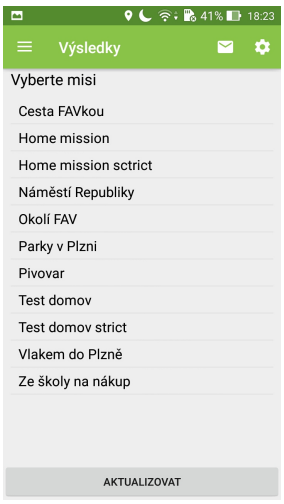
Stisknutím tlačítka *Skóre misí* se na obrazovce zobrazí seznam se jmény všech misí na serveru – viz obrázek B.15(b) a tlačítko *Aktualizovat* pro jeho aktualizaci.

Kliknutím na položku mise v seznamu se na obrazovce zobrazí výsledky pro danou misi – viz obrázek B.15(c). Obrazovku tvoří název mise, tlačítko *Aktualizovat* pro aktualizaci výsledků a tabulka s výsledky.

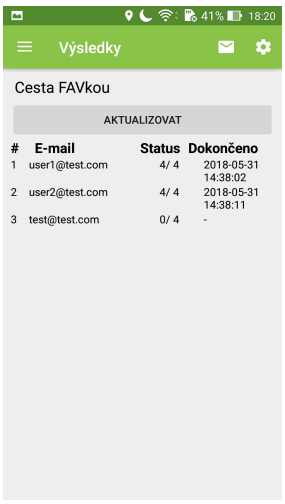
V tabulce jsou uvedeni všichni hráč, kteří mají tuto misi dokončenou nebo jí mají právě aktivní, a jsou seřazeni podle data dokončení mise. Tabulka obsahuje emailovou adresu hráče, jeho status v misi (počet splněných bodů / počet bodů v misi) a datum dokončení mise. Pokud má hráč misi ještě aktivní má místo data dokončení zobrazen znak pomlčky.



(a) Výsledky všech registrovaných hráčů



(b) Seznam misí ve výsledcích



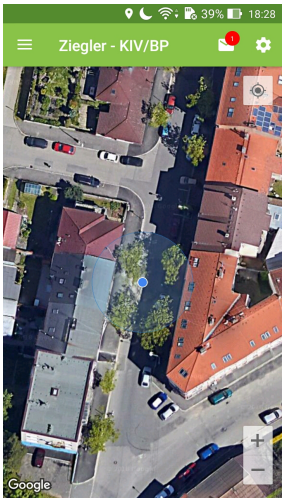
(c) Výsledky dané mise

Obrázek B.15: Obrazovky s výsledky

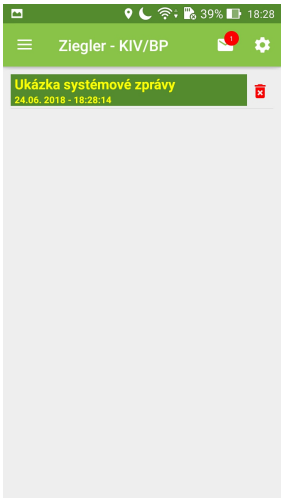
B.2.9 Systémové zprávy od serveru

V navigačním panelu obrazovky je zobrazena ikona obálky symbolizující systémové zprávy. Pokud přijde od serveru nová systémová zpráva, tak se na této ikoně objeví odznak s počtem nepřečtených systémových zpráv – viz obrázek B.16(a).

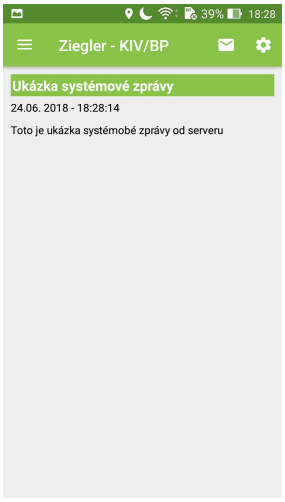
Kliknutím na ikonu se na obrazovce zobrazí seznam všech systémových zpráv. Zprávy, které dosud nebyly přečteny jsou označeny žlutým písmem – viz obrázek B.16(b). Kliknutím na položku zprávy se zobrazí její obsah, jak můžete vidět na obrázku B.16(c), a zpráva se označí jako přečtená.



(a) Nově příchozí systémová zpráva



(b) Seznam všech systémových zprav s jednou nepřečtenou zprávou



(c) Systémová zpráva

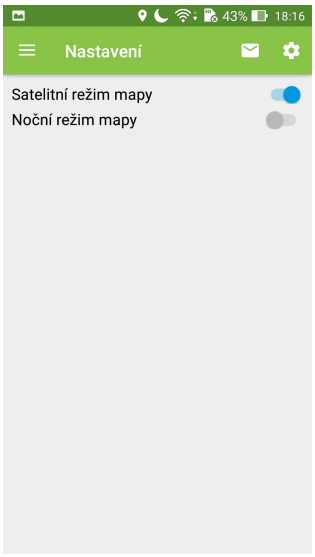
Obrázek B.16: Systémové zprávy

B.2.10 Nastavení aplikace

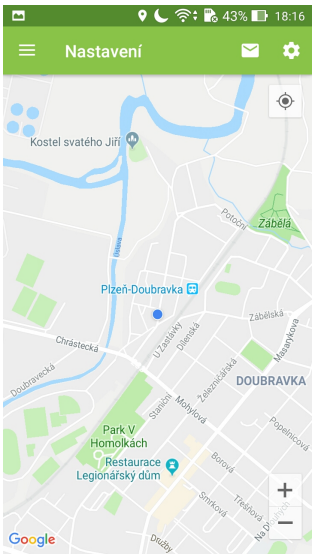
Kliknutím na ikonu nastavení na navigačním panelu aplikace nebo výběrem položky *Nastavení* v navigačním menu se objeví obrazovka s nastavením aplikace – viz obrázek B.17(a).

Nastavení obsahuje pouze dvě položky s jejich přepínači: *Satelitní režim mapy* a *Noční režim mapy*. Ty slouží k nastavení režimu zobrazené mapy. Pokud jsou obě tyto položky vypnuté, mapa je zobrazena ve vektorovém (výchozím) režimu – viz obrázek B.17(b). Zapnutím položky *Satelitní režim mapy* se mapa bude zobrazovat v satelitním režimu, který můžete vidět na obrázku B.17(c). Zapnutím druhé možnosti (*Noční režim mapy*) se bude

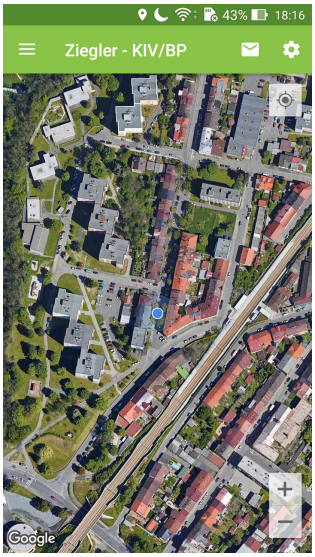
mapa zobrazovat ve vektorovém režimu, který má tmavý motiv – viz obrázek B.17(d). Z těchto možností může být povolena pouze jedna nebo žádná.



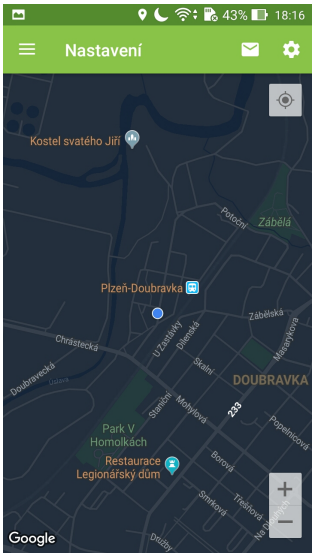
(a) Nastavení aplikace



(b) Výchozí zobrazení



(c) Satelitní režim



(d) Noční režim

Obrázek B.17: Nastavení typu mapy

C Obsah CD

K práci je přiloženo CD s následující strukturou:

Adresář **instalace** – Instalace aplikací serveru a klienta.

Adresář **projekt** – Projekty aplikací serveru a klienta a textu práce.

Soubor **zieglerz_bp.pdf** – Text bakalářské práce.

Soubor **readme.txt** – Obsah CD.